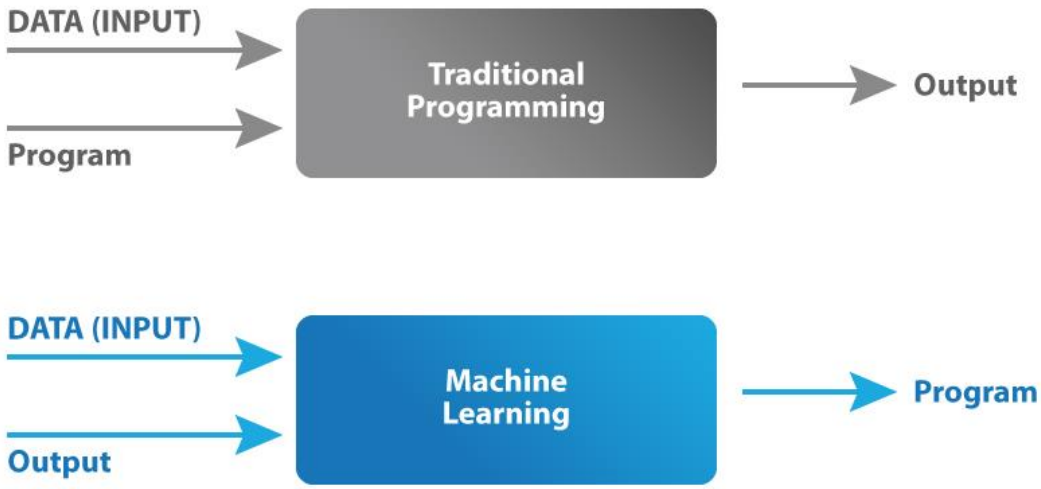
Data Science | 30 Days of Machine Learning | Day - 1

Educator Name: Nishant Dhote
Support Team: **+91-7880-113-112**

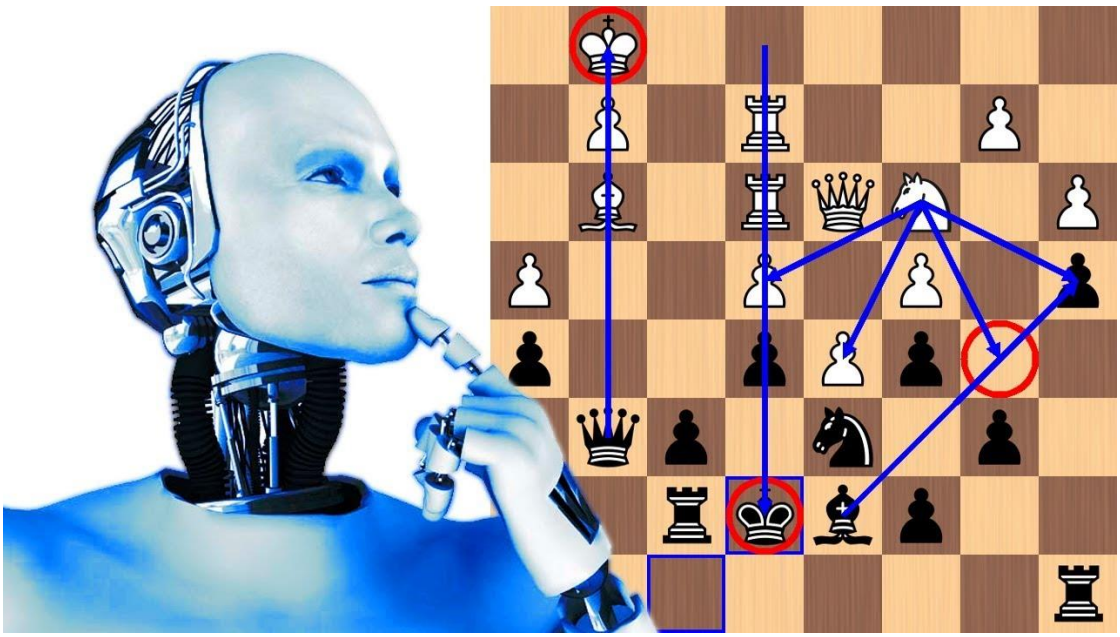
What is Machine Learning:

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed. It evolved from the study of pattern recognition and computational learning theory in artificial intelligence.

Machine learning focuses on the study of algorithms that can learn from data and make predictions (or decisions) based on models built from sample inputs.



Example: Chess Board Game

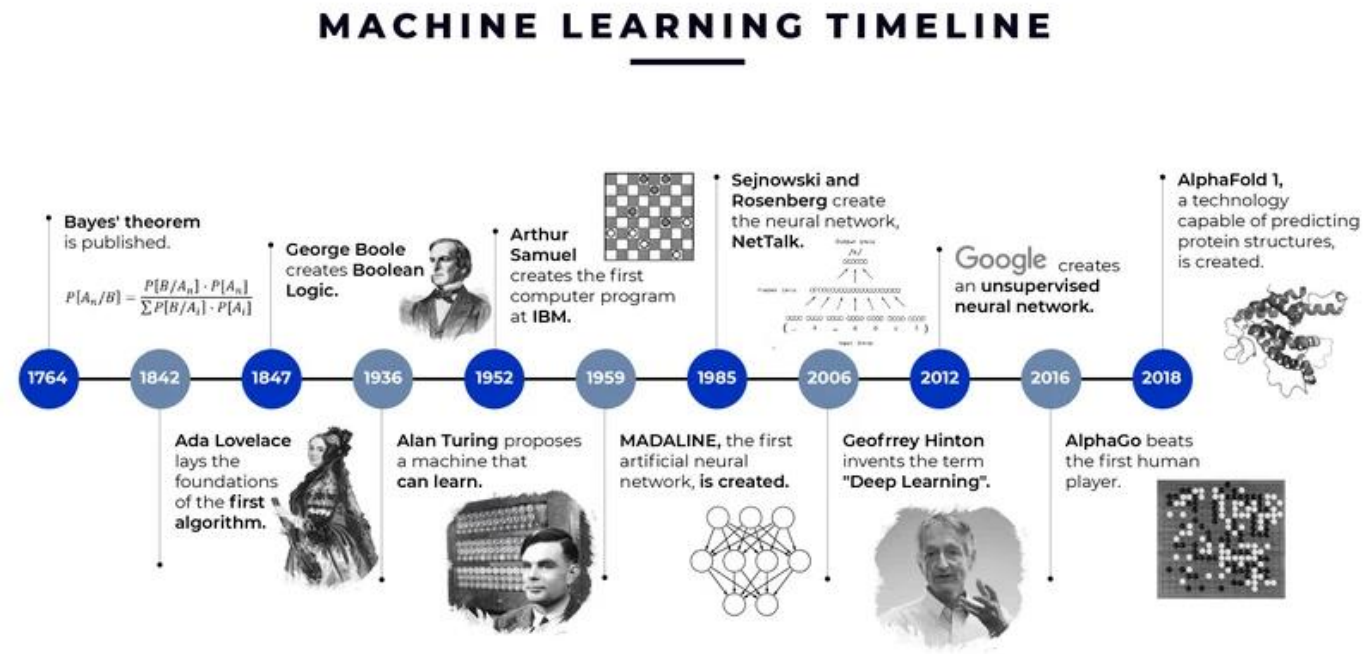


How has Machine Learning evolved? The History of ML

Despite its rise in popularity in the last years, Machine Learning is not a new tool. It has been around for centuries, at least theoretically.

From Thomas Bayes, who in the 18th century laid the foundations of statistics to develop this technology, to the creation of Alpha Go, the first machine to beat a human opponent in the famous game Go, Machine Learning has grown along with humanity.

Here is a brief timeline of the most important events in the history of machine learning

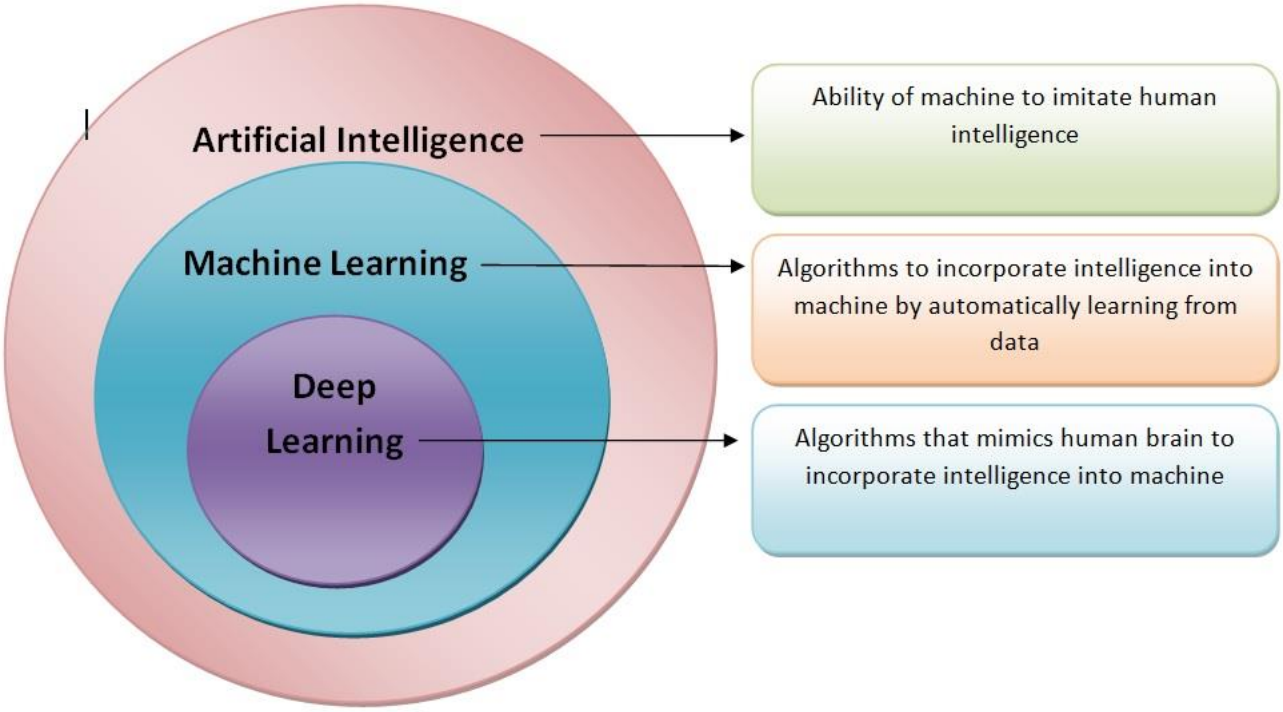


Read Article : https://www.techtarget.com/whatis/A-Timeline-of-Machine-Learning-History

Alpha Go: https://deepmind.google/technologies/alphago/

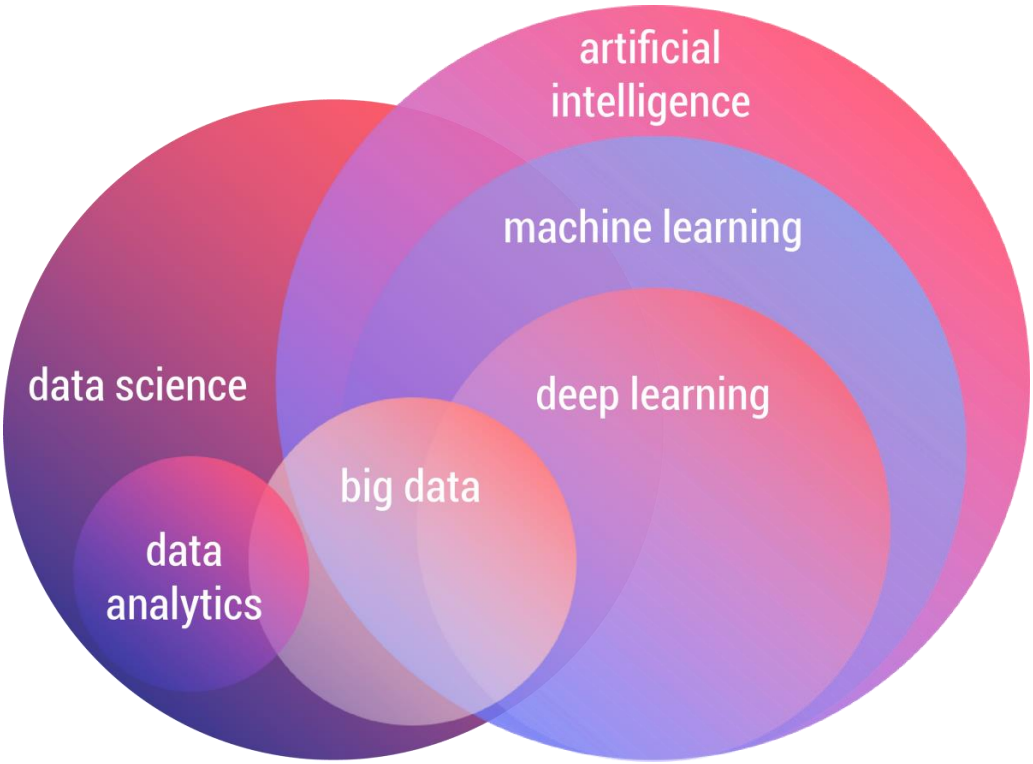
ML vs DL vs AI:

AI simulates human intelligence to perform tasks and make decisions. ML is a subset of AI that uses algorithms to learn patterns from data. DL is a subset of ML that employs artificial neural networks for complex tasks. AI may or may not require large datasets; it can use predefined rules.



Data Science Vs Data Analytics Vs ML/AI/DL

Data science is an umbrella term that encompasses data analytics, data mining, machine learning, and several other related disciplines. While a data scientist is expected to forecast the future based on past patterns, data analysts extract meaningful insights from various data sources.



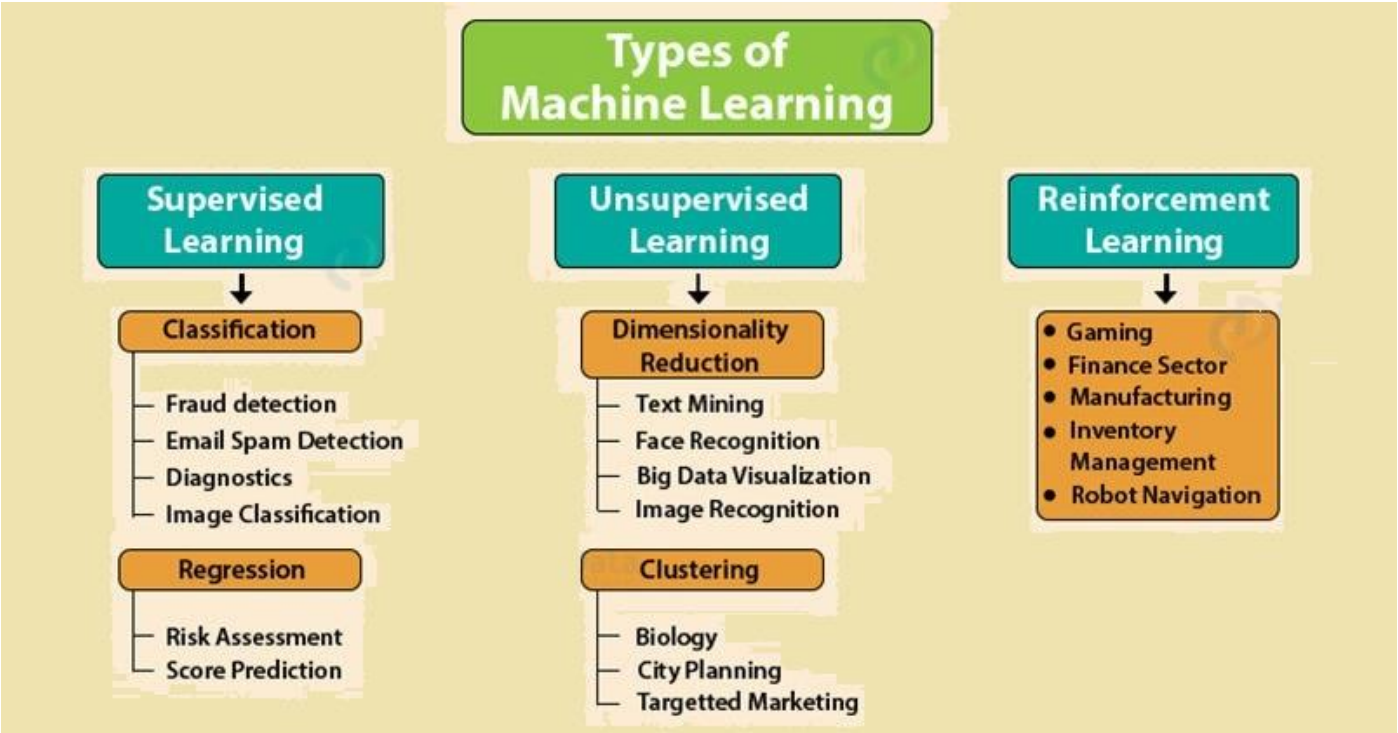
Types of machine learning

What are the 4 types of machine learning?

There are four types of machine learning algorithms: supervised, semi-supervised, unsupervised and reinforcement.

What are the 3 types of machine learning?

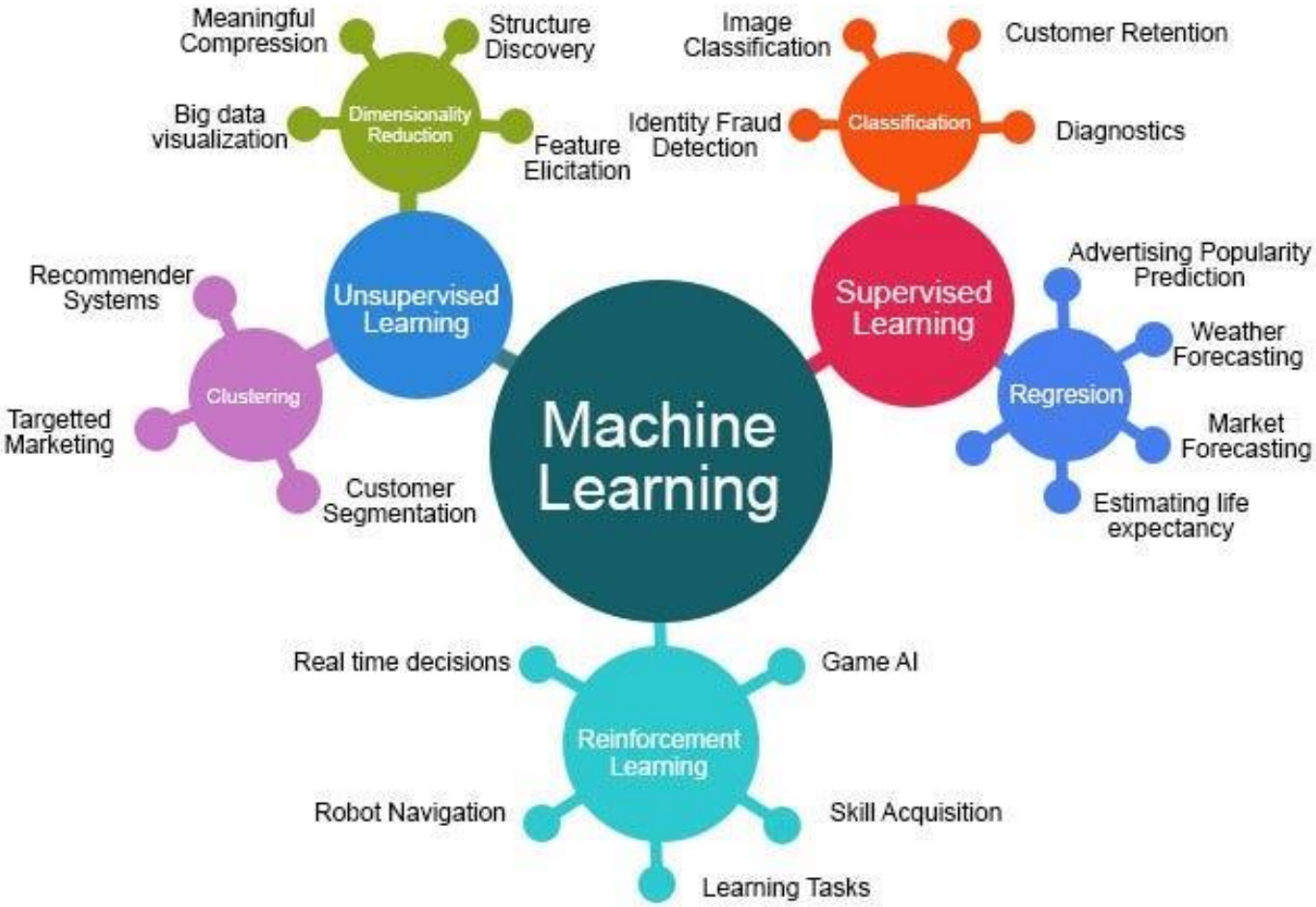
Machine learning involves showing a large volume of data to a machine so that it can learn and make predictions, find patterns, or classify data. The three machine learning types are supervised, unsupervised, and reinforcement learning



Type of machine learning

1. Supervised ML
   - Regression
   - Classification
2. Un Supervised ML
   - Clustering
   - Dimensionality Reduction
   - Anomaly Detection
   - Association Rule Learning
3. Semi supervised ML
4. Reinforcement ML

Machine Learning Algorithms



Hands

1. Supervised Machine Learning:

Supervised learning is defined as when a model gets trained on a "Labelled Dataset". Labelled datasets have both input and output parameters. In Supervised Learning algorithms learn to map points between inputs and correct outputs. It has both training and validation datasets labelled. Let's understand it with the help of an example.

Example: If we have 10,000 Students Data who have placed or some un placed and we want to find the new predictive result output.

Output

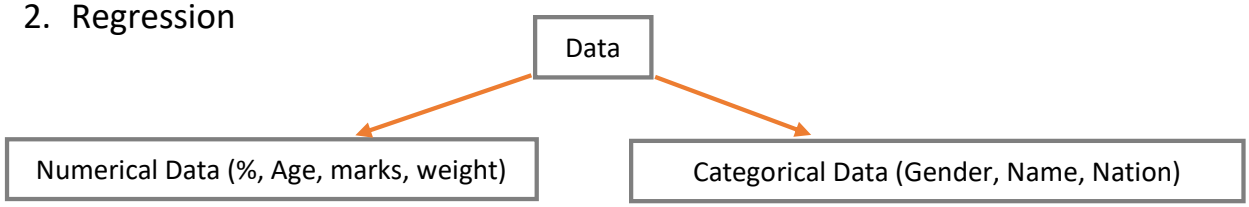
S.No	Student_Name	St_Percentage	St_CGPA	St_Placement	Package
1	Nishant	87%	7.8	Yes	8 LPA
2	Swati	75%	7.2	No	0 LPA
3	Ragini	91%	8.4	Yes	10 LPA
4	Salman	68%	6.5	No	0 LPA
.					
.					
10K	XYZ	00%	0.0	No	0

Input Data

Conclusion: Draw Mathematical Relationship for generate a new result or indicate the prediction.
Input (87%, 7.8) + Output (Learning ML) = (Program) New Result Output

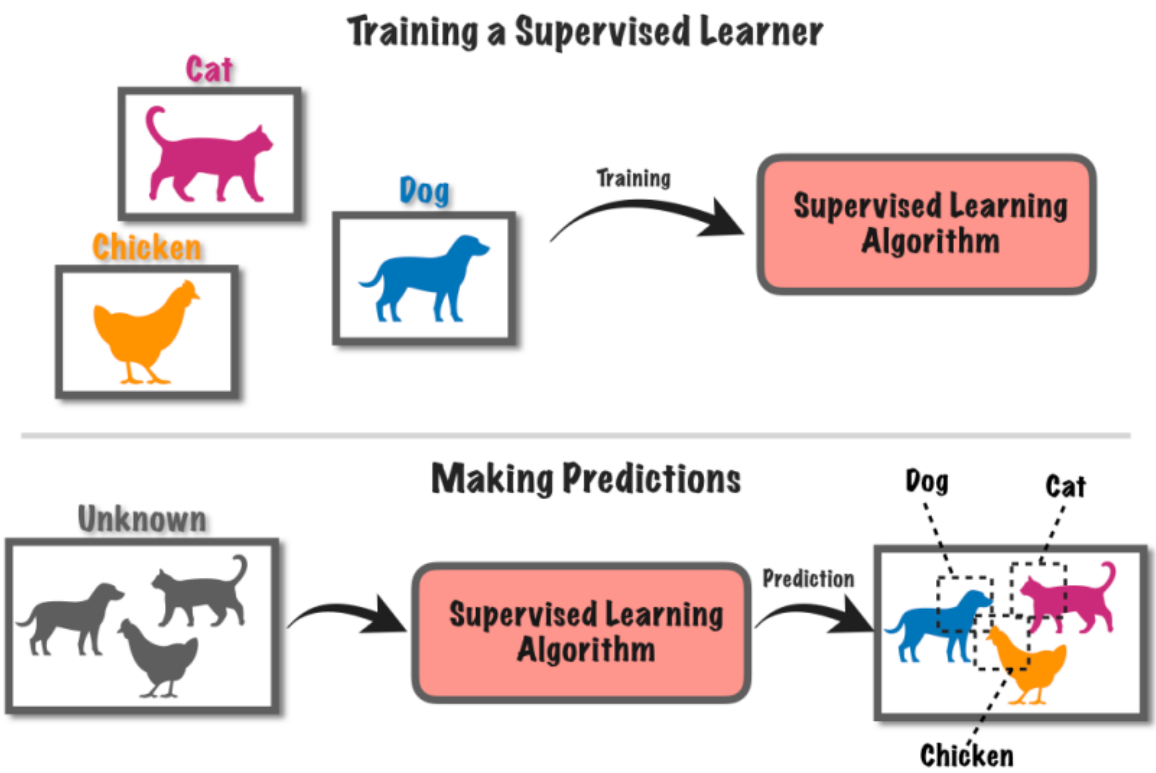
There are two main categories of supervised learning that are mentioned below:

1. Classification
2. Regression



Numerical Output → Regression (Output column is Numerical)
Categorical Output → Classification (Output Column is Categorical)

Example: Consider a scenario where you have to build an image classifier to differentiate between cats and dogs. If you feed the datasets of dogs and cats labelled images to the algorithm, the machine will learn to classify between a dog or a cat from these labelled images. When we input new dog or cat images that it has never seen before, it will use the learned algorithms and predict whether it is a dog or a cat. This is how supervised learning works, and this is particularly an image classification.



Advantages of Supervised Machine Learning

- Supervised Learning models can have high accuracy as they are trained on labelled data.
- The process of decision-making in supervised learning models is often interpretable.
- It can often be used in pre-trained models which saves time and resources when developing new models from scratch.

Disadvantages of Supervised Machine Learning

- It has limitations in knowing patterns and may struggle with unseen or unexpected patterns that are not present in the training data.
- It can be time-consuming and costly as it relies on labelled data only.
- It may lead to poor generalizations based on new data.

Applications of Supervised Learning

Supervised learning is used in a wide variety of applications, including:

- Image classification: Identify objects, faces, and other features in images.
- Natural language processing: Extract information from text, such as sentiment, entities, and relationships.
- Speech recognition: Convert spoken language into text.
- Recommendation systems: Make personalized recommendations to users.
- Predictive analytics: Predict outcomes, such as sales, customer churn, and stock prices.
- Medical diagnosis: Detect diseases and other medical conditions.
- Fraud detection: Identify fraudulent transactions.
- Autonomous vehicles: Recognize and respond to objects in the environment.
- Email spam detection: Classify emails as spam or not spam.
- Quality control in manufacturing: Inspect products for defects.
- Credit scoring: Assess the risk of a borrower defaulting on a loan.
- Gaming: Recognize characters, analyze player behavior, and create NPCs.
- Customer support: Automate customer support tasks.
- Weather forecasting: Make predictions for temperature, precipitation, and other meteorological parameters.
- Sports analytics: Analyse player performance, make game predictions, and optimize strategies.

Industries Helping Hands

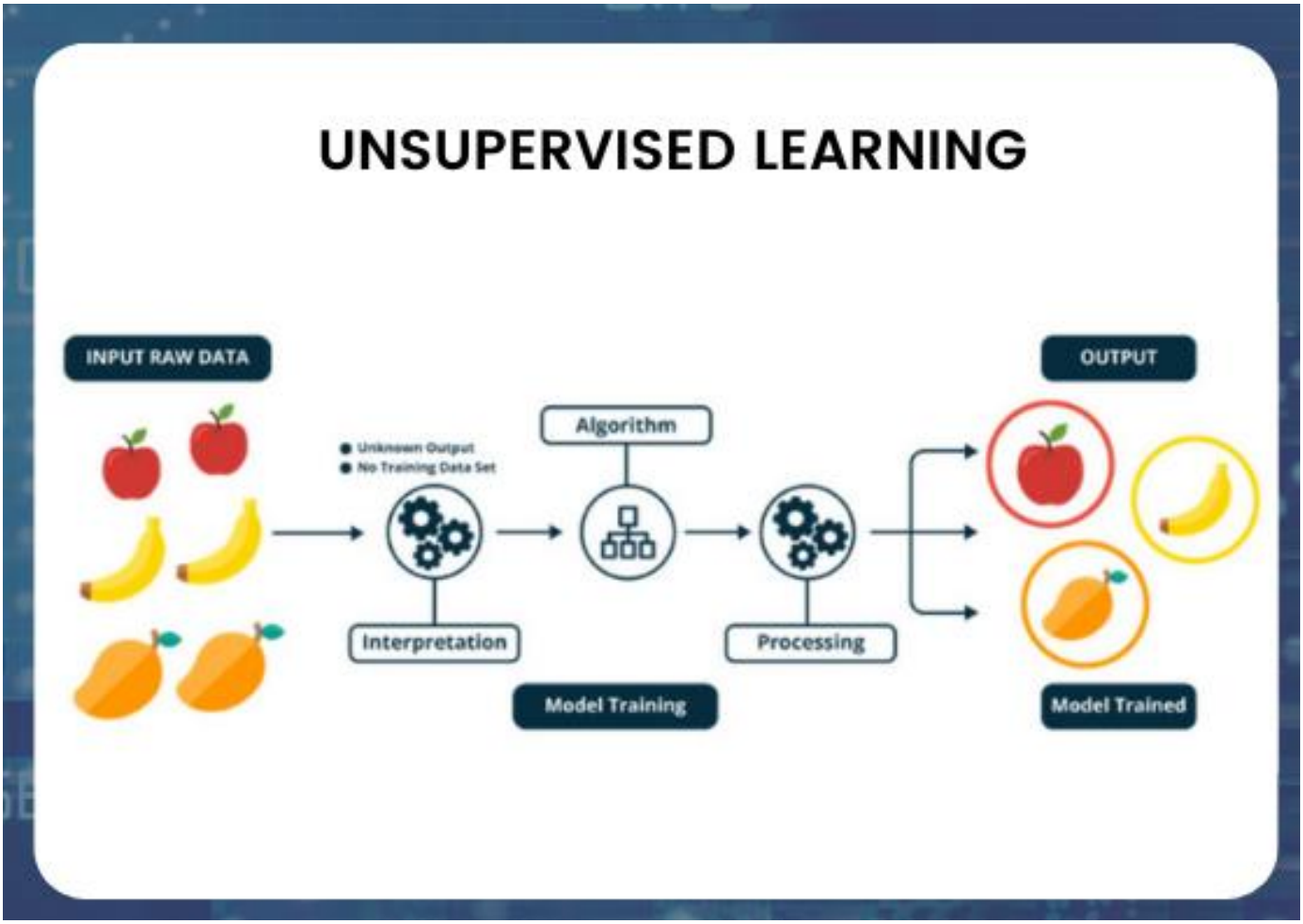
2. Un Supervised Machine Learning:

Unsupervised learning is a type of machine learning technique in which an algorithm discovers patterns and relationships using unlabelled data. Unlike supervised learning, unsupervised learning doesn't involve providing the algorithm with labelled target outputs. The primary goal of Unsupervised learning is often to discover hidden patterns, similarities, or clusters within the data, which can then be used for various purposes, such as data exploration, visualization, dimensionality reduction, and more.

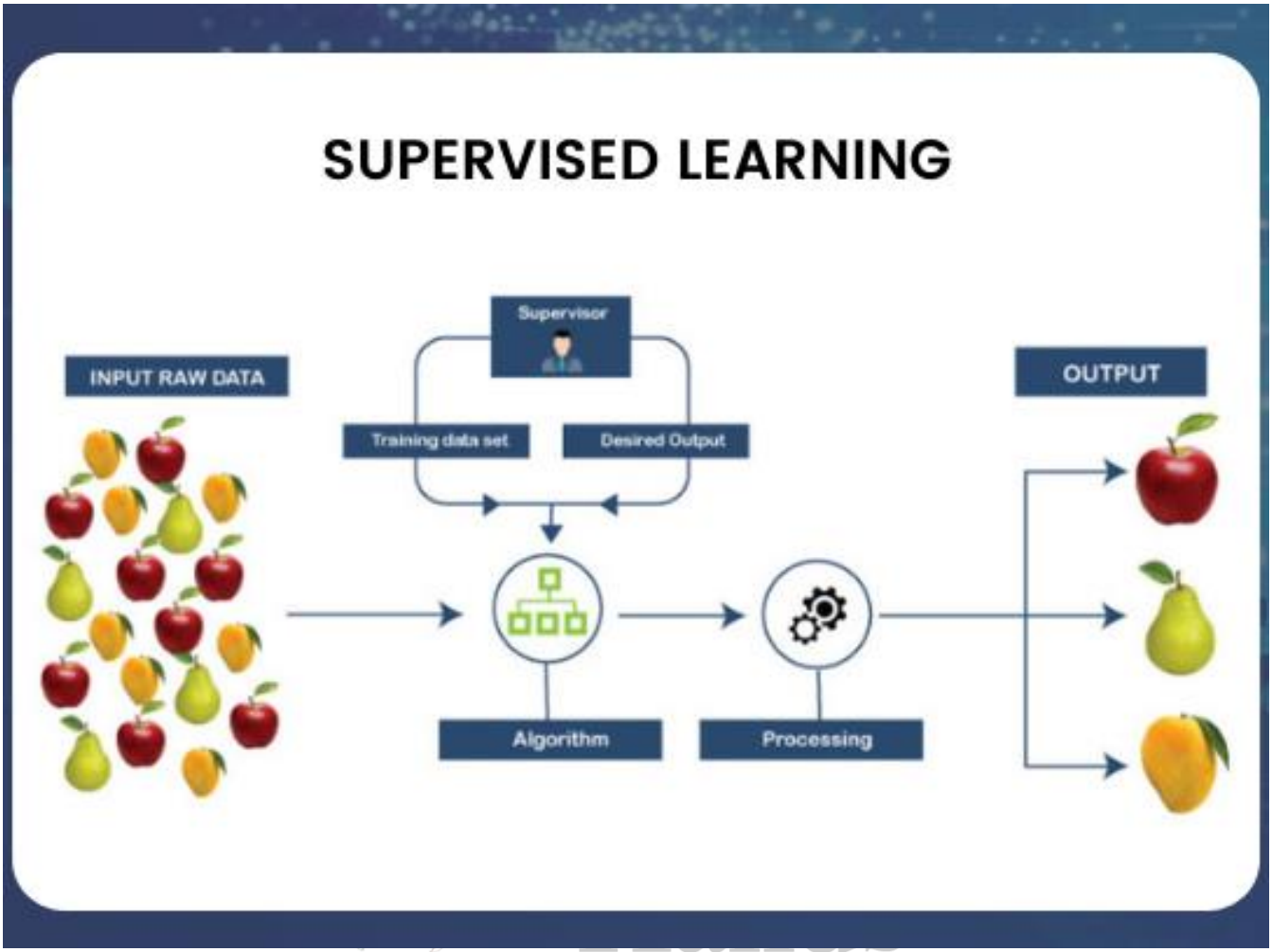
There are two main categories of unsupervised learning that are mentioned below:
1. Clustering
2. Dimensionality Reduction
(3. Anomaly Detection & 4. Association Rule Learning)

Unsupervised Learning algorithms:
Below is the list of some popular unsupervised learning algorithms.

- K-means clustering
- KNN (k-nearest neighbours)
- Hierarchal clustering
- Anomaly detection
- Neural Networks
- Principle Component Analysis
- Independent Component Analysis
- Singular value decomposition



Supervised ML Vs Un-Supervised ML



Advantages of Unsupervised Machine Learning

- It helps to discover hidden patterns and various relationships between the data.
- Used for tasks such as customer segmentation, anomaly detection, and data exploration.
- It does not require labelled data and reduces the effort of data labelling.

Disadvantages of Unsupervised Machine Learning

- Without using labels, it may be difficult to predict the quality of the model's output.
- Cluster Interpretability may not be clear and may not have meaningful interpretations.
- It has techniques such as auto encoders and dimensionality reduction that can be used to extract meaningful features from raw data.

Applications of Unsupervised Learning

Here are some common applications of unsupervised learning:

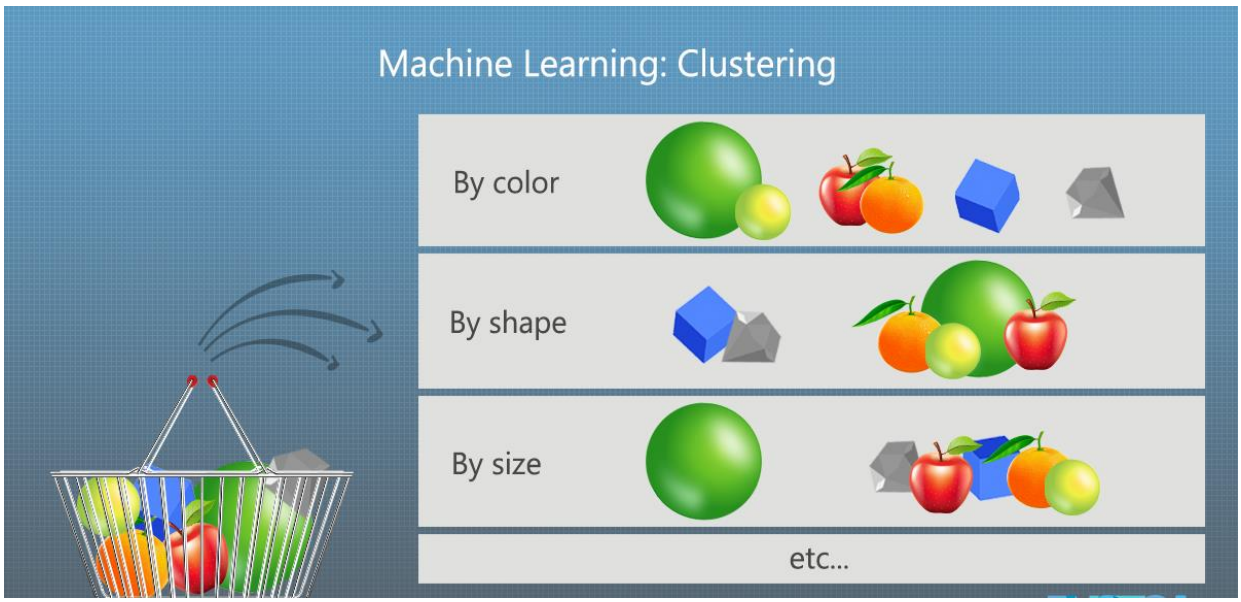
- Clustering: Group similar data points into clusters.
- Anomaly detection: Identify outliers or anomalies in data.
- Dimensionality reduction: Reduce the dimensionality of data while preserving its essential information.
- Recommendation systems: Suggest products, movies, or content to users based on their historical behaviour or preferences.
- Topic modelling: Discover latent topics within a collection of documents.
- Density estimation: Estimate the probability density function of data.
- Image and video compression: Reduce the amount of storage required for multimedia content.
- Data pre-processing: Help with data pre-processing tasks such as data cleaning, imputation of missing values, and data scaling.
- Market basket analysis: Discover associations between products.
- Genomic data analysis: Identify patterns or group genes with similar expression profiles.
- Image segmentation: Segment images into meaningful regions.
- Community detection in social networks: Identify communities or groups of individuals with similar interests or connections.
- Customer behaviour analysis: Uncover patterns and insights for better marketing and product recommendations.
- Content recommendation: Classify and tag content to make it easier to recommend similar items to users.
- Exploratory data analysis (EDA): Explore data and gain insights before defining specific tasks.

There are 4 main categories of unsupervised learning:

1. Clustering
2. Dimensionality Reduction
3. Anomaly Detection
4. Association Rule Learning

Industries Helping Hands

1. Clustering: It is basically a collection of objects on the basis of similarity and dissimilarity between them. Let's understand by example.

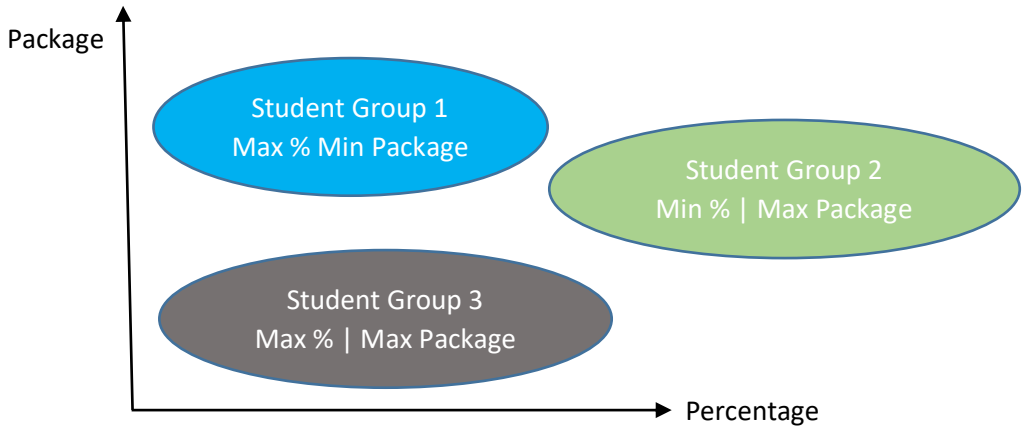


Example: If we have 10,000 Students Data who have placed or some un placed and we want to find the new predictive result output.

Output

S.No	Student_Name	St_Percentage	St_CGPA	St_Placement	Package
1	Nishant	87%	7.8	Yes	8 LPA
2	Swati	75%	7.2	No	0 LPA
3	Ragini	91%	8.4	Yes	10 LPA
4	Salman	68%	6.5	No	0 LPA
.					
.					
10K	XYZ	00%	0.0	No	0

Input Data



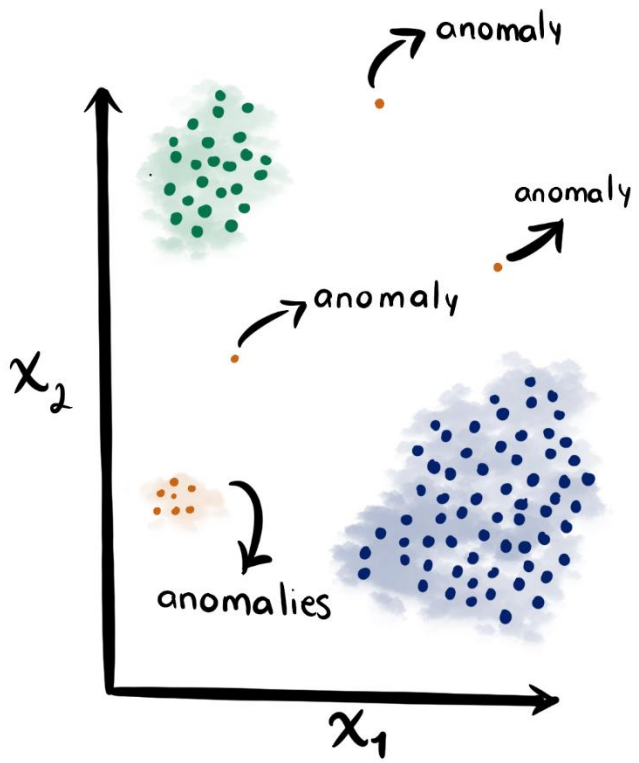
2. Dimensionality Reduction: Dimensionality reduction is the process of reducing the number of features (or dimensions) in a dataset while retaining as much information as possible.

   Example: Many input columns is not required in the algorithm, it's remove extra columns.

Type of Rooms	No Of Rooms	Room Size
Bedroom	3	10*15 | 10*10 | 10*12
Washroom	2	6*6 | 6*8
Kitchen	1	10*8

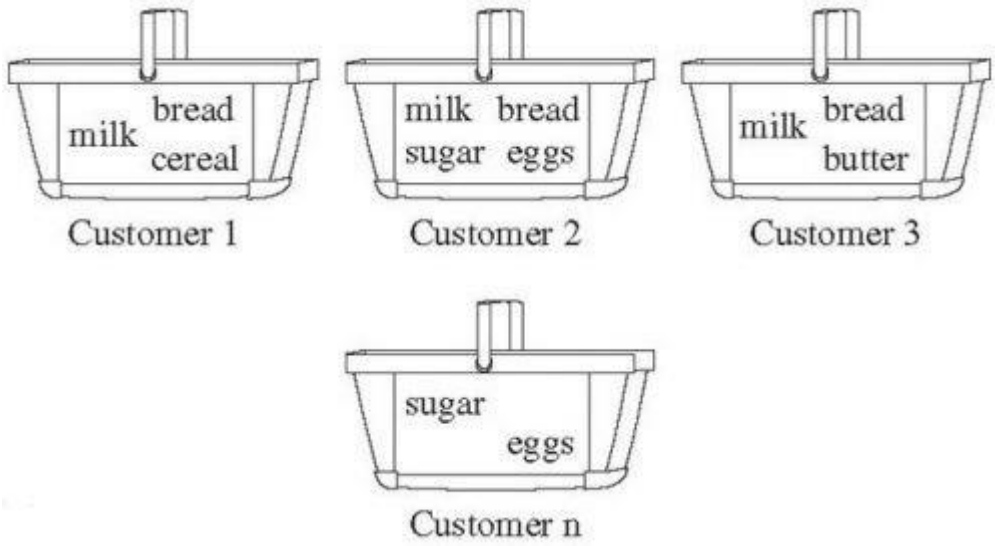
Area
370 Sqft
84 Sqft
80 Sqft

3. Anomaly Detection: Anomaly detection in machine learning is the process of identifying such anomalies in a dataset. Anomaly detection is an important technique in machine learning and data mining, as it can be used to detect unusual behaviour, identify errors, and discover new insights in large datasets.



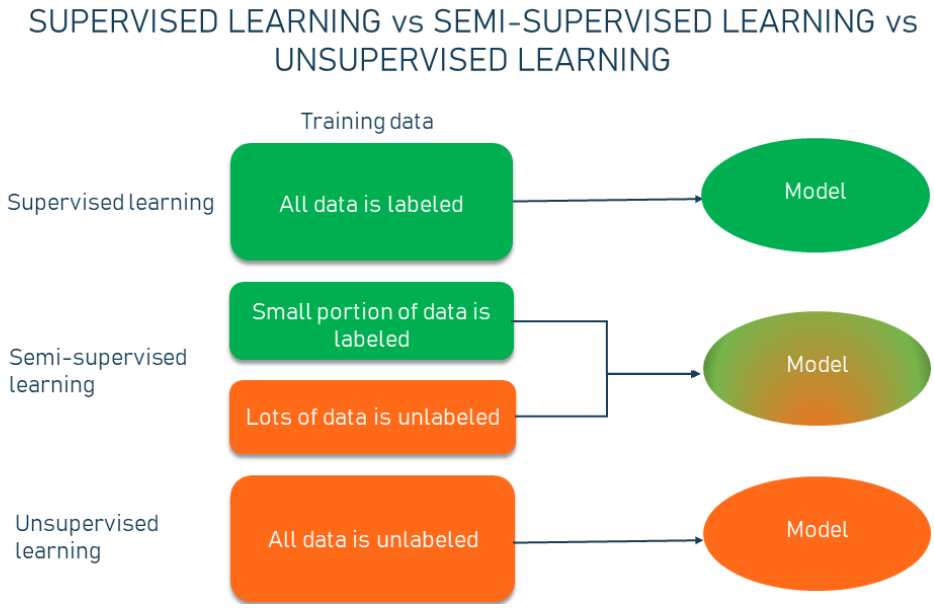
4. Association Rule Learning
   Association rule mining is a technique used to uncover hidden relationships between variables in large datasets.

   Example: Study many purchase product in shopping mart.



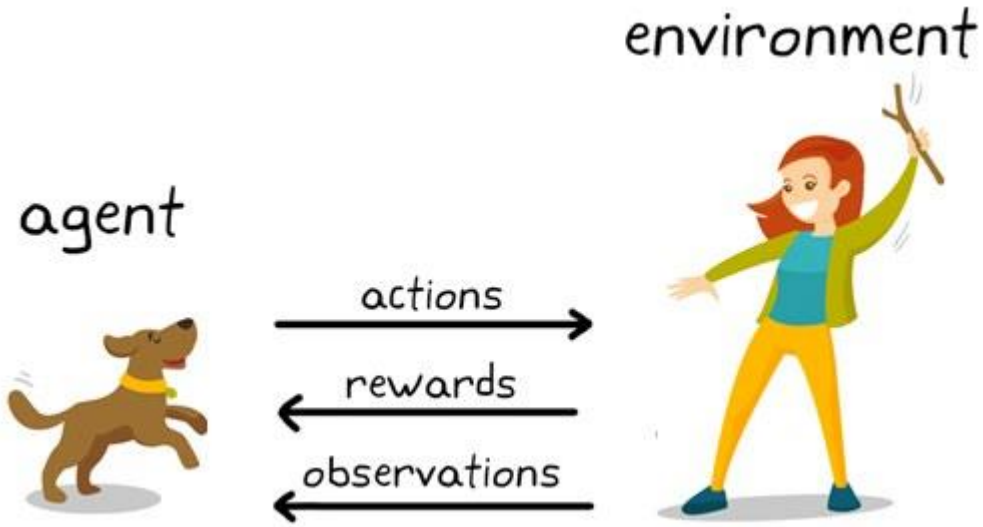
3. **Semi supervised machine learning:** (Partially Supervised and Un Supervised)
   We want to ignore labelling, because it's very costly that's why we use "Semi supervised machine learning"

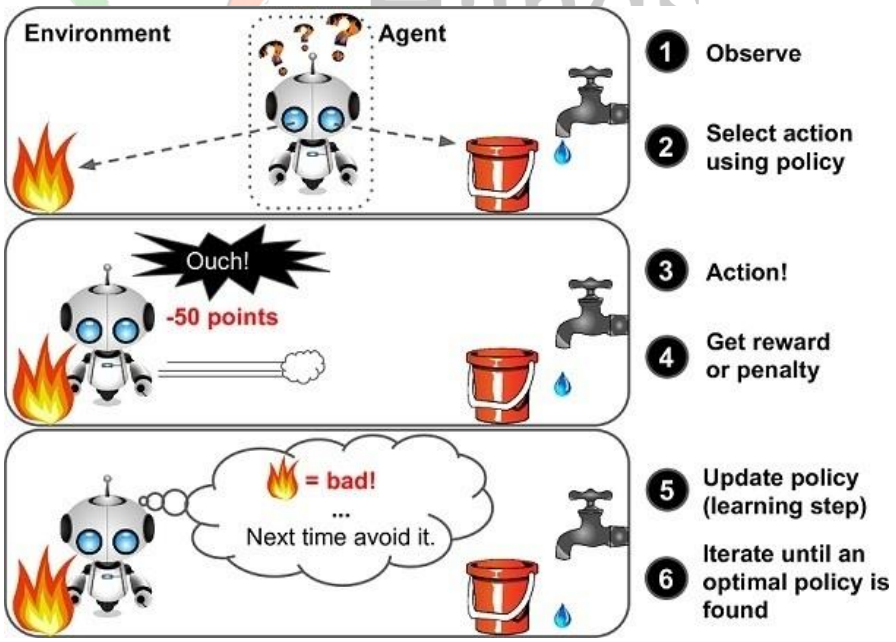
   Example: Google photos



4. Reinforcement machine learning: Reinforcement learning is a machine learning training method based on rewarding desired behaviours and punishing undesired ones. In general, a reinforcement learning agent -- the entity being trained -- is able to perceive and interpret its environment, take actions and learn through trial and error.

Example: Self Drive Cars | Go Game Deep Mind Company Google





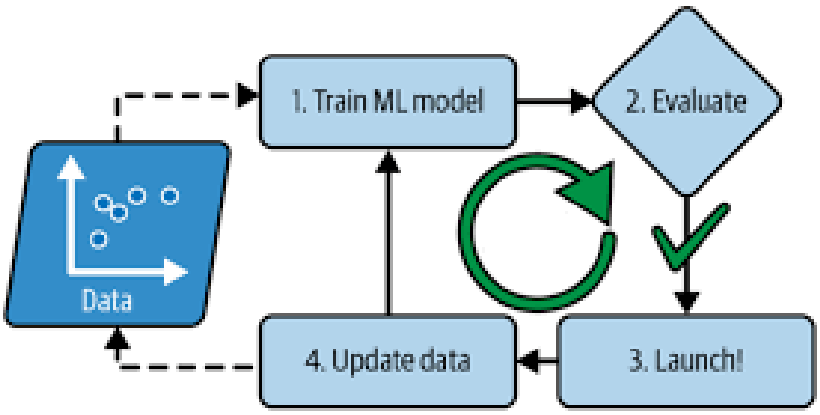
Data Science | 30 Days of Machine Learning | Day - 2

Educator Name: Nishant Dhote
Support Team: **+91-7880-113-112**

What is Batch / Offline Machine Learning?

What is Online Machine Learning?

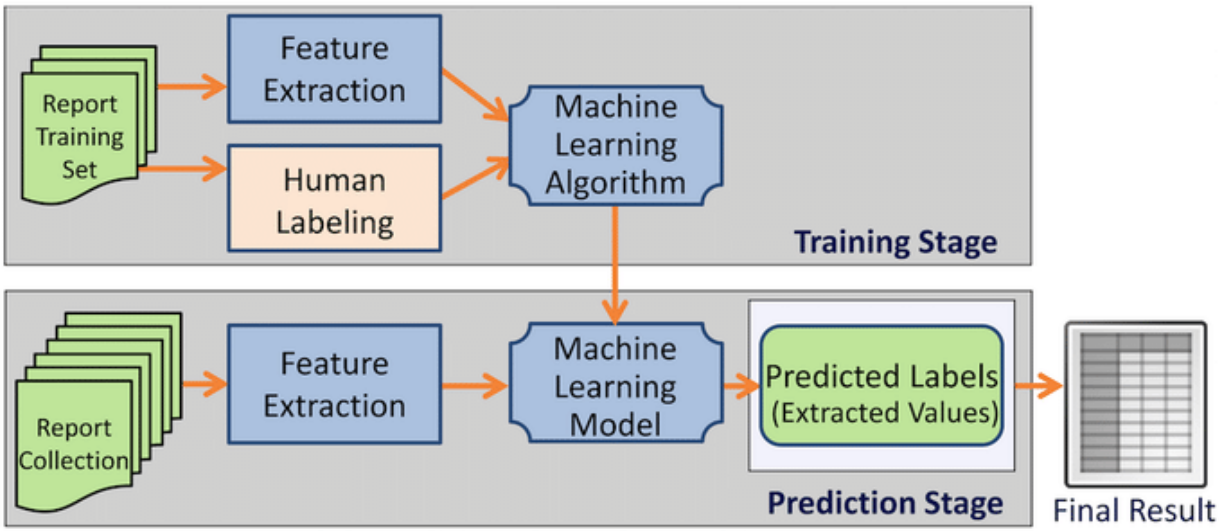
Difference Between Online Vs Offline Machine Learning?



Batch Machine Learning (Offline ML):

In batch learning, data is accumulated over a period of time. The machine learning model is then trained with this accumulated data from time to time in batches. It is the direct opposite of online learning because the model is unable to learn incrementally from a stream of live data. In batch learning, the machine learning algorithm updates its parameters only after consuming batches of new data.

The fact that models are trained with large batches of accumulated data means that more time and resources such as CPU, memory space, and disk input/output are needed.



(a) Traditional Batch Machine Learning

Batch size (machine learning)

Batch size is a term used in machine learning and refers to the number of training examples utilized in one iteration. The batch size can be one of three options:

Batch mode: where the batch size is equal to the total dataset thus making the iteration and epoch values equivalent

Mini-batch mode: where the batch size is greater than one but less than the total dataset size. Usually, a number that can be divided into the total dataset size.

Stochastic mode: where the batch size is equal to one. Therefore, the gradient and the neural network parameters are updated after each sample.

Advantages: (When we use?)

- Lower production costs
- Flexibility
- Better quality control
- Increased efficiency

Disadvantages:

- Large-scale failures
- They are monolithic in nature
- Complex to manage
- Require very expensive

Online Machine Learning:

Online machine learning is a method of machine learning where the model incrementally learns from a stream of data points in real-time. It's a dynamic process that adapts its predictive algorithm over time, allowing the model to change as new data arrives. This method is incredibly significant in today's rapidly evolving data-rich environments because it can provide timely and accurate predictions.

Real time example: Chat Bot | YouTube or OTT Recommended | Stock | Fraud Detection

Library use in Python: River Library (https://riverml.xyz/dev/) | Vowpal Wabbit (https://vowpalwabbit.org/)

Advantages:(What are the Benefits of Online Machine Learning?)

Adaptability. Just like the cyclist learning as they go, online machine learning can adapt to new patterns in the data, improving its performance over time.

Scalability. Since online learning processes data one at a time, it doesn't require the storage capacity that batch learning does. This makes it scalable to big data applications.

Real-time predictions. Unlike batch learning that might be outdated by the time it's implemented, online learning provides real-time insights, which can be critical in many applications like stock trading and health monitoring.

Efficiency. As online machine learning allows for continuous learning and updating of models, this can lead to faster and more cost-efficient decision-making processes.

Disadvantages:(What are the Limitations of Online Machine Learning?)

Sensitive to sequence or Tricky to Use. The order in which the data is presented can impact the learning process. An unusual data point can significantly alter the model's parameters, leading to decreased accuracy.

Less control over training or Risky. Unlike batch learning, where you can control the training process, online learning is always on. An unexpected influx of bad quality data can lead to poor predictions.

Lack of interpretability. Online learning algorithms, especially those based on deep learning or neural networks, can be highly complex and difficult to interpret. This lack of interpretability can make it challenging to understand and explain the model's decisions.

If we facing problem in online learning system so real time technique, we use: We use anomaly detection system so use switch the system in Offline Mode or rollback the system online to offline batch learning.

Difference Between Online Vs Offline Machine Learning?

Offline Learning	Features	Online Learning
Less complex as model is constant	**Complexity**	Dynamic complexity as the model keeps evolving over time
Fewer computations, single time batch-based training	**Computational Power**	Continuous data ingestions result in consequent model refinement computations
Easier to implement	**Use in Production**	Difficult to implement and manage
Image Classification or anything related to Machine Learning - where data patterns remains constant without sudden concept drifts	**Applications**	Used in finance, economics, heath where new data patterns are constantly emerging
Industry proven tools. E.g. Sci-kit, TensorFlow, Pytorch, Keras, Spark Mlib	**Tools**	Active research/New project tools: E.g. MOA, SAMOA, scikit-multiflow, streamDM

Helping Hands

Instance Based Machine Learning

Model Based Machine Learning

Instance Based Vs Model Based Machine Learning

Think about how human learn, answer is memorised or generalized

Learning → Memorising (Example of Instanced Based ML)

Learning → Generalized (Example of Model Based ML)

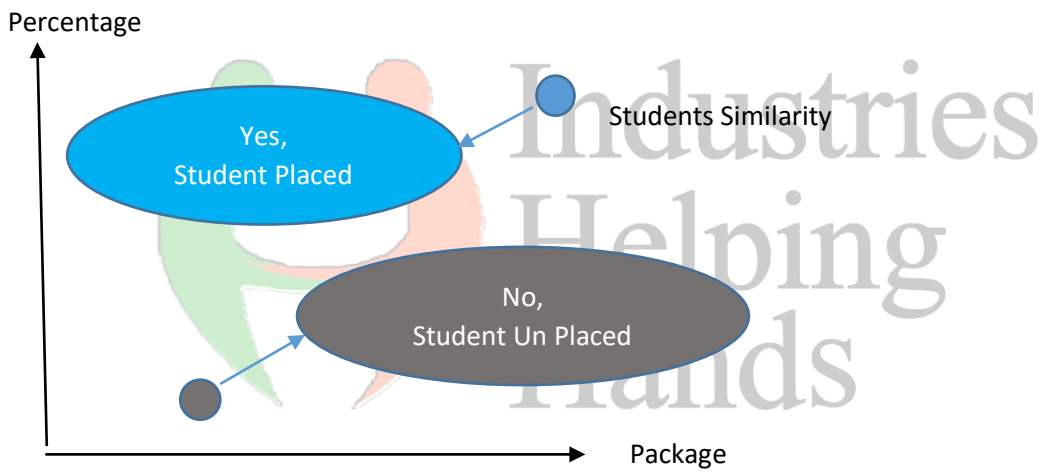
 # Instance Based Machine Learning:

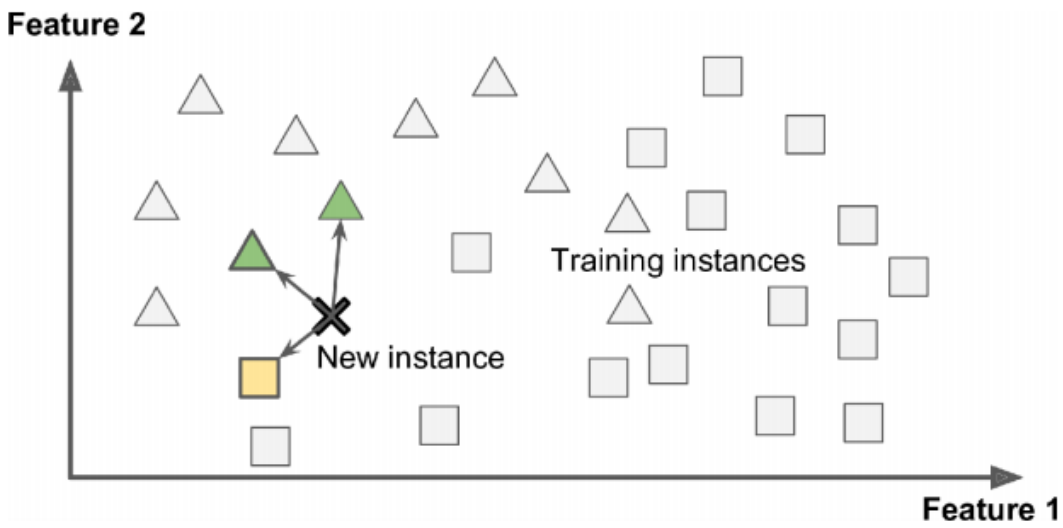
The Machine Learning systems which are categorized as instance-based learning are the systems that learn the training examples by heart and then generalizes to new instances based on some similarity measure. It is called instance-based because it builds the hypotheses from the training instances. It is also known as memory-based learning or lazy-learning (because they delay processing until a new instance must be classified).

The time complexity of this algorithm depends upon the size of training data. Each time whenever a new query is encountered, its previously stores data is examined. And assign to a target function value for the new instance.

Example: Sample Dataset

S.No	Student_Name	St_Percentage	St_CGPA	St_Placement	Package
1	Nishant	87%	7.8	Yes	8 LPA
2	Swati	75%	7.2	No	0 LPA
3	Ragini	91%	8.4	Yes	10 LPA
4	Salman	68%	6.5	No	0 LPA
.					
.					
10K	XYZ	00%	0.0	No	0

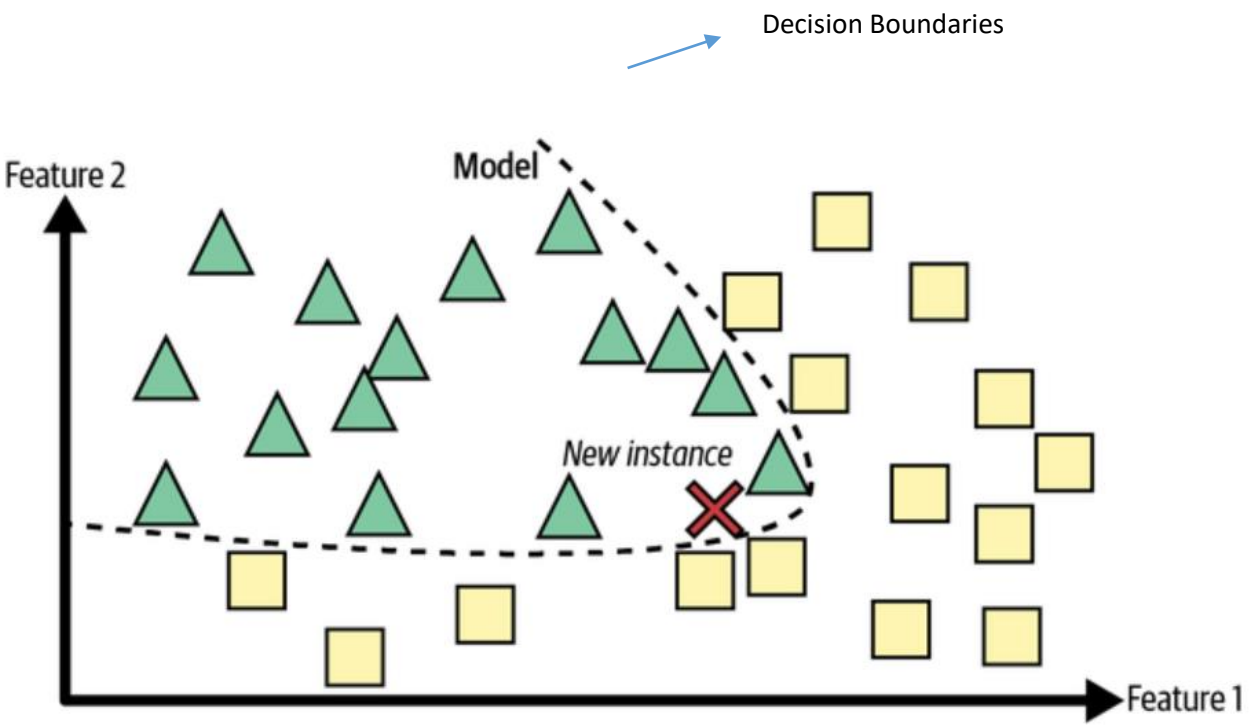




The new instance would be classified as a triangle because the majority (2 Triangles vs 1 Square) of the most similar instances belong to that class

Model Based Machine Learning:

Model-based learning (also known as structure-based or eager learning) takes a different approach by constructing models from the training data that can generalize better than instance-based methods. This involves using algorithms like linear regression, logistic regression, random forest, etc. trees to create an underlying model from which predictions can be made for new data points. The picture below represents how the prediction about the class is decided based on boundary learned from training data rather than comparing with learned data set based on similarity measures.



The model based learning approach has several benefits over instance-based methods, such as faster processing speeds and better generalization capabilities due to its use of an underlying model rather than relying solely on memorized examples. However, this approach requires more time and effort to develop and tune the model for optimal performance on unseen data sets.

Instance Based Vs Model Based Machine Learning

Model-Based Machine Learning	Instance-Based Machine Learning
Prepare the data for model training	Prepare the data for model training. No difference here
Train model from training data to estimate model parameters i.e. discover patterns	Do not train model. Pattern discovery postponed until scoring query received
Store the model in suitable form	There is no model to store
Generalize the rules in form of model, even before scoring instance is seen	No generalization before scoring. Only generalize for each scoring instance individually as and when seen
Predict for unseen scoring instance using model	Predict for unseen scoring instance using training data directly
Can throw away input/training data after model training	Input/training data must be kept since each query uses part or full set of training observations
Requires a known model form	May not have explicit model form
Storing models generally requires less storage	Storing training data generally requires more storage

#Challenges in Machine Learning:

#Application of Machine Learning:

#Machine Learning Development Life Cycle:

Challenges in Machine Learning:

- Data Collection
- Insufficient Data
- Non Reprehensive Data (Sampling Noise, Sampling Bias)
- Poor Quality Data
- Irrelevant Features
- Overfitting & Under fitting
- Software Integration
- Deployment & Cost Involved

 # Application of Machine Learning:

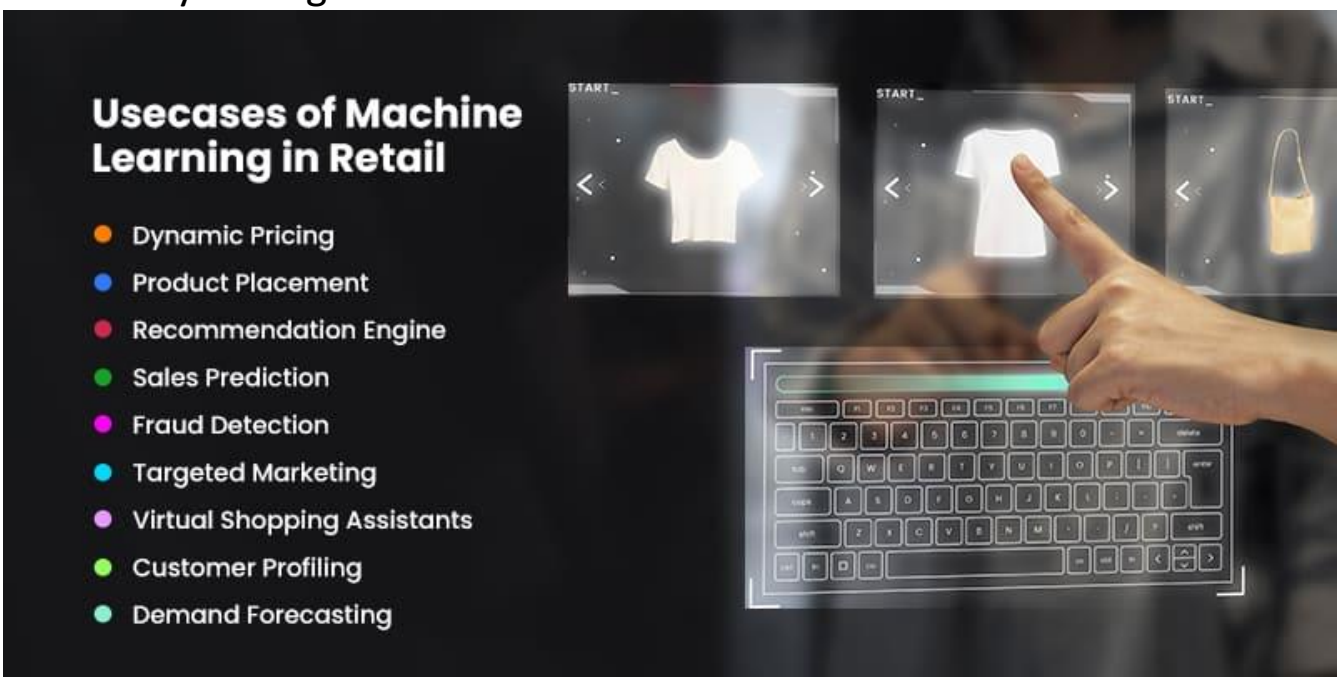
1. Retail / E-Commerce Sector
2. Banking and Finance Sector
3. Logistics and Transportation Sector
4. Manufacturing Sector
5. Social Networking Sector
6. Many More Sector

 # Machine Learning Development Life Cycle (MLDLC/MLDC):

1. Frame the problem
2. Gathering data
3. Data pre processing
4. Exploratory data analysis
5. Feature engineering & selection
6. Model training, evaluation and selection
7. Model deployment
8. Testing
9. Optimize

Industries Helping Hands

Data Story Telling: Curious Data Minds



Case Study Link: https://www.matellio.com/blog/machine-learning-use-cases-in-retail/

Industries Helping Hands

Welcome to 30 Days ML

How to upload / Import Data

In [6]:
```
# pandas
import pandas as pd
```

In [7]:
```
pd.read_csv('hotel_bookings.csv')
```

Out[7]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_nun
0	Resort Hotel	0	342	2015	July	
1	Resort Hotel	0	737	2015	July	
2	Resort Hotel	0	7	2015	July	
3	Resort Hotel	0	13	2015	July	
4	Resort Hotel	0	14	2015	July	
...	...	...	...	...	...	
119385	City Hotel	0	23	2017	August	
119386	City Hotel	0	102	2017	August	
119387	City Hotel	0	34	2017	August	
119388	City Hotel	0	109	2017	August	
119389	City Hotel	0	205	2017	August	

119390 rows × 32 columns

Gathering Data With CSV

In [8]:
```
import pandas as pd
```

In [9]:
```
df = pd.read_csv('aug_train.csv')
```

In [10]:
```
df
```

Out[10]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_univers
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollme
1	29725	city_40	0.776	Male	No relevent experience	no_enrollme
2	11561	city_21	0.624	NaN	No relevent experience	Full time cou
3	33241	city_115	0.789	NaN	No relevent experience	N
4	666	city_162	0.767	Male	Has relevent experience	no_enrollme
...	...	...	...	...	...	
19153	7386	city_173	0.878	Male	No relevent experience	no_enrollme
19154	31398	city_103	0.920	Male	Has relevent experience	no_enrollme
19155	24576	city_103	0.920	Male	Has relevent experience	no_enrollme
19156	5756	city_65	0.802	Male	Has relevent experience	no_enrollme
19157	23834	city_67	0.855	NaN	No relevent experience	no_enrollme

19158 rows × 14 columns

3. Opening a csv file from an URL

In [11]:

```
import requests
from io import StringIO
url = "https://raw.githubusercontent.com/cs109/2014_data/master/countries.csv"
headers = {"User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.0) Ge
req = requests.get(url, headers=headers)
data = StringIO(req.text)
pd.read_csv(data)
```

Out[11]:

	Country	Region
0	Algeria	AFRICA
1	Angola	AFRICA
2	Benin	AFRICA
3	Botswana	AFRICA
4	Burkina	AFRICA
...	...	...
189	Paraguay	SOUTH AMERICA
190	Peru	SOUTH AMERICA
191	Suriname	SOUTH AMERICA
192	Uruguay	SOUTH AMERICA
193	Venezuela	SOUTH AMERICA

194 rows × 2 columns

4. Sep Parameter

In [12]:
```
pd.read_csv('movie_titles_metadata.tsv')
```

Out[12]:

	m0\t10 things i hate about you\t1999\t6.90\t62847\t['comedy' 'romance']
0	m1\t1492: conquest of paradise\t1992\t6.20\t10...
1	m2\t15 minutes\t2001\t6.10\t25854\t['action' '...
2	m3\t2001: a space odyssey\t1968\t8.40\t163227\...
3	m4\t48 hrs.\t1982\t6.90\t22289\t['action' 'com...
4	m5\tthe fifth element\t1997\t7.50\t133756\t['a...
...	...
611	m612\twatchmen\t2009\t7.80\t135229\t['action' ...
612	m613\txxx\t2002\t5.60\t53505\t['action' 'adven...
613	m614\tx-men\t2000\t7.40\t122149\t['action' 'sc...
614	m615\tyoung frankenstein\t1974\t8.00\t57618\t[...
615	m616\tzulu dawn\t1979\t6.40\t1911\t['action' '...

616 rows × 1 columns

In [13]:
```
pd.read_csv('movie_titles_metadata.tsv',sep='\t')
```

Out[13]:

	m0	10 things i hate about you	1999	6.90	62847	['comedy' 'romance']
0	m1	1492: conquest of paradise	1992	6.2	10421.0	['adventure' 'biography' 'drama' 'history']
1	m2	15 minutes	2001	6.1	25854.0	['action' 'crime' 'drama' 'thriller']
2	m3	2001: a space odyssey	1968	8.4	163227.0	['adventure' 'mystery' 'sci-fi']
3	m4	48 hrs.	1982	6.9	22289.0	['action' 'comedy' 'crime' 'drama' 'thriller']
4	m5	the fifth element	1997	7.5	133756.0	['action' 'adventure' 'romance' 'sci-fi' 'thri...
...	...	...	...	...	...	...
611	m612	watchmen	2009	7.8	135229.0	['action' 'crime' 'fantasy' 'mystery' 'sci-fi'...
612	m613	xxx	2002	5.6	53505.0	['action' 'adventure' 'crime']
613	m614	x-men	2000	7.4	122149.0	['action' 'sci-fi']
614	m615	young frankenstein	1974	8.0	57618.0	['comedy' 'sci-fi']
615	m616	zulu dawn	1979	6.4	1911.0	['action' 'adventure' 'drama' 'history' 'war']

616 rows × 6 columns

In [14]:

```
pd.read_csv('movie_titles_metadata.tsv',sep='\t',names=['sno','name','release_year'
```

Out[14]:

	sno	name	release_year	rating	votes	genres
0	m0	10 things i hate about you	1999	6.9	62847.0	['comedy' 'romance']
1	m1	1492: conquest of paradise	1992	6.2	10421.0	['adventure' 'biography' 'drama' 'history']
2	m2	15 minutes	2001	6.1	25854.0	['action' 'crime' 'drama' 'thriller']
3	m3	2001: a space odyssey	1968	8.4	163227.0	['adventure' 'mystery' 'sci-fi']
4	m4	48 hrs.	1982	6.9	22289.0	['action' 'comedy' 'crime' 'drama' 'thriller']
...	...	...	...	...	...	...
612	m612	watchmen	2009	7.8	135229.0	['action' 'crime' 'fantasy' 'mystery' 'sci-fi'...
613	m613	xxx	2002	5.6	53505.0	['action' 'adventure' 'crime']
614	m614	x-men	2000	7.4	122149.0	['action' 'sci-fi']
615	m615	young frankenstein	1974	8.0	57618.0	['comedy' 'sci-fi']
616	m616	zulu dawn	1979	6.4	1911.0	['action' 'adventure' 'drama' 'history' 'war']

617 rows × 6 columns

5. Index_col parameter

In [15]:
```
pd.read_csv('aug_train.csv')
```

Out[15]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_univers
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollme
1	29725	city_40	0.776	Male	No relevent experience	no_enrollme
2	11561	city_21	0.624	NaN	No relevent experience	Full time cou
3	33241	city_115	0.789	NaN	No relevent experience	N
4	666	city_162	0.767	Male	Has relevent experience	no_enrollme
...	...	...	...	...	...	
19153	7386	city_173	0.878	Male	No relevent experience	no_enrollme
19154	31398	city_103	0.920	Male	Has relevent experience	no_enrollme
19155	24576	city_103	0.920	Male	Has relevent experience	no_enrollme
19156	5756	city_65	0.802	Male	Has relevent experience	no_enrollme
19157	23834	city_67	0.855	NaN	No relevent experience	no_enrollme

19158 rows × 14 columns

6. Header parameter

In [18]:
```
pd.read_csv('test.csv')
```

Out[18]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	U
0	0	enrollee_id	city	city_development_index	gender	relevent_experience	enrolle
1	1	29725	city_40	0.776	Male	No relevent experience	no_
2	2	11561	city_21	0.624	NaN	No relevent experience	Full
3	3	33241	city_115	0.789	NaN	No relevent experience	
4	4	666	city_162	0.767	Male	Has relevent experience	no_

In [19]:
```
pd.read_csv('test.csv',header=1)
```

Out[19]:

	0	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_universit
0	1	29725	city_40	0.776	Male	No relevent experience	no_enrollmen
1	2	11561	city_21	0.624	NaN	No relevent experience	Full time cours
2	3	33241	city_115	0.789	NaN	No relevent experience	Nal
3	4	666	city_162	0.767	Male	Has relevent experience	no_enrollmen

7. Use_col parameter

In [20]:
```
pd.read_csv('aug_train.csv',usecols=['enrollee_id','gender','education_level'])
```

Out[20]:

	enrollee_id	gender	education_level
0	8949	Male	Graduate
1	29725	Male	Graduate
2	11561	NaN	Graduate
3	33241	NaN	Graduate
4	666	Male	Masters
...	...	...	...
19153	7386	Male	Graduate
19154	31398	Male	Graduate
19155	24576	Male	Graduate
19156	5756	Male	High School
19157	23834	NaN	Primary School

19158 rows × 3 columns

8. Squeeze parameters

In [27]:
```
pd.read_csv('aug_train.csv',usecols=['gender'],squeeze=True)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[27], line 1
----> 1 pd.read_csv('aug_train.csv',usecols=['enrolle_id'],squeeze=True)

TypeError: read_csv() got an unexpected keyword argument 'squeeze'
```

9. Skiprows/nrows Parameter

In [28]:
```
pd.read_csv('aug_train.csv')
```

Out[28]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_univers
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollme
1	29725	city_40	0.776	Male	No relevent experience	no_enrollme
2	11561	city_21	0.624	NaN	No relevent experience	Full time cou
3	33241	city_115	0.789	NaN	No relevent experience	N
4	666	city_162	0.767	Male	Has relevent experience	no_enrollme
...	...	...	...	...	...	
19153	7386	city_173	0.878	Male	No relevent experience	no_enrollme
19154	31398	city_103	0.920	Male	Has relevent experience	no_enrollme
19155	24576	city_103	0.920	Male	Has relevent experience	no_enrollme
19156	5756	city_65	0.802	Male	Has relevent experience	no_enrollme
19157	23834	city_67	0.855	NaN	No relevent experience	no_enrollme

19158 rows × 14 columns

In [29]:

```
pd.read_csv('aug_train.csv',skiprows=[0,1])
```

Out[29]:

	29725	city_40	0.7759999999999999	Male	No relevent experience	no_enrollment	Graduate	ST
0	11561	city_21	0.624	NaN	No relevent experience	Full time course	Graduate	ST
1	33241	city_115	0.789	NaN	No relevent experience	NaN	Graduate	Busir Deg
2	666	city_162	0.767	Male	Has relevent experience	no_enrollment	Masters	ST
3	21651	city_176	0.764	NaN	Has relevent experience	Part time course	Graduate	ST
4	28806	city_160	0.920	Male	Has relevent experience	no_enrollment	High School	N
...	...	...	...	...	...	...	...	
19151	7386	city_173	0.878	Male	No relevent experience	no_enrollment	Graduate	Humani
19152	31398	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	ST
19153	24576	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	ST
19154	5756	city_65	0.802	Male	Has relevent experience	no_enrollment	High School	N
19155	23834	city_67	0.855	NaN	No relevent experience	no_enrollment	Primary School	N

19156 rows × 14 columns

In [30]:
```
pd.read_csv('aug_train.csv',skiprows=[5,6])
```

Out[30]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_univers
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollme
1	29725	city_40	0.776	Male	No relevent experience	no_enrollme
2	11561	city_21	0.624	NaN	No relevent experience	Full time cou
3	33241	city_115	0.789	NaN	No relevent experience	N
4	28806	city_160	0.920	Male	Has relevent experience	no_enrollme
...	...	...	...	...	...	
19151	7386	city_173	0.878	Male	No relevent experience	no_enrollme
19152	31398	city_103	0.920	Male	Has relevent experience	no_enrollme
19153	24576	city_103	0.920	Male	Has relevent experience	no_enrollme
19154	5756	city_65	0.802	Male	Has relevent experience	no_enrollme
19155	23834	city_67	0.855	NaN	No relevent experience	no_enrollme

19156 rows × 14 columns

In [31]:

```
pd.read_csv('aug_train.csv',nrows=100)
```

Out[31]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment
2	11561	city_21	0.624	NaN	No relevent experience	Full time course
3	33241	city_115	0.789	NaN	No relevent experience	NaN
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment
...	...	...	...	...	...	...
95	12081	city_65	0.802	Male	Has relevent experience	Full time course
96	7364	city_160	0.920	NaN	No relevent experience	Full time course
97	11184	city_74	0.579	NaN	No relevent experience	Full time course
98	7016	city_65	0.802	Male	Has relevent experience	no_enrollment
99	8695	city_11	0.550	Male	Has relevent experience	no_enrollment

100 rows × 14 columns

10. Encoding parameter

In [32]:

```
pd.read_csv('zomato.csv')
```

```
---------------------------------------------------------------------------
UnicodeDecodeError                        Traceback (most recent call last)
Cell In[32], line 1
----> 1 pd.read_csv('zomato.csv')

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:912, in read_csv(f
ilepath_or_buffer, sep, delimiter, header, names, index_col, usecols, dtype, engin
e, converters, true_values, false_values, skipinitialspace, skiprows, skipfooter,
nrows, na_values, keep_default_na, na_filter, verbose, skip_blank_lines, parse_dat
es, infer_datetime_format, keep_date_col, date_parser, date_format, dayfirst, cach
e_dates, iterator, chunksize, compression, thousands, decimal, lineterminator, quo
techar, quoting, doublequote, escapechar, comment, encoding, encoding_errors, dial
ect, on_bad_lines, delim_whitespace, low_memory, memory_map, float_precision, stor
age_options, dtype_backend)
    899 kwds_defaults = _refine_defaults_read(
    900     dialect,
    901     delimiter,
   (...)
    908     dtype_backend=dtype_backend,
    909 )
    910 kwds.update(kwds_defaults)
--> 912 return _read(filepath_or_buffer, kwds)

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:577, in _read(file
path_or_buffer, kwds)
    574 _validate_names(kwds.get("names", None))
    576 # Create the parser.
--> 577 parser = TextFileReader(filepath_or_buffer, **kwds)
    579 if chunksize or iterator:
    580     return parser

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:1407, in TextFileR
eader.__init__(self, f, engine, **kwds)
   1404     self.options["has_index_names"] = kwds["has_index_names"]
   1406 self.handles: IOHandles | None = None
-> 1407 self._engine = self._make_engine(f, self.engine)

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:1679, in TextFileR
eader._make_engine(self, f, engine)
   1676     raise ValueError(msg)
   1678 try:
-> 1679     return mapping[engine](f, **self.options)
   1680 except Exception:
   1681     if self.handles is not None:

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\c_parser_wrapper.py:93, in CP
arserWrapper.__init__(self, src, **kwds)
     90 if kwds["dtype_backend"] == "pyarrow":
     91     # Fail here loudly instead of in cython after reading
     92     import_optional_dependency("pyarrow")
---> 93 self._reader = parsers.TextReader(src, **kwds)
     95 self.unnamed_cols = self._reader.unnamed_cols
     97 # error: Cannot determine type of 'names'

File ~\anaconda3\Lib\site-packages\pandas\_libs\parsers.pyx:550, in pandas._libs.p
arsers.TextReader.__cinit__()

File ~\anaconda3\Lib\site-packages\pandas\_libs\parsers.pyx:639, in pandas._libs.p
arsers.TextReader._get_header()

File ~\anaconda3\Lib\site-packages\pandas\_libs\parsers.pyx:850, in pandas._libs.p
arsers.TextReader._tokenize_rows()

File ~\anaconda3\Lib\site-packages\pandas\_libs\parsers.pyx:861, in pandas._libs.p
```

```
arsers.TextReader._check_tokenize_status()

File ~\anaconda3\Lib\site-packages\pandas\_libs\parsers.pyx:2021, in pandas._libs.
parsers.raise_parser_error()

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xed in position 7044: invalid
continuation byte
```

In [33]:

```
pd.read_csv('zomato.csv',encoding='latin-1')
```

Out[33]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	L
0	6317637	Le Petit Souffle	162	Makati City	Third Floor, Century City Mall, Kalayaan Avenu...	Century City Mall, Poblacion, Makati City	Century City Mall, Poblacion, Makati City, Mak...	12
1	6304287	Izakaya Kikufuji	162	Makati City	Little Tokyo, 2277 Chino Roces Avenue, Legaspi...	Little Tokyo, Legaspi Village, Makati City	Little Tokyo, Legaspi Village, Makati City, Ma...	12
2	6300002	Heat - Edsa Shangri-La	162	Mandaluyong City	Edsa Shangri-La, 1 Garden Way, Ortigas, Mandal...	Edsa Shangri-La, Ortigas, Mandaluyong City	Edsa Shangri-La, Ortigas, Mandaluyong City, Ma...	12
3	6318506	Ooma	162	Mandaluyong City	Third Floor, Mega Fashion Hall, SM Megamall, O...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	12
4	6314302	Sambo Kojin	162	Mandaluyong City	Third Floor, Mega Atrium, SM Megamall, Ortigas...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	12
...	...	...	...	...	...	...	...	
9546	5915730	NamlÛ± Gurme	208	ÛÁstanbul	Kemanke□ô Karamustafa Pa□ôa Mahallesi, RÛ±htÛ±...	Karakí_y	Karakí_y, ÛÁstanbul	2
9547	5908749	Ceviz AÛôacÛ±	208	ÛÁstanbul	Ko□ôuyolu Mahallesi, Muhittin îstí_ndaÛô Cadd...	Ko□ôuyolu	Ko□ôuyolu, ÛÁstanbul	2
9548	5915807	Huqqa	208	ÛÁstanbul	Kuruí_e□ôme Mahallesi, Muallim Naci Caddesi, N...	Kuruí_e□ôme	Kuruí_e□ôme, ÛÁstanbul	2
9549	5916112	A□ô□ôk Kahve	208	ÛÁstanbul	Kuruí_e□ôme Mahallesi, Muallim Naci Caddesi, N...	Kuruí_e□ôme	Kuruí_e□ôme, ÛÁstanbul	2
9550	5927402	Walter's Coffee Roastery	208	ÛÁstanbul	CafeaÛôa Mahallesi, BademaltÛ± Sokak, No 21/B,...	Moda	Moda, ÛÁstanbul	2

9551 rows × 21 columns

11. Skip bad lines

In [34]:
```
pd.read_csv('zomato.csv', address=';', encoding="latin-1")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[34], line 1
----> 1 pd.read_csv('zomato.csv', address=';', encoding="latin-1")

TypeError: read_csv() got an unexpected keyword argument 'address'
```

In [35]:
```
pd.read_csv('zomato.csv', address=';', encoding="latin-1",error_bad_lines=False)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[35], line 1
----> 1 pd.read_csv('zomato.csv', address=';', encoding="latin-1",error_bad_lines=
False)

TypeError: read_csv() got an unexpected keyword argument 'address'
```

12.dtypes parameter

In [37]:
```
pd.read_csv('aug_train.csv').info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19158 entries, 0 to 19157
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   enrollee_id             19158 non-null  int64
 1   city                    19158 non-null  object
 2   city_development_index  19158 non-null  float64
 3   gender                  14650 non-null  object
 4   relevent_experience     19158 non-null  object
 5   enrolled_university     18772 non-null  object
 6   education_level         18698 non-null  object
 7   major_discipline        16345 non-null  object
 8   experience              19093 non-null  object
 9   company_size            13220 non-null  object
 10  company_type            13018 non-null  object
 11  last_new_job            18735 non-null  object
 12  training_hours          19158 non-null  int64
 13  target                  19158 non-null  float64
dtypes: float64(2), int64(2), object(10)
memory usage: 2.0+ MB
```

In [38]:
```
pd.read_csv('aug_train.csv',dtype={'target':int}).info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19158 entries, 0 to 19157
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   enrollee_id             19158 non-null  int64
 1   city                    19158 non-null  object
 2   city_development_index  19158 non-null  float64
 3   gender                  14650 non-null  object
 4   relevent_experience     19158 non-null  object
 5   enrolled_university     18772 non-null  object
 6   education_level         18698 non-null  object
 7   major_discipline        16345 non-null  object
 8   experience              19093 non-null  object
 9   company_size            13220 non-null  object
 10  company_type            13018 non-null  object
 11  last_new_job            18735 non-null  object
 12  training_hours          19158 non-null  int64
 13  target                  19158 non-null  int32
dtypes: float64(1), int32(1), int64(2), object(10)
memory usage: 2.0+ MB
```

13. Handling Dates

In [39]:
```
pd.read_csv('IPL Matches 2008-2020.csv').info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 816 entries, 0 to 815
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   id               816 non-null    int64
 1   city             803 non-null    object
 2   date             816 non-null    object
 3   player_of_match  812 non-null    object
 4   venue            816 non-null    object
 5   neutral_venue    816 non-null    int64
 6   team1            816 non-null    object
 7   team2            816 non-null    object
 8   toss_winner      816 non-null    object
 9   toss_decision    816 non-null    object
 10  winner           812 non-null    object
 11  result           812 non-null    object
 12  result_margin    799 non-null    float64
 13  eliminator       812 non-null    object
 14  method           19 non-null     object
 15  umpire1          816 non-null    object
 16  umpire2          816 non-null    object
dtypes: float64(1), int64(2), object(14)
memory usage: 108.5+ KB
```

In [40]:
```
pd.read_csv('IPL Matches 2008-2020.csv',parse_dates=['date']).info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 816 entries, 0 to 815
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   id               816 non-null    int64
 1   city             803 non-null    object
 2   date             816 non-null    datetime64[ns]
 3   player_of_match  812 non-null    object
 4   venue            816 non-null    object
 5   neutral_venue    816 non-null    int64
 6   team1            816 non-null    object
 7   team2            816 non-null    object
 8   toss_winner      816 non-null    object
 9   toss_decision    816 non-null    object
 10  winner           812 non-null    object
 11  result           812 non-null    object
 12  result_margin    799 non-null    float64
 13  eliminator       812 non-null    object
 14  method           19 non-null     object
 15  umpire1          816 non-null    object
 16  umpire2          816 non-null    object
dtypes: datetime64[ns](1), float64(1), int64(2), object(13)
memory usage: 108.5+ KB
```

In [41]:
```
pd.read_csv('IPL Matches 2008-2020.csv',parse_dates=['date']).info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 816 entries, 0 to 815
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   id               816 non-null    int64
 1   city             803 non-null    object
 2   date             816 non-null    datetime64[ns]
 3   player_of_match  812 non-null    object
 4   venue            816 non-null    object
 5   neutral_venue    816 non-null    int64
 6   team1            816 non-null    object
 7   team2            816 non-null    object
 8   toss_winner      816 non-null    object
 9   toss_decision    816 non-null    object
 10  winner           812 non-null    object
 11  result           812 non-null    object
 12  result_margin    799 non-null    float64
 13  eliminator       812 non-null    object
 14  method           19 non-null     object
 15  umpire1          816 non-null    object
 16  umpire2          816 non-null    object
dtypes: datetime64[ns](1), float64(1), int64(2), object(13)
memory usage: 108.5+ KB
```

In [42]:
```
pd.read_csv('IPL Matches 2008-2020.csv)
```

```
  Cell In[42], line 1
    pd.read_csv('IPL Matches 2008-2020.csv)
                ^
SyntaxError: unterminated string literal (detected at line 1)
```

14. Convertors

In [44]:
```
pd.read_csv('IPL Matches 2008-2020.csv',converters={'team1':rename})
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[44], line 1
----> 1 pd.read_csv('IPL Matches 2008-2020.csv',converters={'team1':rename})

NameError: name 'rename' is not defined
```

15. na_values parameter

In [45]:
```
pd.read_csv('aug_train.csv')
```

Out[45]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_univers
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollme
1	29725	city_40	0.776	Male	No relevent experience	no_enrollme
2	11561	city_21	0.624	NaN	No relevent experience	Full time cou
3	33241	city_115	0.789	NaN	No relevent experience	N
4	666	city_162	0.767	Male	Has relevent experience	no_enrollme
...	...	...	...	...	...	
19153	7386	city_173	0.878	Male	No relevent experience	no_enrollme
19154	31398	city_103	0.920	Male	Has relevent experience	no_enrollme
19155	24576	city_103	0.920	Male	Has relevent experience	no_enrollme
19156	5756	city_65	0.802	Male	Has relevent experience	no_enrollme
19157	23834	city_67	0.855	NaN	No relevent experience	no_enrollme

19158 rows × 14 columns

In [46]:
```
pd.read_csv('aug_train.csv',na_values=['Male',])
```

Out[46]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_univers
0	8949	city_103	0.920	NaN	Has relevent experience	no_enrollme
1	29725	city_40	0.776	NaN	No relevent experience	no_enrollme
2	11561	city_21	0.624	NaN	No relevent experience	Full time cou
3	33241	city_115	0.789	NaN	No relevent experience	N
4	666	city_162	0.767	NaN	Has relevent experience	no_enrollme
...	...	...	...	...	...	
19153	7386	city_173	0.878	NaN	No relevent experience	no_enrollme
19154	31398	city_103	0.920	NaN	Has relevent experience	no_enrollme
19155	24576	city_103	0.920	NaN	Has relevent experience	no_enrollme
19156	5756	city_65	0.802	NaN	Has relevent experience	no_enrollme
19157	23834	city_67	0.855	NaN	No relevent experience	no_enrollme

19158 rows × 14 columns

16. Loading a huge dataset in chunks

In [47]:
```
pd.read_csv('aug_train.csv')
```

Out[47]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_univers
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollme
1	29725	city_40	0.776	Male	No relevent experience	no_enrollme
2	11561	city_21	0.624	NaN	No relevent experience	Full time cou
3	33241	city_115	0.789	NaN	No relevent experience	N
4	666	city_162	0.767	Male	Has relevent experience	no_enrollme
...	...	...	...	...	...	
19153	7386	city_173	0.878	Male	No relevent experience	no_enrollme
19154	31398	city_103	0.920	Male	Has relevent experience	no_enrollme
19155	24576	city_103	0.920	Male	Has relevent experience	no_enrollme
19156	5756	city_65	0.802	Male	Has relevent experience	no_enrollme
19157	23834	city_67	0.855	NaN	No relevent experience	no_enrollme

19158 rows × 14 columns

In [56]:
```
dfs = pd.read_csv('aug_train.csv',chunksize=5000)
```

```
  Cell In[56], line 2
    for chunks in dfs:
                      ^
SyntaxError: incomplete input
```

In [58]:
```
for chunks in dfs:
    print(chunk.shape)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[58], line 2
      1 for chunks in dfs:
----> 2     print(chunk.shape)

NameError: name 'chunk' is not defined
```

In [59]:
```
pd.read_csv('aug_train.csv')
```

Out[59]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_univers
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollme
1	29725	city_40	0.776	Male	No relevent experience	no_enrollme
2	11561	city_21	0.624	NaN	No relevent experience	Full time cou
3	33241	city_115	0.789	NaN	No relevent experience	N
4	666	city_162	0.767	Male	Has relevent experience	no_enrollme
...	...	...	...	...	...	
19153	7386	city_173	0.878	Male	No relevent experience	no_enrollme
19154	31398	city_103	0.920	Male	Has relevent experience	no_enrollme
19155	24576	city_103	0.920	Male	Has relevent experience	no_enrollme
19156	5756	city_65	0.802	Male	Has relevent experience	no_enrollme
19157	23834	city_67	0.855	NaN	No relevent experience	no_enrollme

19158 rows × 14 columns

In [63]:
```
dfs = pd.read_csv('aug_train.csv',chunksize=5000)
for chunks in dfs:
```

```
  Cell In[63], line 2
    for chunks in dfs:
                      ^
SyntaxError: incomplete input
```

In [62]:
```

```

```
  Cell In[62], line 1
    for chunks in dfs:
                      ^
SyntaxError: incomplete input
```

Working with JSON/SQL

In [64]:
```
import pandas as pd
```

In [65]:
```
pd.read_json('train.json')
```

Out[65]:

	id	cuisine	ingredients
0	10259	greek	[romaine lettuce, black olives, grape tomatoes...
1	25693	southern_us	[plain flour, ground pepper, salt, tomatoes, g...
2	20130	filipino	[eggs, pepper, salt, mayonaise, cooking oil, g...
3	22213	indian	[water, vegetable oil, wheat, salt]
4	13162	indian	[black pepper, shallots, cornflour, cayenne pe...
...	...	...	...
39769	29109	irish	[light brown sugar, granulated sugar, butter, ...
39770	11462	italian	[KRAFT Zesty Italian Dressing, purple onion, b...
39771	2238	irish	[eggs, citrus fruit, raisins, sourdough starte...
39772	41882	chinese	[boneless chicken skinless thigh, minced garli...
39773	2362	mexican	[green chile, jalapeno chilies, onions, ground...

39774 rows × 3 columns

In [66]:

```
pd.read_json('https://api.exchangerate-api.com/v4/latest/INR')
```

Out[66]:

	provider	WARNING_UPGRADE_TO_V6	terms	base	date
AED	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023-11-23
AFN	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023-11-23
ALL	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023-11-23
AMD	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023-11-23
ANG	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023-11-23
...	...	...	...	...	...
XPF	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023-11-23
YER	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023-11-23
ZAR	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023-11-23
ZMW	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023-11-23
ZWL	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023-11-23

162 rows × 7 columns

Working with SQL

In [67]:
```
!pip install mysql.connector
```

```
Collecting mysql.connector
  Downloading mysql-connector-2.2.9.tar.gz (11.9 MB)
     ---------------------------------------- 0.0/11.9 MB ? eta -:--:--
     ---------------------------------------- 0.0/11.9 MB ? eta -:--:--
     ---------------------------------------- 0.1/11.9 MB 1.0 MB/s eta 0:00:12
      --------------------------------------- 0.3/11.9 MB 1.5 MB/s eta 0:00:08
     - -------------------------------------- 0.4/11.9 MB 1.9 MB/s eta 0:00:06
     -- ------------------------------------- 0.6/11.9 MB 2.3 MB/s eta 0:00:05
     -- ------------------------------------- 0.9/11.9 MB 2.7 MB/s eta 0:00:04
     ---- ----------------------------------- 1.4/11.9 MB 3.7 MB/s eta 0:00:03
     ------ --------------------------------- 1.9/11.9 MB 4.3 MB/s eta 0:00:03
     ------- -------------------------------- 2.2/11.9 MB 4.6 MB/s eta 0:00:03
     --------- ------------------------------ 2.7/11.9 MB 4.8 MB/s eta 0:00:02
     ---------- ----------------------------- 3.0/11.9 MB 5.1 MB/s eta 0:00:02
     ----------- ---------------------------- 3.4/11.9 MB 5.2 MB/s eta 0:00:02
     ------------- -------------------------- 4.0/11.9 MB 5.6 MB/s eta 0:00:02
     -------------- ------------------------- 4.3/11.9 MB 5.7 MB/s eta 0:00:02
     --------------- ------------------------ 4.6/11.9 MB 5.7 MB/s eta 0:00:02
     --------------- ------------------------ 4.7/11.9 MB 5.5 MB/s eta 0:00:02
     ----------------- ---------------------- 5.2/11.9 MB 5.3 MB/s eta 0:00:02
     ------------------ --------------------- 5.4/11.9 MB 5.1 MB/s eta 0:00:02
     -------------------- ------------------- 6.2/11.9 MB 5.6 MB/s eta 0:00:02
     ---------------------- ----------------- 6.8/11.9 MB 5.6 MB/s eta 0:00:01
     ------------------------ --------------- 7.1/11.9 MB 5.8 MB/s eta 0:00:01
     ------------------------ --------------- 7.4/11.9 MB 5.7 MB/s eta 0:00:01
     ------------------------- -------------- 7.6/11.9 MB 5.6 MB/s eta 0:00:01
     --------------------------- ------------ 8.0/11.9 MB 5.6 MB/s eta 0:00:01
     ---------------------------- ----------- 8.4/11.9 MB 5.7 MB/s eta 0:00:01
     ------------------------------ --------- 9.0/11.9 MB 5.8 MB/s eta 0:00:01
     ------------------------------- -------- 9.5/11.9 MB 5.9 MB/s eta 0:00:01
     --------------------------------- ------ 9.8/11.9 MB 6.0 MB/s eta 0:00:01
     ---------------------------------- ----- 10.4/11.9 MB 6.4 MB/s eta 0:00:01
     ------------------------------------ --- 10.8/11.9 MB 6.8 MB/s eta 0:00:01
     ------------------------------------- -- 11.2/11.9 MB 7.0 MB/s eta 0:00:01
     ---------------------------------------  11.6/11.9 MB 6.9 MB/s eta 0:00:01
     ---------------------------------------  11.8/11.9 MB 7.0 MB/s eta 0:00:01
     ---------------------------------------  11.8/11.9 MB 7.0 MB/s eta 0:00:01
     ---------------------------------------- 11.9/11.9 MB 6.4 MB/s eta 0:00:00
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Building wheels for collected packages: mysql.connector
  Building wheel for mysql.connector (setup.py): started
  Building wheel for mysql.connector (setup.py): finished with status 'done'
  Created wheel for mysql.connector: filename=mysql_connector-2.2.9-cp311-cp311-wi
n_amd64.whl size=247958 sha256=7cb80c9a2740fd25dc6d73e14c769a1f045366bc132788cf320
6feca09ff8a69
  Stored in directory: c:\users\asus\appdata\local\pip\cache\wheels\17\cd\ed\2d49e
9bac69cf09382e4c7cc20a2511202b48324b87db26019
Successfully built mysql.connector
Installing collected packages: mysql.connector
Successfully installed mysql.connector-2.2.9
```

In [68]:
```
import mysql.connector
```

In [72]:
```
conn = mysql.connector.connect(host='localhost',user='root',password='',database='w
```

```
---------------------------------------------------------------------------
ProgrammingError                          Traceback (most recent call last)
Cell In[72], line 1
----> 1 conn = mysql.connector.connect(host='localhost',user='root',password='',da
tabase='world')

File ~\anaconda3\Lib\site-packages\mysql\connector\__init__.py:179, in connect(*ar
gs, **kwargs)
    177     return CMySQLConnection(*args, **kwargs)
    178 else:
--> 179     return MySQLConnection(*args, **kwargs)

File ~\anaconda3\Lib\site-packages\mysql\connector\connection.py:95, in MySQLConne
ction.__init__(self, *args, **kwargs)
     92 self._pool_config_version = None
     94 if len(kwargs) > 0:
---> 95     self.connect(**kwargs)

File ~\anaconda3\Lib\site-packages\mysql\connector\abstracts.py:716, in MySQLConne
ctionAbstract.connect(self, **kwargs)
    713     self.config(**kwargs)
    715 self.disconnect()
--> 716 self._open_connection()
    717 self._post_connection()

File ~\anaconda3\Lib\site-packages\mysql\connector\connection.py:208, in MySQLConn
ection._open_connection(self)
    206 self._socket.open_connection()
    207 self._do_handshake()
--> 208 self._do_auth(self._user, self._password,
    209                 self._database, self._client_flags, self._charset_id,
    210                 self._ssl)
    211 self.set_converter_class(self._converter_class)
    212 if self._client_flags & ClientFlag.COMPRESS:

File ~\anaconda3\Lib\site-packages\mysql\connector\connection.py:144, in MySQLConn
ection._do_auth(self, username, password, database, client_flags, charset, ssl_opt
ions)
    137 packet = self._protocol.make_auth(
    138     handshake=self._handshake,
    139     username=username, password=password, database=database,
    140     charset=charset, client_flags=client_flags,
    141     ssl_enabled=self._ssl_active,
    142     auth_plugin=self._auth_plugin)
    143 self._socket.send(packet)
--> 144 self._auth_switch_request(username, password)
    146 if not (client_flags & ClientFlag.CONNECT_WITH_DB) and database:
    147     self.cmd_init_db(database)

File ~\anaconda3\Lib\site-packages\mysql\connector\connection.py:173, in MySQLConn
ection._auth_switch_request(self, username, password)
    171 packet = self._socket.recv()
    172 if packet[4] != 1:
--> 173     return self._handle_ok(packet)
    174 else:
    175     auth_data = self._protocol.parse_auth_more_data(packet)

File ~\anaconda3\Lib\site-packages\mysql\connector\connection.py:331, in MySQLConn
ection._handle_ok(self, packet)
    329     return ok_pkt
    330 elif packet[4] == 255:
--> 331     raise errors.get_exception(packet)
    332 raise errors.InterfaceError('Expected OK packet')
```

```
ProgrammingError: 1045 (28000): Access denied for user 'root'@'localhost' (using p
assword: NO)
```

In [73]:
```
df = pd.read_sql_query("SELECT * FROM countrylanguage",conn)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[73], line 1
----> 1 df = pd.read_sql_query("SELECT * FROM countrylanguage",conn)

NameError: name 'conn' is not defined
```

In [74]:
```
df
```

Out[74]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_univers
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollme
1	29725	city_40	0.776	Male	No relevent experience	no_enrollme
2	11561	city_21	0.624	NaN	No relevent experience	Full time cou
3	33241	city_115	0.789	NaN	No relevent experience	N
4	666	city_162	0.767	Male	Has relevent experience	no_enrollme
...	...	...	...	...	...	
19153	7386	city_173	0.878	Male	No relevent experience	no_enrollme
19154	31398	city_103	0.920	Male	Has relevent experience	no_enrollme
19155	24576	city_103	0.920	Male	Has relevent experience	no_enrollme
19156	5756	city_65	0.802	Male	Has relevent experience	no_enrollme
19157	23834	city_67	0.855	NaN	No relevent experience	no_enrollme

19158 rows × 14 columns

In [ ]:

Data Science | 30 Days of Machine Learning | Day - 3

Educator Name: Nishant Dhote
Support Team: **+91-7880-113-112**

 # Machine Learning Development Life Cycle (MLDLC/MLDC):

 # Data science life cycle (DSLC):

 # Tools used in Machine Learning? Installing: Anaconda | Jupiter Notebook (IDEs)

Optional Tools: Spyder | PyCharm | Noteable | Google Colab | Kaggle Notebooks | Microsoft Azure Notebooks | Apache Zeplin | Count.co and Many More

#How to import dataset and download data files

#How we create virtual environment

#Data Gathering

#Working with CSV Files

#Working with JSON/SQL

#Fetching data from an API

#Fetching data using web scraping

#Framing a Machine Learning Problem

Industries Helping Hands

 # Machine Learning Development Life Cycle (MLDLC/MLDC):

1. Frame the problem
2. Gathering data
3. Data pre processing
4. Exploratory data analysis
5. Feature engineering & selection
6. Model training, evaluation and selection
7. Model deployment
8. Testing
9. Optimize

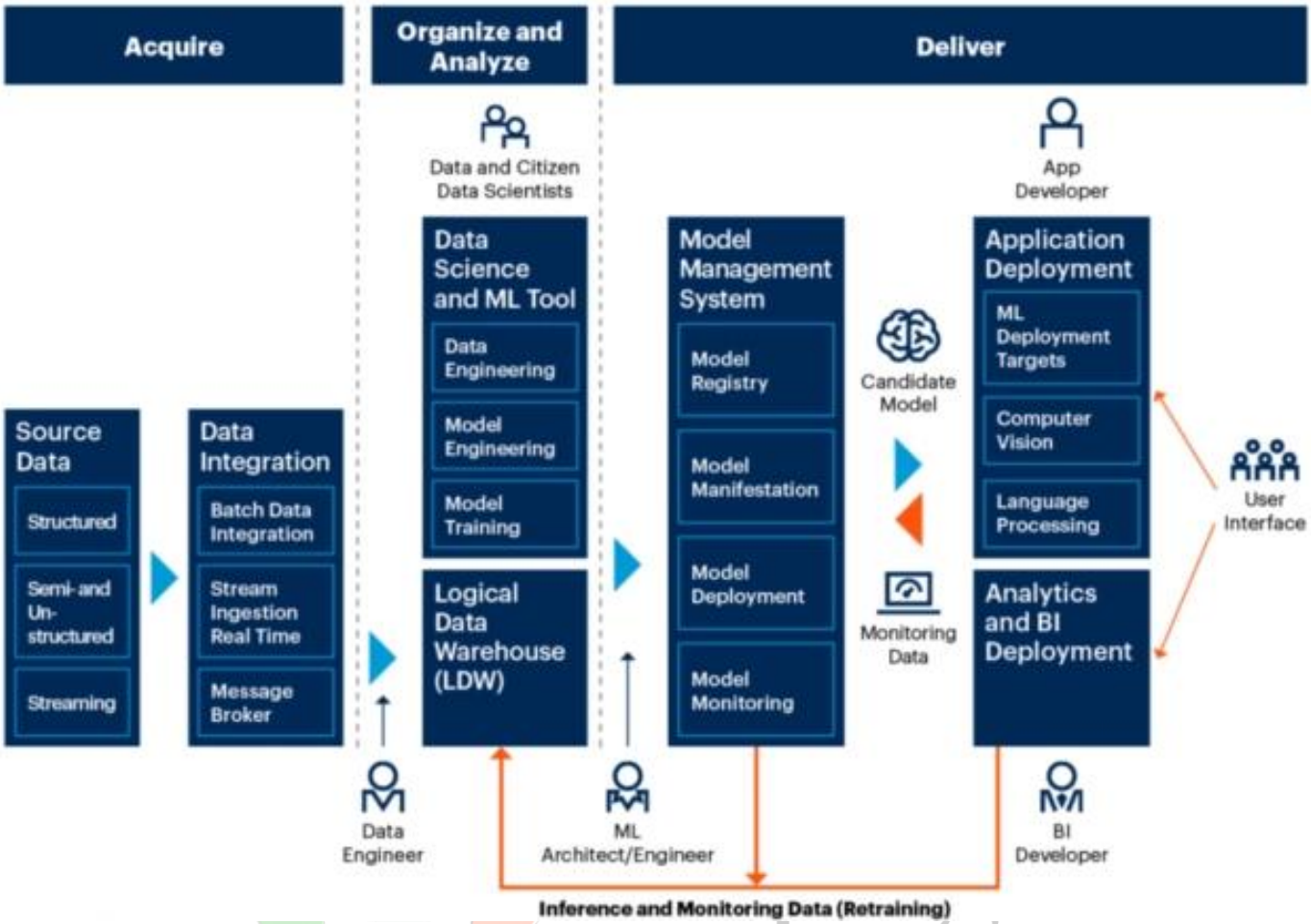
   Note: Some books mention in 10-12 steps in MLDLC.

What is SDLC?

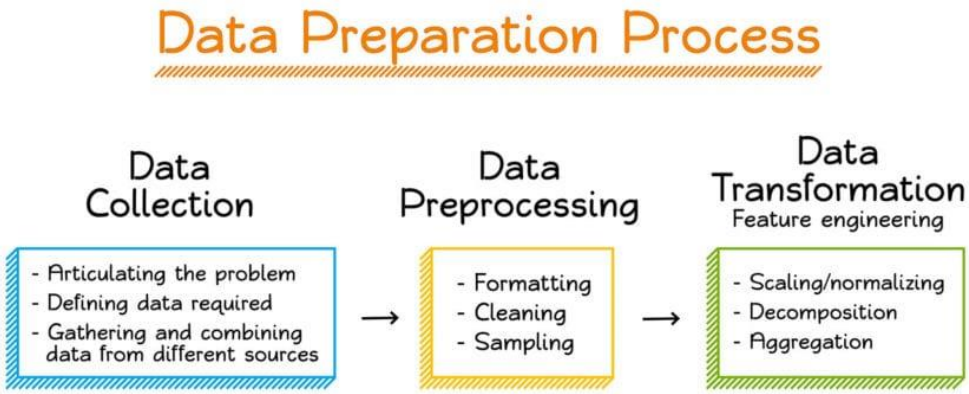
The software development lifecycle (SDLC) is the cost-effective and time-efficient process that development teams use to design and build high-quality software. The goal of SDLC is to minimize project risks through forward planning so that software meets customer expectations during production and beyond. This methodology outlines a series of steps that divide the software development process into tasks you can assign, complete, and measure.

Read Blog: Click Here

Architecture to Operationalize MLDLC

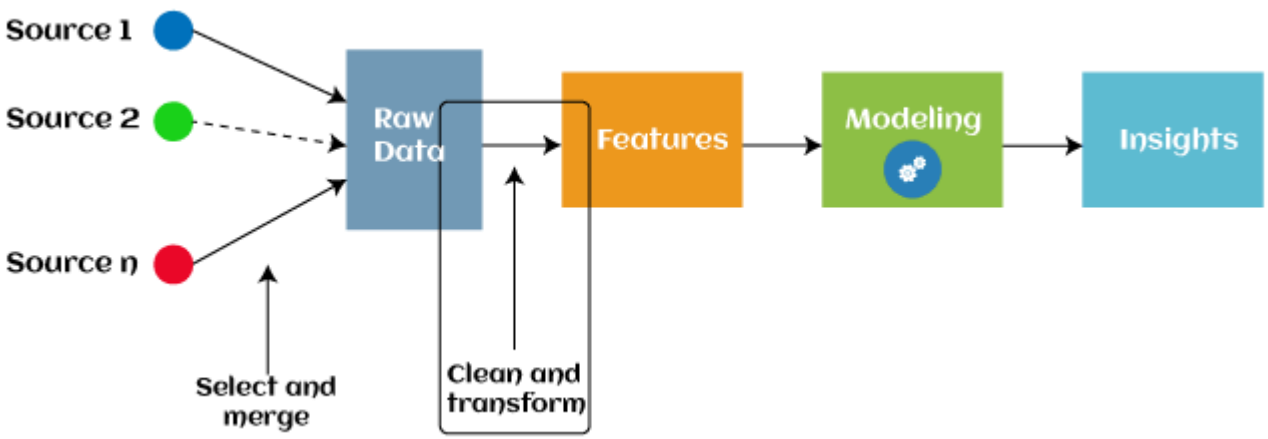


1. **Frame the problem:** The first stage of MLDLC is all about "What do we want?" Project planning is a vital role in the software delivery lifecycle since this is the part where the team estimates the cost and defines the requirements of the new software.
2. **Gathering data:** The second step of MLDLC is gathering maximum information from the client requirements for the product. Discuss each detail and specification of the product with the customer. Data collection means pooling data by scraping, capturing, and loading it from multiple sources, including offline and online sources. High volumes of data collection or data creation can be the hardest part of a machine learning project, especially at scale.
3. **Data pre-processing:** Data pre-processing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models**.**

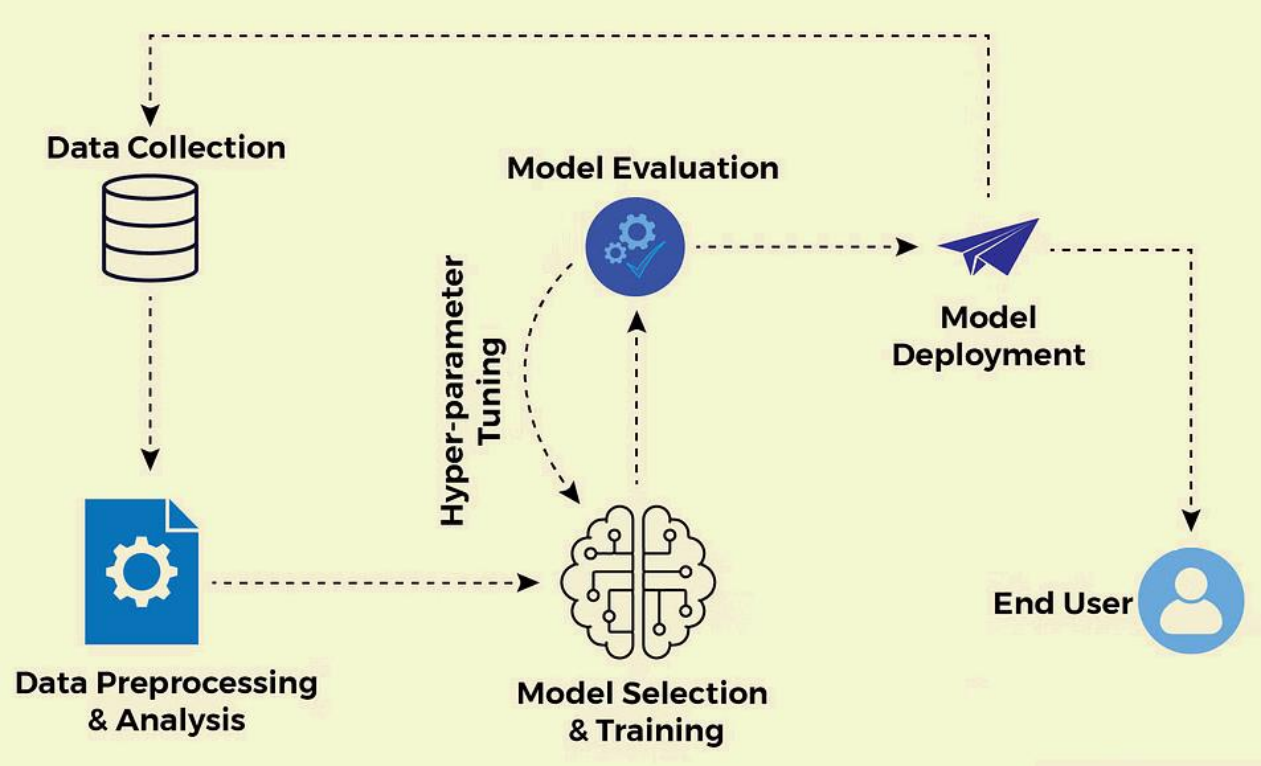


4. **Exploratory Data Analysis (EDA):** Exploratory data analysis (EDA) is used by data scientists to analyse and investigate data sets and summarize their main characteristics, often employing data visualization methods.

5. **Feature Engineering & Selection:** Feature engineering in Machine learning consists of mainly 5 processes: Feature Creation, Feature Transformation, Feature Extraction, Feature Selection, and Feature Scaling. It is an iterative process that requires experimentation and testing to find the best combination of features for a given problem.

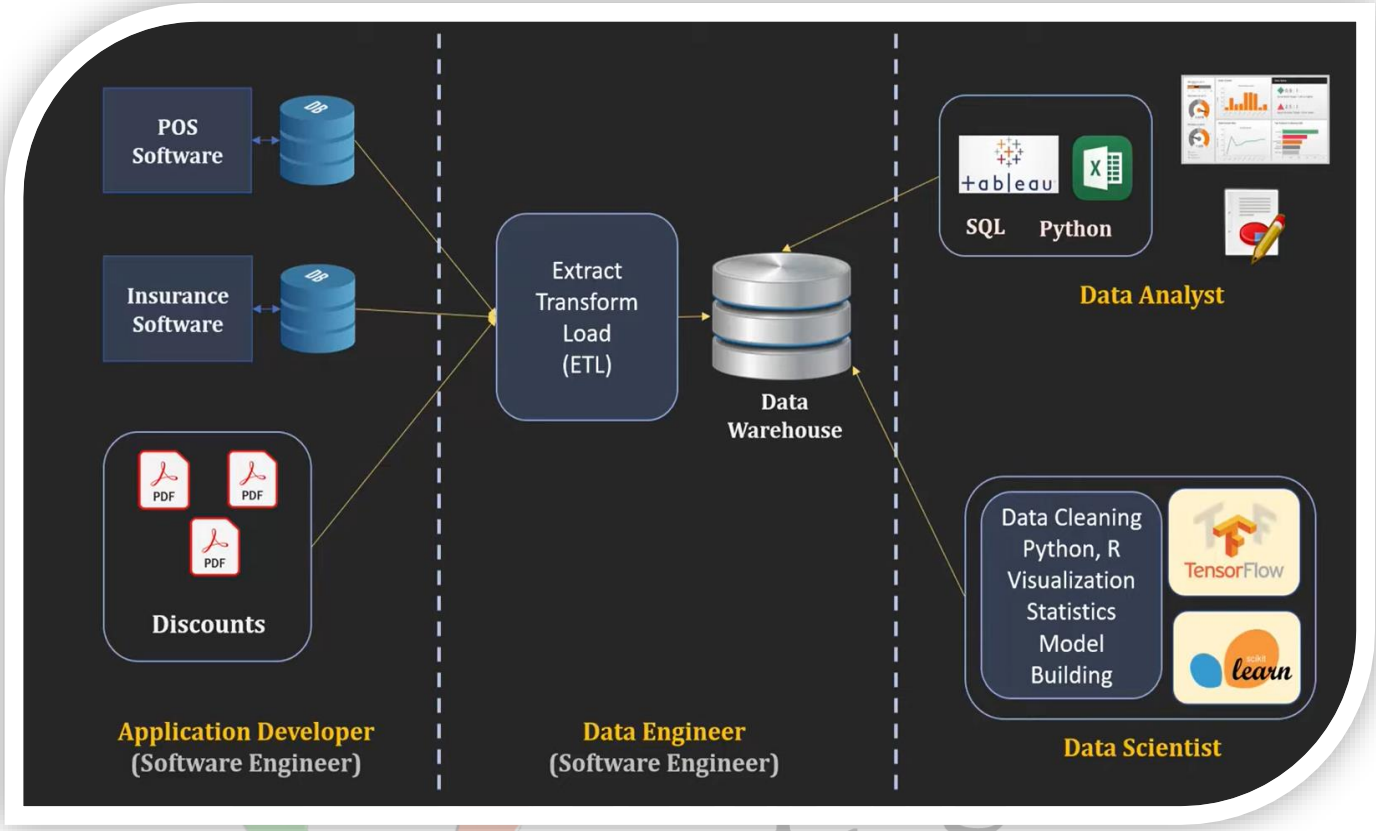


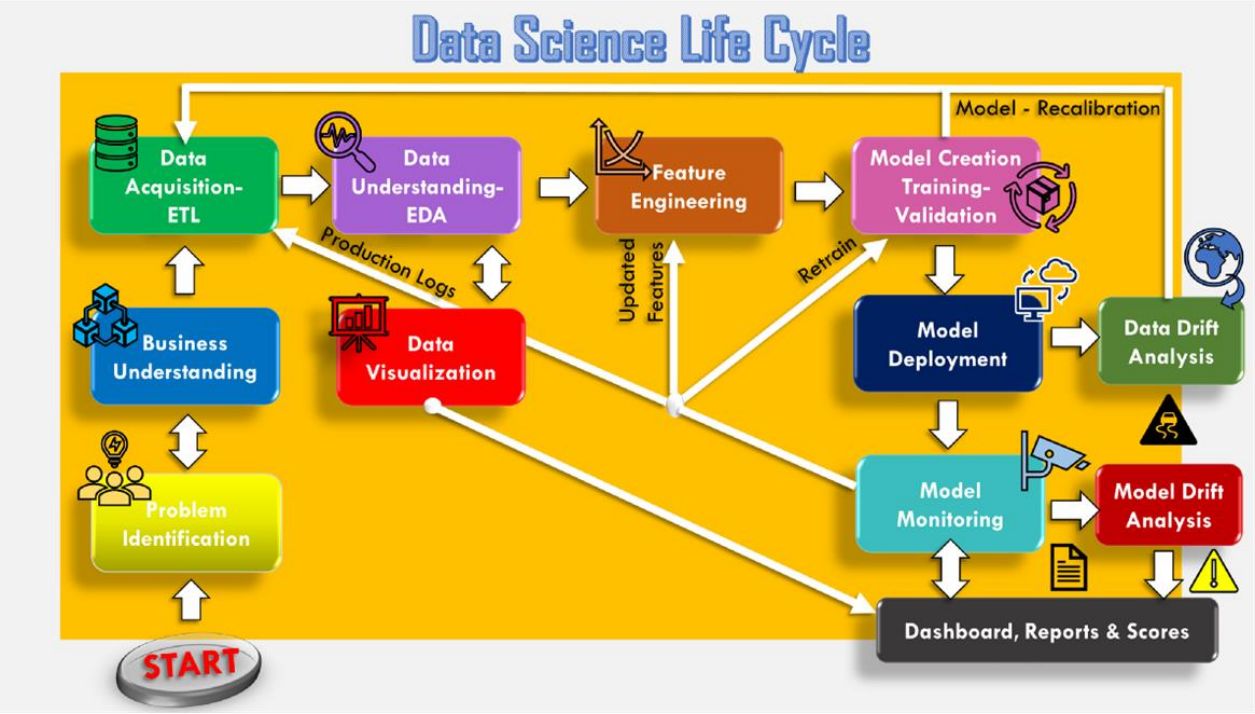
6. **Model training, evaluation and selection:** Model evaluation is the process that uses some metrics which help us to analyse the performance of the model. As we all know that model development is a multi-step process and a check should be kept on how well the model generalizes future predictions. Therefore, evaluating a model plays a vital role so that we can judge the performance of our model.

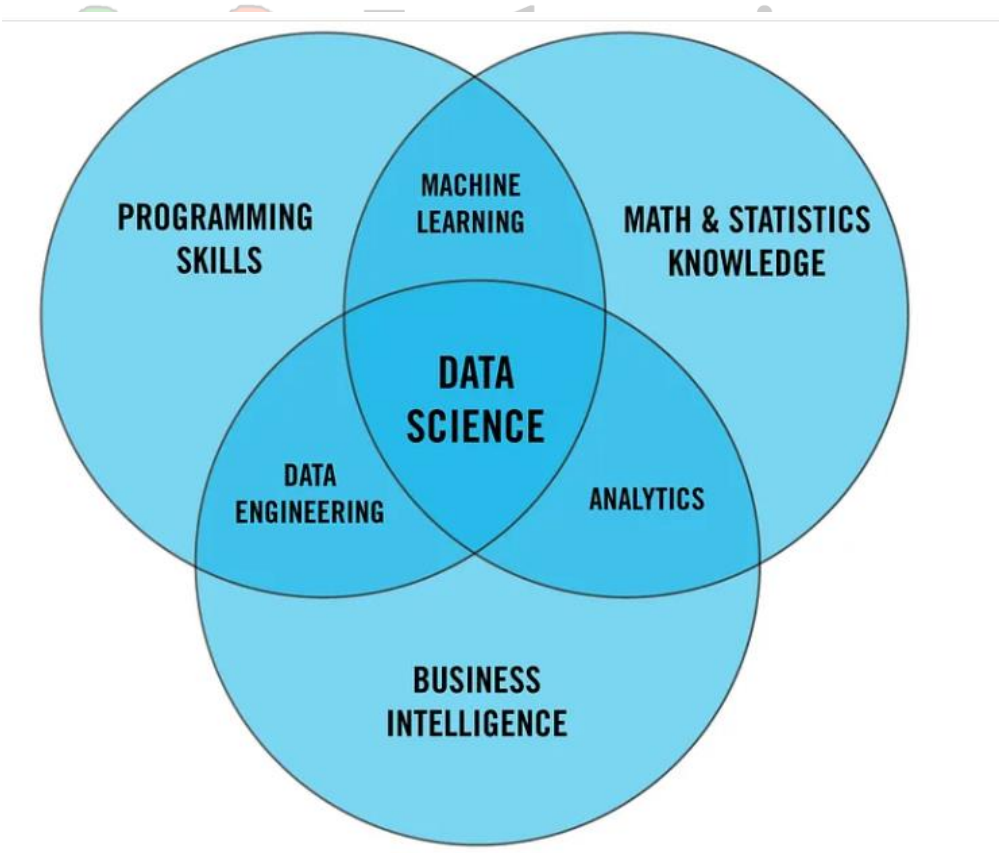


7. **Model deployment:** Deploying a machine learning model, known as model deployment, simply means to integrate a machine learning model and integrate it into an existing production environment.
8. **Testing:** In machine learning, model testing is referred to as the process where the performance of a fully trained model is evaluated on a testing set.
9. **Optimize:** Machine learning optimisation is the process of iteratively improving the accuracy of a machine learning model, lowering the degree of error. Machine learning models learn to generalise and make predictions about new live data based on insight learned from training data.

Data Scientist Work?
- DATA ANALYST Vs DATA SCIENTIST Vs DATA ENGINEER




Data Science Life Cycle
Model - Recalibration
Data Acquisition-ETL
Data Understanding-EDA
Feature Engineering
Model Creation Training-Validation
Production Logs
Updated Features
Retrain
Business Understanding
Data Visualization
Model Deployment
Data Drift Analysis
Problem Identification
Model Monitoring
Model Drift Analysis
Dashboard, Reports & Scores
START


PROGRAMMING SKILLS
MACHINE LEARNING
MATH & STATISTICS KNOWLEDGE
DATA SCIENCE
DATA ENGINEERING
ANALYTICS
BUSINESS INTELLIGENCE

Tools used in Machine Learning?

Installing:

Anaconda: https://www.anaconda.com/download

Jupiter Notebook (IDEs): https://jupyter.org/

Discuss Notebook Interface and Navigations.

Installation Video Link : https://youtu.be/AS1a5K8zejk?si=ilKogijLjqdjhThi

#How to import dataset and download data files

Step 1 - Upload dataset in the system folder.

Step 2 – Run Code

```
# pandas
import pandas as pd
```

Step 3 – Run → pd.read_csv('hotel_bookings.csv')

If you want to Download Notebook so click on File→ Download us

#How we create virtual environment

What Is a Virtual Environment?

The main purpose of Python virtual environments is to create an isolated environment for Python projects. This means that each project can have its own dependencies, regardless of what dependencies every other project has.

The great thing about this is that there are no limits to the number of environments you can have since they're just directories containing a few scripts. Plus, they're easily created.

Required Dataset Links: https://github.com/TheiScale/30_Days_Machine_Learning

#Data Gathering

#Working with CSV Files

1. Importing pandas

```
import pandas as pd
```

2. Opening a local csv file

```
df = pd.read_csv('aug_train.csv')
```

3. Opening a csv file from an URL

URL Link: https://raw.githubusercontent.com/cs109/2014_data/master/countries.csv

```
import requests

from io import StringIO

url = "https://raw.githubusercontent.com/cs109/2014_data/master/countries.csv"

headers = {"User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.0) Gecko/20100101 Firefox/66.0"}

req = requests.get(url, headers=headers)

data = StringIO(req.text)

pd.read_csv(data)
```

4. Sep Parameter

Step 1 : pd.read_csv('movie_titles_metadata.tsv')

Step 2 :

```
pd.read_csv('movie_titles_metadata.tsv',sep='\t')
```

Step 3 :

```
pd.read_csv('movie_titles_metadata.tsv',sep='\t',names=['sno','name','release_year','rating','votes','genres'])
```

5. Index_col parameter

Step 1 : pd.read_csv('aug_train.csv')

Step 2: pd.read_csv('aug_train.csv',index_col='enrollee_id')

6. Header parameter

Step 1 : pd.read_csv('test.csv')

Step 2 : pd.read_csv('test.csv',header=1)

7. Use_col parameter

pd.read_csv('aug_train.csv',usecols=['enrollee_id','gender','education_level'])

8. Squeeze parameters

pd.read_csv('aug_train.csv',usecols=['gender'],squeeze=True)

9. Skiprows/nrows Parameter

Step 1 : pd.read_csv('aug_train.csv',skiprows=[0,1])

pd.read_csv('aug_train.csv',nrows=100)

10. Encoding parameter

Step 1 : pd.read_csv('zomato.csv')

pd.read_csv('zomato.csv',encoding='latin-1')

11. Skip bad lines

Step 1 : pd.read_csv('zomato.csv', address=';', encoding="latin-1")

pd.read_csv('BX-Books.csv', address=';', encoding="latin-1",error_bad_lines=False)

12.dtypes parameter

Step 1 : pd.read_csv('aug_train.csv').info()

```
pd.read_csv('aug_train.csv',dtype={'target':int}).info()
```

13. Handling Dates

Step 1 : pd.read_csv('IPL Matches 2008-2020.csv').info()

```
pd.read_csv('IPL Matches 2008-2020.csv',parse_dates=['date']).info()
```

14. Convertors

```
pd.read_csv('IPL Matches 2008-2020.csv',converters={'team1':rename})
------------------------
def rename(name):
    if name == "Royal Challengers Bangalore":
        return "RCB"
    else:
        return name
------------------------
IN: rename("Royal Challengers Bangalore")
OUT: 'RCB'
```

15. na_values parameter

Step 1 : pd.read_csv('aug_train.csv')

```
pd.read_csv('aug_train.csv',na_values=['Male',])
```

16. Loading a huge dataset in chunks

Step 1 : pd.read_csv('aug_train.csv')

```
dfs = pd.read_csv('aug_train.csv',chunksize=5000)
for chunks in dfs:
    print(chunk.shape)
```

#Working with JSON/SQL

JSON : JavaScript Object Notation.

API → JSON → JAVA | Python

What is JSON used for?

JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax. It is commonly used for transmitting data in web applications (e.g., sending some data from the server to the client, so it can be displayed on a web page, or vice versa)

Code : Jupiter Notebook

```
import pandas as pd
```

Woking with JSON

```
pd.read_json('train.json')
```

URL Data with JSON

```
pd.read_json('https://api.exchangerate-api.com/v4/latest/INR')
```

Working with SQL (Structured Query Language)

SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database.

Xampp download link: https://www.apachefriends.org/index.html

XAMPP helps a local host or server to test its website and clients via computers and laptops before releasing it to the main server. It is a platform that furnishes a suitable environment to test and verify the working of projects based on Apache, Perl, MySQL database, and PHP through the system of the host itself.

SQL Series Xampp Install : https://youtu.be/ZcDm262cam4?si=tkU5tEs5G3RqAm0o

SQL dataset : https://github.com/TheiScale/30_Days_Machine_Learning

Pandas read_json documentation:
https://pandas.pydata.org/docs/reference/api/pandas.read_json.html

Pandas read_sql_query documentation:
https://pandas.pydata.org/docs/reference/api/pandas.read_sql_query.html#pandas.read_sql_query

Code Jupiter Notebook:

Working with SQL

```
!pip install mysql.connector
---------------------------
import mysql.connector
---------------------------
conn =
mysql.connector.connect(host='localhost',user='root',password='',database='world')
--------------------
pd.read_sql_query("SELECT * FROM city",conn)
----or----
df = pd.read_sql_query("SELECT * FROM countrylanguage",conn)
df
```

#Framing a Machine Learning Problem

#Data Gathering

#Fetching data from an API

#Fetching data using web scraping

Data Story Telling: Curious Data Minds

Case Study Link: https://data-flair.training/blogs/data-science-in-agriculture/

https://intellias.com/how-to-encourage-farmers-to-use-big-data-analytics-in-agriculture/

Day 4 Machine Learning | iScale

In [1]:
```
import pandas as pd
import requests
```

API KEY Paste

In [2]:
```
response = requests.get('https://api.themoviedb.org/3/movie/top_rated?api_key=5bc643f
```

6 Col Select code

In [3]:
```
df = pd.DataFrame(response.json()['results'])[['id','title','release_date','overview'
```

DF Show

In [5]:
```
df.head()
```

Out[5]:

	id	title	release_date	overview	popularity	vote_average
0	238	The Godfather	1972-03-14	Spanning the years 1945 to 1955, a chronicle o...	146.546	8.709
1	278	The Shawshank Redemption	1994-09-23	Framed in the 1940s for the double murder of h...	127.603	8.705
2	240	The Godfather Part II	1974-12-20	In the continuing saga of the Corleone crime f...	77.946	8.590
3	424	Schindler's List	1993-12-15	The true story of how businessman Oskar Schind...	86.658	8.571
4	19404	Dilwale Dulhania Le Jayenge	1995-10-20	Raj is a rich, carefree, happy-go-lucky second...	50.500	8.546

Get a data from page 5

In [6]:
```
response = requests.get('https://api.themoviedb.org/3/movie/top_rated?api_key=5bc643f
```

In [7]:
```
df1 = pd.DataFrame(response.json()['results'])[['id','title','release_date','overview
```

In [8]:
```
df1
```

Out[8]:

	id	title	release_date	overview	popularity	vote_average
0	16672	Woman in the Dunes	1964-02-15	An entomologist suffers extreme psychological ...	22.474	8.274
1	670	Oldboy	2003-11-21	With no clue how he came to be imprisoned, dru...	54.408	8.300
2	42269	We All Loved Each Other So Much	1974-12-21	Three partisans bound by a strong friendship r...	12.403	8.265
3	10376	The Legend of 1900	1998-10-28	The story of a virtuoso piano player who lives...	23.066	8.265
4	20334	The Shop Around the Corner	1940-01-12	Two employees at a gift shop can barely stand ...	19.810	8.264
5	1585	It's a Wonderful Life	1946-12-20	A holiday favourite for generations... George...	52.639	8.264
6	315162	Puss in Boots: The Last Wish	2022-12-07	Puss in Boots discovers that his passion for a...	258.974	8.263
7	299534	Avengers: Endgame	2019-04-24	After the devastating events of Avengers: Infi...	171.100	8.261
8	635302	Demon Slayer -Kimetsu no Yaiba- The Movie: Mug...	2020-10-16	Tanjiro Kamado, joined with Inosuke Hashibira,...	93.825	8.257
9	618344	Justice League Dark: Apokolips War	2020-05-05	Earth is decimated after intergalactic tyrant ...	30.353	8.300
10	8587	The Lion King	1994-06-24	A young lion prince is cast out of his pride b...	112.646	8.256
11	1026227	There’s Still Tomorrow	2023-10-26	Delia is a typical housewife in post-war Italy...	15.630	8.252
12	299536	Avengers: Infinity War	2018-04-25	As the Avengers and their allies have continue...	255.971	8.252
13	361743	Top Gun: Maverick	2022-05-24	After more than thirty years of service as one...	166.031	8.248
14	265177	Mommy	2014-09-19	A peculiar neighbor offers hope to a recent wi...	22.680	8.246
15	508965	Klaus	2019-11-08	When Jesper distinguishes himself as the Posta...	45.654	8.245
16	504253	I Want to Eat Your Pancreas	2018-09-01	After his classmate and crush is diagnosed wit...	38.057	8.245
17	851644	20th Century Girl	2022-10-06	Yeon-du asks her best friend Bora to collect a...	93.524	8.244
18	626332	Flamin' Hot	2023-03-11	The inspiring true story of Richard Montañez, ...	70.358	8.244
19	490132	Green Book	2018-11-16	Tony Lip, a bouncer in 1962, is hired to drive...	78.481	8.244

In [10]:
```
df1.shape
```

Out[10]:
```
(20, 6)
```

In [11]:
```
df1.to_csv('movies.csv')
```

In [12]:
```
df1
```

Out[12]:

	id	title	release_date	overview	popularity	vote_average
0	16672	Woman in the Dunes	1964-02-15	An entomologist suffers extreme psychological ...	22.474	8.274
1	670	Oldboy	2003-11-21	With no clue how he came to be imprisoned, dru...	54.408	8.300
2	42269	We All Loved Each Other So Much	1974-12-21	Three partisans bound by a strong friendship r...	12.403	8.265
3	10376	The Legend of 1900	1998-10-28	The story of a virtuoso piano player who lives...	23.066	8.265
4	20334	The Shop Around the Corner	1940-01-12	Two employees at a gift shop can barely stand ...	19.810	8.264
5	1585	It's a Wonderful Life	1946-12-20	A holiday favourite for generations... George...	52.639	8.264
6	315162	Puss in Boots: The Last Wish	2022-12-07	Puss in Boots discovers that his passion for a...	258.974	8.263
7	299534	Avengers: Endgame	2019-04-24	After the devastating events of Avengers: Infi...	171.100	8.261
8	635302	Demon Slayer -Kimetsu no Yaiba- The Movie: Mug...	2020-10-16	Tanjiro Kamado, joined with Inosuke Hashibira,...	93.825	8.257
9	618344	Justice League Dark: Apokolips War	2020-05-05	Earth is decimated after intergalactic tyrant ...	30.353	8.300
10	8587	The Lion King	1994-06-24	A young lion prince is cast out of his pride b...	112.646	8.256
11	1026227	There’s Still Tomorrow	2023-10-26	Delia is a typical housewife in post-war Italy...	15.630	8.252
12	299536	Avengers: Infinity War	2018-04-25	As the Avengers and their allies have continue...	255.971	8.252
13	361743	Top Gun: Maverick	2022-05-24	After more than thirty years of service as one...	166.031	8.248
14	265177	Mommy	2014-09-19	A peculiar neighbor offers hope to a recent wi...	22.680	8.246
15	508965	Klaus	2019-11-08	When Jesper distinguishes himself as the Posta...	45.654	8.245
16	504253	I Want to Eat Your Pancreas	2018-09-01	After his classmate and crush is diagnosed wit...	38.057	8.245
17	851644	20th Century Girl	2022-10-06	Yeon-du asks her best friend Bora to collect a...	93.524	8.244
18	626332	Flamin' Hot	2023-03-11	The inspiring true story of Richard Montañez, ...	70.358	8.244
19	490132	Green Book	2018-11-16	Tony Lip, a bouncer in 1962, is hired to drive...	78.481	8.244

In [13]:
```
df.to_csv('movies_Page1.csv')
```

In [14]:
```
df
```

Out[14]:

	id	title	release_date	overview	popularity	vote_average
0	238	The Godfather	1972-03-14	Spanning the years 1945 to 1955, a chronicle o...	146.546	8.709
1	278	The Shawshank Redemption	1994-09-23	Framed in the 1940s for the double murder of h...	127.603	8.705
2	240	The Godfather Part II	1974-12-20	In the continuing saga of the Corleone crime f...	77.946	8.590
3	424	Schindler's List	1993-12-15	The true story of how businessman Oskar Schind...	86.658	8.571
4	19404	Dilwale Dulhania Le Jayenge	1995-10-20	Raj is a rich, carefree, happy-go-lucky second...	50.500	8.546
5	389	12 Angry Men	1957-04-10	The defense and the prosecution have rested an...	51.052	8.545
6	129	Spirited Away	2001-07-20	A young girl, Chihiro, becomes trapped in a st...	118.006	8.540
7	496243	Parasite	2019-05-30	All unemployed, Ki-taek's family takes peculia...	127.163	8.515
8	155	The Dark Knight	2008-07-16	Batman raises the stakes in his war on crime. ...	138.469	8.513
9	497	The Green Mile	1999-12-10	A supernatural tale set on death row in a Sout...	76.548	8.509
10	372058	Your Name.	2016-08-26	High schoolers Mitsuha and Taki are complete s...	92.874	8.505
11	680	Pulp Fiction	1994-09-10	A burger-loving hit man, his philosophical par...	84.105	8.489
12	122	The Lord of the Rings: The Return of the King	2003-12-01	Aragorn is revealed as the heir to the ancient...	112.150	8.476
13	13	Forrest Gump	1994-06-23	A man with a low IQ has accomplished great thi...	98.160	8.476
14	429	The Good, the Bad and the Ugly	1966-12-23	While the Civil War rages on between the Union...	70.084	8.470
15	769	GoodFellas	1990-09-12	The true story of Henry Hill, a half-Irish, ha...	85.372	8.466
16	12477	Grave of the Fireflies	1988-04-15	In the final months of World War II, 14-year-o...	0.600	8.456
17	637	Life Is Beautiful	1997-12-20	A touching story of an Italian book seller of ...	56.330	8.455
18	346	Seven Samurai	1954-04-26	A samurai answers a village's request for prot...	41.085	8.453
19	11216	Cinema Paradiso	1988-11-17	A filmmaker recalls his childhood, when he fel...	45.706	8.449

In [ ]:

Data Science | 30 Days of Machine Learning | Day - 4

Educator Name: Nishant Dhote
Support Team: **+91-7880-113-112**

#Data Gathering

1. **Working with CSV Files – Day 3**
2. **Working with JSON/SQL – Day 3**

#Framing a Machine Learning Problem

#Data Gathering

3. **Fetching data from an API**
4. **Fetching data using web scraping**

➔ Framing a Machine Learning Problem

Framing an ML problem
After verifying that your problem is best solved using either a predictive ML or a generative AI approach, you're ready to frame your problem in ML terms. You frame a problem in ML terms by completing the following tasks:

- Define the ideal outcome and the model's goal.
- Identify the model's output.
- Define success metrics.

Read Article: https://developers.google.com/machine-learning/problem-framing/ml-framing

1. **Business problem to ML problem**
2. **Type of problem**
3. **Current solution**
4. **Getting data**
5. **Metrics to measure**
6. **Online Vs Batch**
7. **Check assumptions**

Case Study Netflix – https://tinyurl.com/4mscpz2t

Step 1: Transaction Business problem to ML problem: Increases our revenue and decrease the churn rate.
Step 2: Identify the problem: Identify the customers who leave the platform. Subscription Discount | UI Not Navigate | Content Demand | Internet Issue Sentiment score
Step 3: Current solution: Overall churn rate prediction in last year review.
Step 4: Getting Data: We required exact date | Watch Time | Search Result Matching | Show Publish Date | Recommendation Click | Left Show in Middle Combine Discuss With - Data Engineering team & create a warehouse.
Step 5: Metrix to measure: Follow check points and rectify the direction of correct Metrix.
Step 6: Online Vs Batch ML: Focus on training model and retraining model. OLTP system in data warehouse Setup: OLTP, or online transactional processing, enables the real-time execution of large numbers of database transactions by large numbers of people, typically over the internet. A database transaction is a change, insertion, deletion, or query of data in a database.
Step 7: Check assumptions: Check one machine learning model work as all condition region and country.

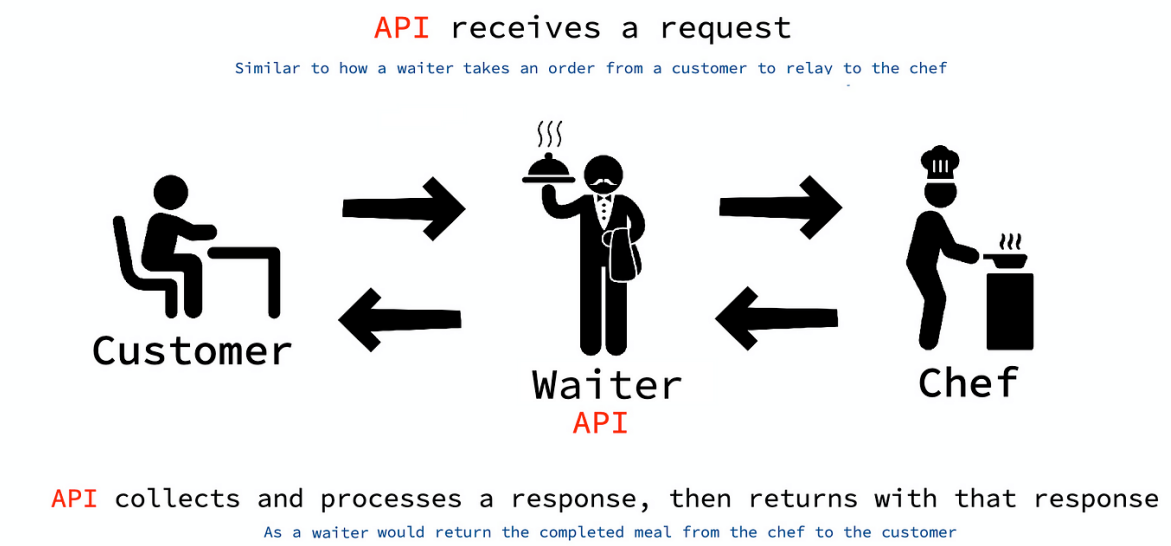
Industries Helping Hands

#Data Gathering

3. Fetching data from an API

What is the API and how it works?

API stands for “application programming interface.” An API is essentially a set of rules that dictate how two machines talk to each other. Some examples of API-based interactions include a cloud application communicating with a server, servers pinging each other, or applications interacting with an operating system.



Website we use for fetching API

TMDB: https://www.themoviedb.org/

TMDB API: https://developer.themoviedb.org/reference/movie-top-rated-list

Generate API Key after login and Paste in this link for JSON File

https://api.themoviedb.org/3/movie/top_rated?api_key=<<api_key>>&language=en-US&page=1

Use tool for understand the JSON File: https://jsonviewer.stack.hu/

Our objective: we want to create data set in python where row and column 8974 * 6

ID | Title | Realised Date | Overview | Popularity | Vote average

Python Code

Day 4 Machine Learning | iScale

```
import pandas as pd

import requests
```

```
 response =
requests.get('https://api.themoviedb.org/3/movie/top_rated?api_key=5bc643f664c8c66e7edce11b77c7214e&language=en-US&page=1')
```

```
 df =
pd.DataFrame(response.json()['results'])[['id','title','release_date','overview','popularity','vote_average']]
```

```
df.head()
```

```
 response =
requests.get('https://api.themoviedb.org/3/movie/top_rated?api_key=5bc643f664c8c66e7edce11b77c7214e&language=en-US&page=2')
```

```
df1 = pd.DataFrame(response.json()['results'])[['id','title','release_date','overview','popularity','vote_average']]
```

```
df1
```

```
df1.shape
```

```
df1.to_csv('movies.csv')
```

#Data Gathering

4. Fetching data using web scraping

Industries Helping Hands

#weclome | Day5

In [1]:
```
import pandas as pd
import requests
from bs4 import BeautifulSoup
```

In [4]:
```
headers={'User-Agent':'Mozilla/5.0 (Windows NT 6.3; Win 64 ; x64) Apple WeKit /537.36
webpage= requests.get('https://www.ambitionbox.com/list-of-companies?page=1',headers=h
```

#use "soup" for beautifulsoup

In [5]:
```
soup=BeautifulSoup(webpage,'lxml')
```

In [22]:
```
#print(soup.prettify())
```

Extract the heading line

In [33]:
```
for n in soup.find_all('h1'):
    print(n.text.strip())
```

```
List of companies in India
```

Extract the Company Name

In [34]:
```
for i in soup.find_all('h2'):
    print(i.text.strip())
```

```
TCS
Accenture
Cognizant
Wipro
HDFC Bank
ICICI Bank
Infosys
Capgemini
HCLTech
Tech Mahindra
Genpact
Axis Bank
Concentrix Corporation
Amazon
Teleperformance
Reliance Jio
IBM
Larsen & Toubro Limited
Reliance Retail
HDB Financial Services
Companies by  Industry
Companies by  Locations
Companies by  Type
Companies by  Badges
```

Extract Comapny Rating

In [40]:
```
for k in soup.find_all('span',class_='companyCardWrapper__companyRatingValue'):
      print(k.text.strip())
```

```
3.8
4.0
3.9
3.8
3.9
4.0
3.9
3.8
3.7
3.7
3.9
3.8
3.9
4.1
3.6
3.9
4.1
4.0
3.9
4.0
```

In [41]:
```
for k in soup.find_all('span'):
      print(k.text.strip())
```

```
+
You can choose to be
                                                        anonymous
anonymous

Companies
Reviews
Salaries
Interviews
All Filters









9,33,400
```

Extract Card Data

In [42]:
```
for j in soup.find_all('span',class_='companyCardWrapper__interLinking'):
      print(j.text.strip())
```

```
IT Services & Consulting | 1 Lakh+ Employees | Public | 55 years old | Mumbai +320 m
ore
IT Services & Consulting | 1 Lakh+ Employees | Public | 34 years old | Dublin +160 m
ore
IT Services & Consulting | 1 Lakh+ Employees | Forbes Global 2000 | 29 years old | T
eaneck. New Jersey. +138 more
IT Services & Consulting | 1 Lakh+ Employees | Public | 78 years old | Bangalore/Ben
galuru +267 more
Banking | 1 Lakh+ Employees | Public | 29 years old | Mumbai +1475 more
Banking | 1 Lakh+ Employees | Public | 29 years old | Mumbai +1233 more
IT Services & Consulting | 1 Lakh+ Employees | Public | 42 years old | Bengaluru/Ban
galore +158 more
IT Services & Consulting | 1 Lakh+ Employees | Public | 56 years old | Paris +110 mo
re
IT Services & Consulting | 1 Lakh+ Employees | Public | 32 years old | Noida +171 mo
re
IT Services & Consulting | 1 Lakh+ Employees | Public | 37 years old | Pune +251 mor
e
IT Services & Consulting | 1 Lakh+ Employees | Public | 26 years old | New York +96
more
Banking | 50k-1 Lakh Employees | Public | 30 years old | Mumbai +1257 more
BPO | 50k-1 Lakh Employees | Public | 40 years old | Fremont +102 more
Internet | 1 Lakh+ Employees | Public | 29 years old | Seattle +416 more
BPO | 50k-1 Lakh Employees | Forbes Global 2000 | 45 years old | Paris +160 more
Telecom | 50k-1 Lakh Employees | Public | 16 years old | Navi Mumbai +1116 more
Software Product | 1 Lakh+ Employees | Public | 112 years old | Armonk +140 more
Engineering & Construction | 10k-50k Employees | Public | 85 years old | Mumbai +610
more
Retail | 1 Lakh+ Employees | 17 years old | Navi Mumbai +942 more
NBFC | 1 Lakh+ Employees | 16 years old | Ahmedabad +914 more
```

In [ ]:

Data Science | 30 Days of Machine Learning | Day - 5

Educator Name: Nishant Dhote
Support Team: **+91-7880-113-112**

#Data Gathering (Previous Classes)

1. **Working with CSV Files – Day 3**
2. **Working with JSON/SQL – Day 3**

3. **Fetching data from an API – Day 4**

#Framing a Machine Learning Problem – Day 4

---------- Today Topics | Day 5-------

#Data Gathering

4. **Fetching data using web scraping**

----Upcoming Topics----

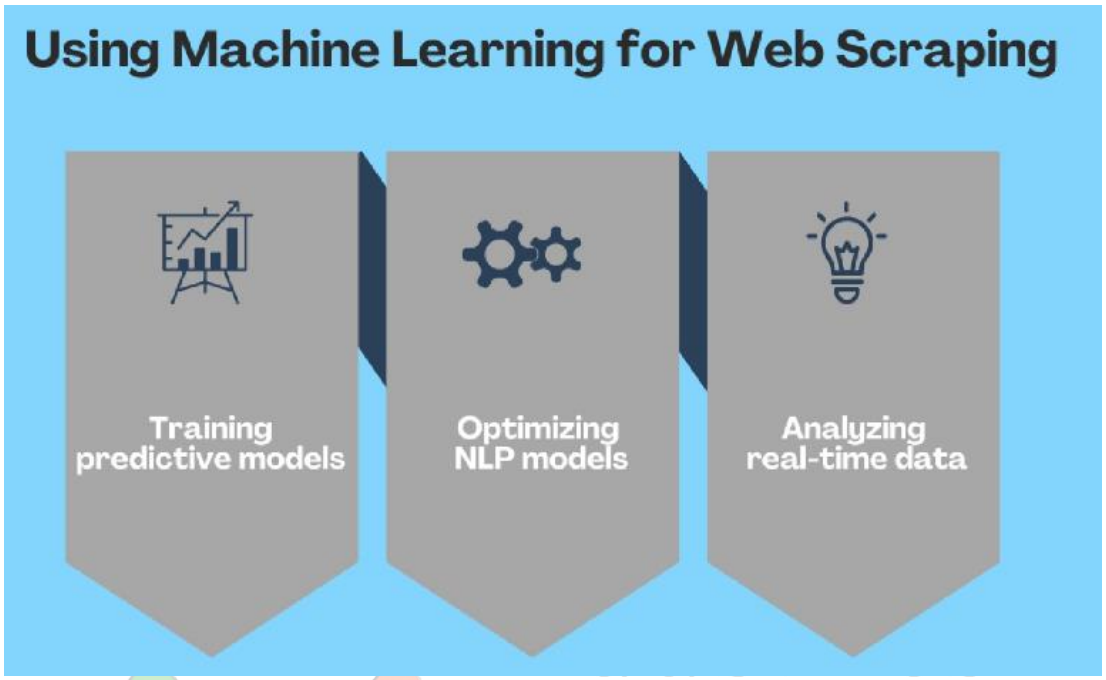
-EDA & Feature Engineering

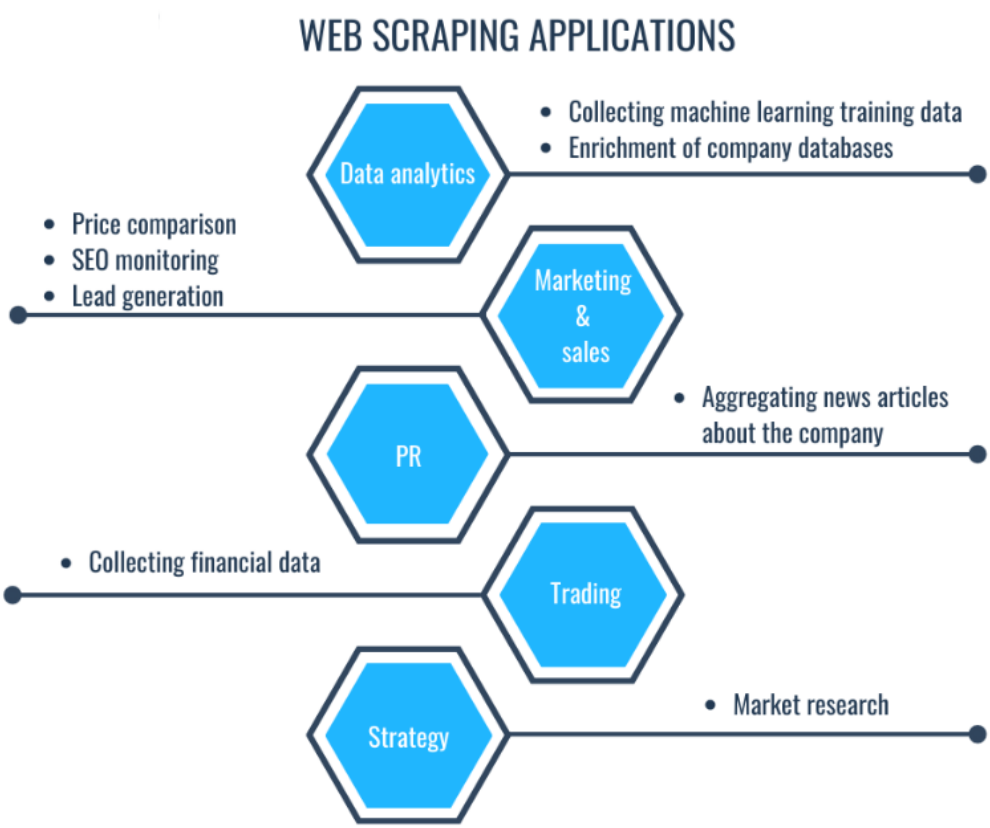
 # How We Understand the Data?

 # What is Web Scraping: It's is a term used for automatically retrieving data from the internet and structuring it in a useful manner. The standard scraping algorithm makes use of static paths to navigate the program through the HTML to the sought-after data.

Web scraping is an automatic method to obtain large amounts of data from websites. Most of this data is unstructured data in an HTML format which is then converted into structured data in a spreadsheet or a database so that it can be used in various applications.

Read Blog: https://research.aimultiple.com/machine-learning-web-scraping/

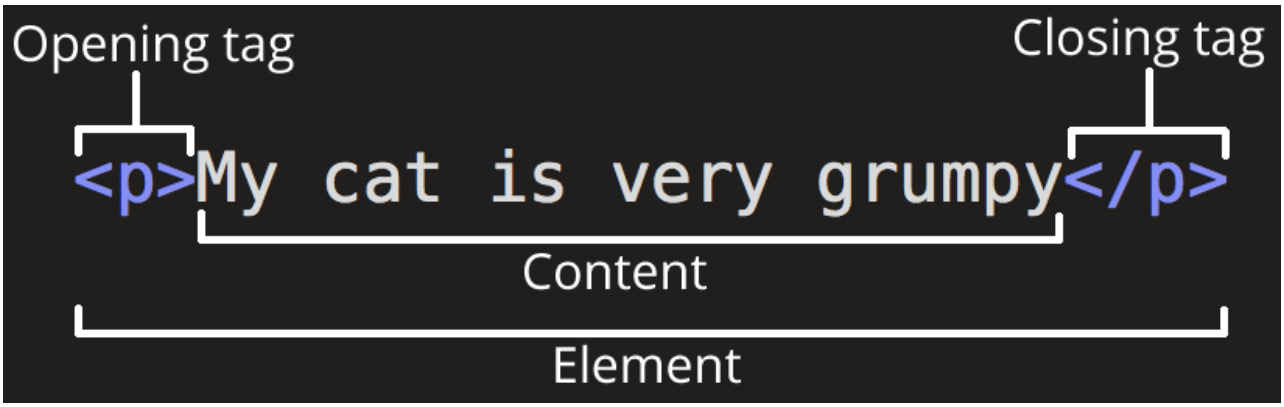


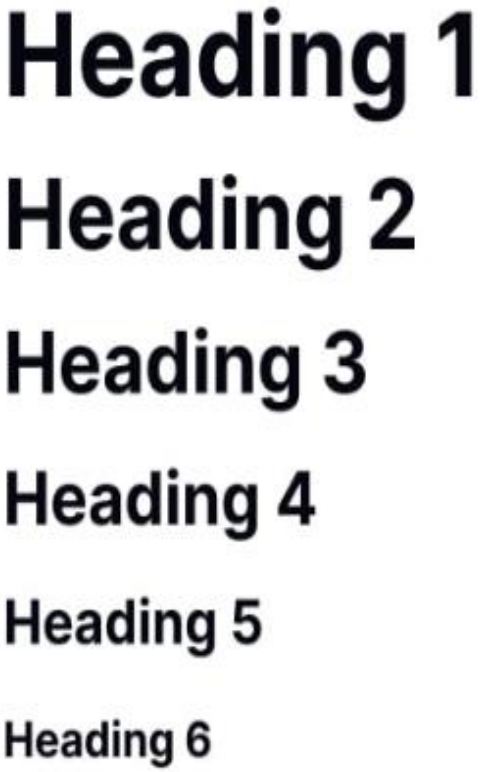


HTML Hierarchy

```
<!DOCTYPE html>
<html>
    <head>
        <title>Title of the Page</title>
    </head>
    <body>
    Tags related to layout and formatting
    </body>
</html>
```

- `<!DOCTYPE html>` ← Tells the document type
- `<html>` ← The Root Element
- `<head>` ← Contains the header information
- `<title>Title of the Page</title>` ← Defines Title of the Page
- `<body>` ← Holds the Content of the Page

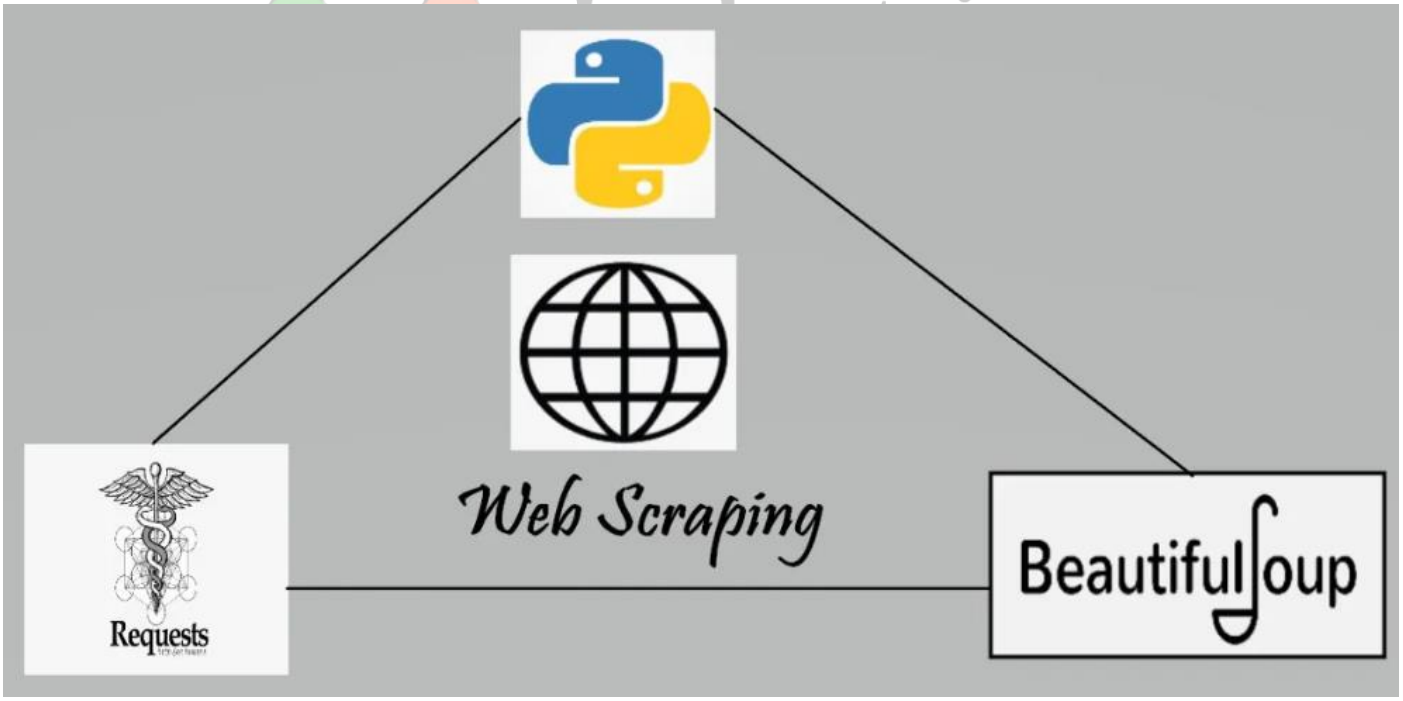





```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

```
<body>

  <div id="content">
    <h1>Heading here</h1>
    <p>Lorem ipsum dolor sit amet.</p>
    <p>Lorem ipsum dolor <em>sit</em> amet.</p>
    <hr>
  </div>

  <div id="nav">
    <ul>
      <li>item 1</li>
      <li>item 2</li>
      <li>item 3</li>
    </ul>
  </div>

</body>
```

```
<div class="container-fluid">
  <h3 class="text-primary text-center">jQuery Playground</h3>
  <div class="row">
    <div class="col-xs-6">
      <div class="well">
        <button class="btn btn-info">Info</button>
        <button class="btn btn-primary">Like</button>
        <button class="btn btn-danger">Delete</button>
      </div>
    </div>
    <div class="col-xs-6">
      <div class="well">
      <div class="btn btn-info">More</div>
      <div class="btn btn-primary">Click</div>
      <div class="btn btn-danger">Report</div>
      </div>
    </div>
  </div>
</div>
```



<Open Jupiter Notebook>

Welcome to 30 Days Machine Learning

Import Libraries

```
import pandas as pd
import requests
from bs4 import BeautifulSoup
```

We use link for web scraping: https://www.ambitionbox.com/list-of-companies?page=1

Import Webpage

```
headers={'User-Agent':'Mozilla/5.0 (Windows NT 6.3; Win 64 ; x64) Apple WeKit
/537.36(KHTML , like Gecko) Chrome/80.0.3987.162 Safari/537.36'}

webpage=requests.get('https://www.ambitionbox.com/list-of-
companies?page=1',headers=headers).text
```

Define “Soup” as BeautifulSoup Library

```
soup=BeautifulSoup(webpage,'lxml')
```

Use Print

```
#print(soup.prettify())
```

Extract the heading line

```
for n in soup.find_all('h1'):
   print(n.text.strip())
```

Extract the Company Name

```
for i in soup.find_all('h2'):
  print(i.text.strip())
```

Extract the rating value

```
for k in soup.find_all('span',class_='companyCardWrapper__companyRatingValue'):
    print(k.text.strip())
```

Extract the all card details

```
for j in soup.find_all('span',class_='companyCardWrapper__interLinking'):
    print(j.text.strip())
```

<End Code>

Upcoming Class- Day 6

- **Exploratory Data Analysis (EDA)**

- **EDA Univariate Analysis**
- **EDA Bivariate Analysis**
- **EDA Multivariate Analysis**

- **Pandas Profiler**
- **Asking Basic Question**

Industries Helping Hands

Data Story Telling : Curious Data Minds

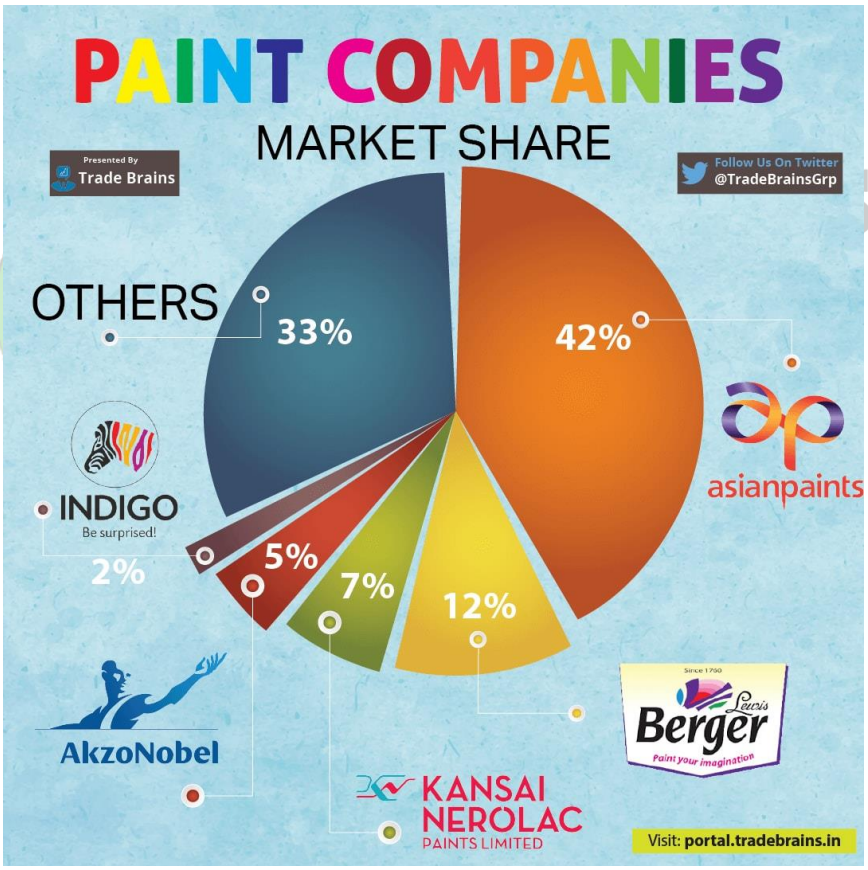
ASIAN PAINTS: INDIA'S BIGGEST DATA SCIENCE COMPANY THAT SELLS PAINT

Read Blogs:
https://startuptalky.com/asian-paints-case-study/

https://d3.harvard.edu/platform-digit/submission/asian-paints-indias-biggest-data-science-company-that-sells-paint/

https://yourstory.com/2023/08/asian-paints-data-science-revolution

https://tradebrains.in/asian-paints-case-study/



Data Science | 30 Days of Machine Learning | Day - 6

Educator Name: Nishant Dhote
Support Team: **+91-7880-113-112**

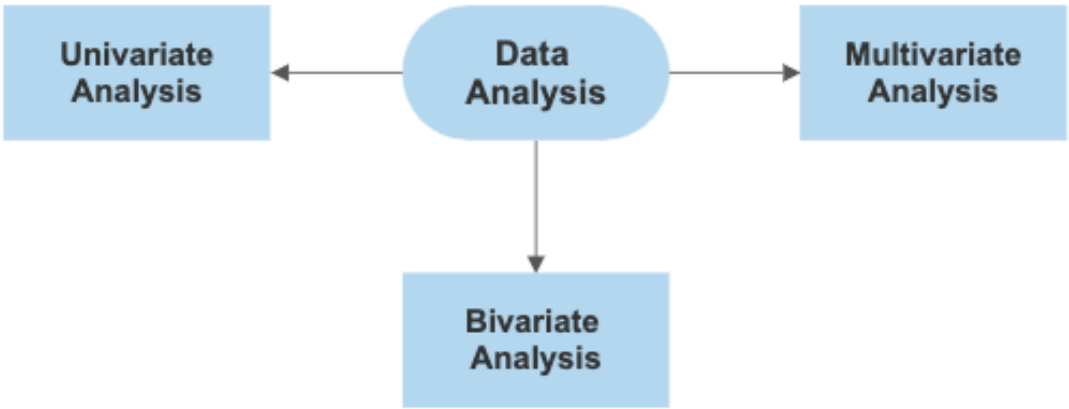
----Today Topics | Day 06----

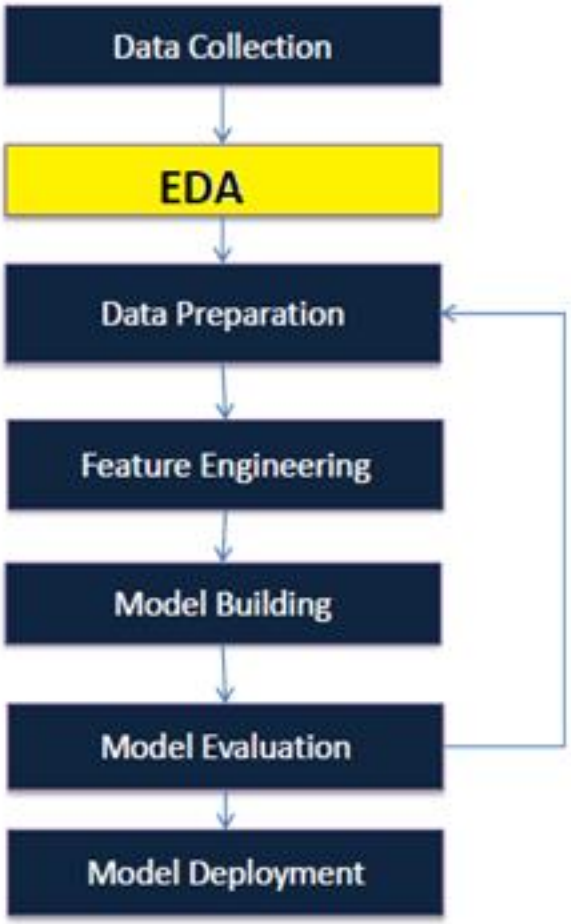
- **EDA: Exploratory Data Analysis**

- **EDA Univariate Analysis**
- **EDA Bivariate Analysis**
- **EDA Multivariate Analysis**
- **# How We Understand the Data?**
- **Feature Engineering**

Exploratory data analysis (EDA) is an approach to analysing data sets to summarize their main characteristics, often with visual methods. Before applying any ML algorithms in data, we need to understand the data which we are going to follow. Without data understanding there will be a possibility of ML model failure. The understanding of data is nothing but this Exploratory Data Analysis (EDA).

It is always better to explore each data set using multiple exploratory techniques and compare the results. Once the data set is fully understood, it is quite possible that data scientist will have to go back to data collection and cleansing phases in order to transform the data set according to the desired business outcomes. The goal of this step is to become confident that the data set is ready to be used in a machine learning algorithm.





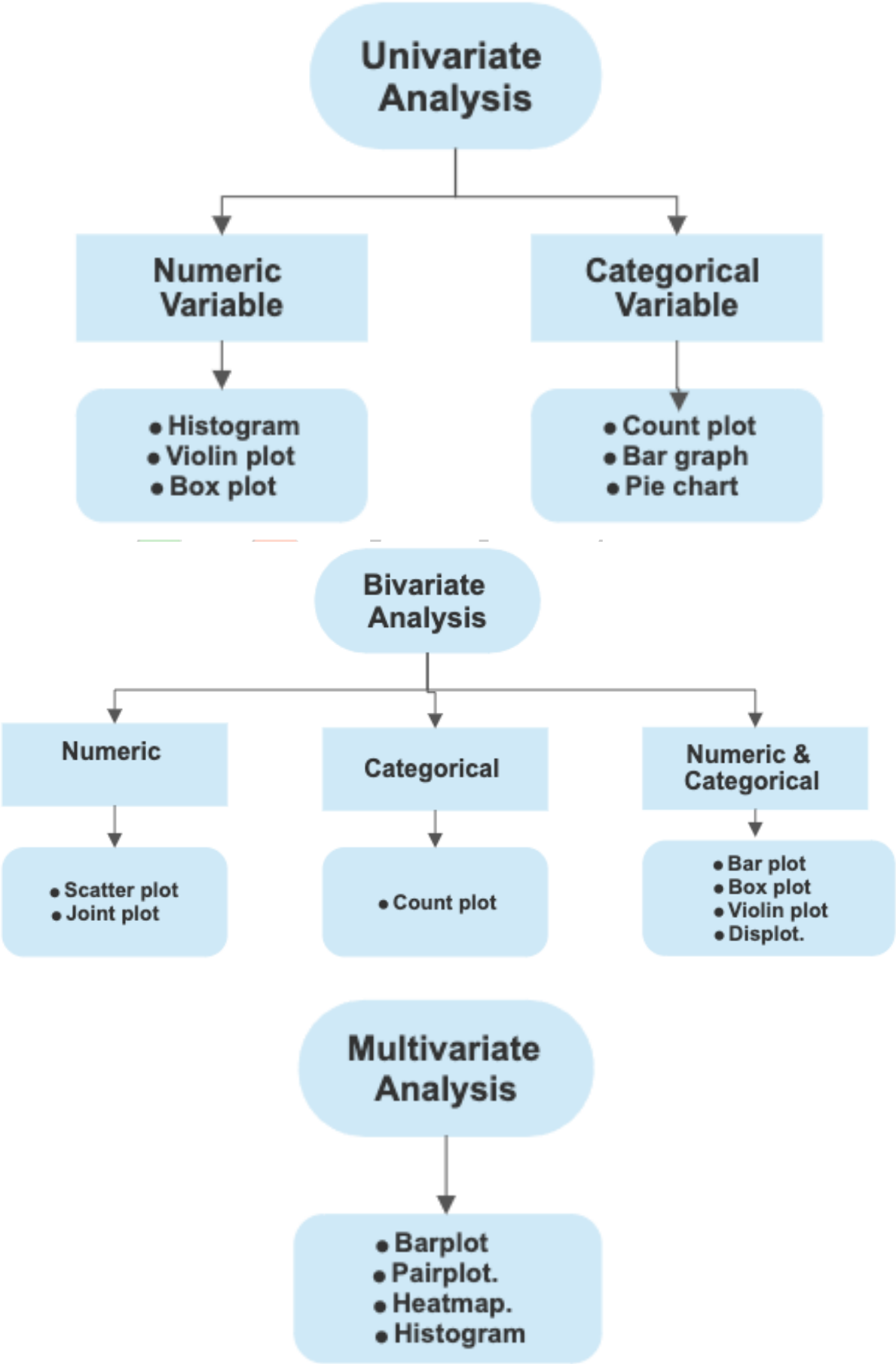
TYPES OF EXPLORATORY DATA ANALYSIS:

1. Univariate Analysis
2. Bivariate Analysis
3. Multivariate Analysis

Univariate EDA involves looking at a single variable at a time. Univariate EDA can help you understand the data distribution and identify any outliers.

Bivariate EDA involves looking at two variables at a time. Bivariate EDA can help you understand the relationship between two variables and identify any patterns that might exist.

Multivariate EDA involves looking at three or more variables at a time. Multivariate EDA can help you understand the relationships between several variables and identify any complex patterns or outliers that might exist.


Univariate Analysis
Numeric Variable
Categorical Variable
• Histogram
• Violin plot
• Box plot
• Count plot
• Bar graph
• Pie chart
Bivariate Analysis
Numeric
Categorical
Numeric & Categorical
• Scatter plot
• Joint plot
• Count plot
• Bar plot
• Box plot
• Violin plot
• Displot.
Multivariate Analysis
• Barplot
• Pairplot.
• Heatmap.
• Histogram

Dataset Link Kaggle: https://www.kaggle.com/competitions/titanic

GitHub Link:
https://github.com/TheiScale/30_Days_Machine_Learning/tree/main/Day%206%20ML

#Import Library and Dataset

```
import pandas as pd
df = pd.read_csv('train.csv')
```

1. **Data Size:**
   - What is the magnitude of the dataset in terms of its rows and columns?
   - How big is the data?

```
df.shape
```

2. **Data Visualization:**
   - Can you provide a visual representation or description of the dataset's structure?
   - How does the data look like?

```
df.sample(5)
```

3. **Column Data Varieties:**
   - What variations in data types exist among the columns in the dataset?
   - What is the data types of cols?

```
df.info()
```

4. **Existence of Missing Values:**
   - Are there any absent values within the dataset? If so, which columns have them, and how many are missing?
   - Are there any missing values?

```
df.isnull().sum()
```

5. **Mathematical Representation:**
   - How is the data characterized mathematically, including measures such as mean, median, and standard deviation?

- How does the data look mathematically?

```
df.describe()
```

6. **Identification of Redundancies:**
   - Is there any repetition or duplication of values within the dataset?
   - Are there duplicate values?

```
df.duplicated().sum()
```

7. **Correlation Examination:**
   - What level of correlation exists between different columns in the dataset?
   - How is the correlation between columns?

```
df.corr(numeric_only = True)
```

1. EDA in Univariate Analysis:

```
<Start Coding>
```

#Import Library

```
import pandas as pd
import seaborn as sns
```

#Define Data Frame as "df"

```
df = pd.read_csv('train.csv')

---
df.head()
```

#1. Categorical Data
#a. Countplot

```
sns.countplot(x='Survived', data=df)
df['Survived'].value_counts()

OR--

sns.countplot(x='Pclass', data=df)
df['Pclass'].value_counts()
```

OR--

```
sns.countplot(x='Embarked', data=df)
df['Embarked'].value_counts()
```

#b. Piechart

```
df['Sex'].value_counts().plot(kind='pie',autopct='%.2f')
```

2. NUMERICAL DATA

a. Histogram

```
import matplotlib.pyplot as plt
plt.hist(df['Age'],bins=5)
```

b. Distplot / Histplot

```
sns.distplot(df['Age'])
```

c. Boxplot

Article 1: click here to read more boxplot or Article 2 click here

```
sns.boxplot(df['Age'])
```

df['Age'].min()

df['Age'].max()

df['Age'].mean()

df['Age'].skew()

Recommended Datasets for practice:

Air Passengers- https://www.kaggle.com/code/chandrimad31/flight-passenger-satisfaction-eda-and-prediction

2.EDA Bivariate Analysis <Day 7>

3.EDA Multivariate Analysis <Day 7>

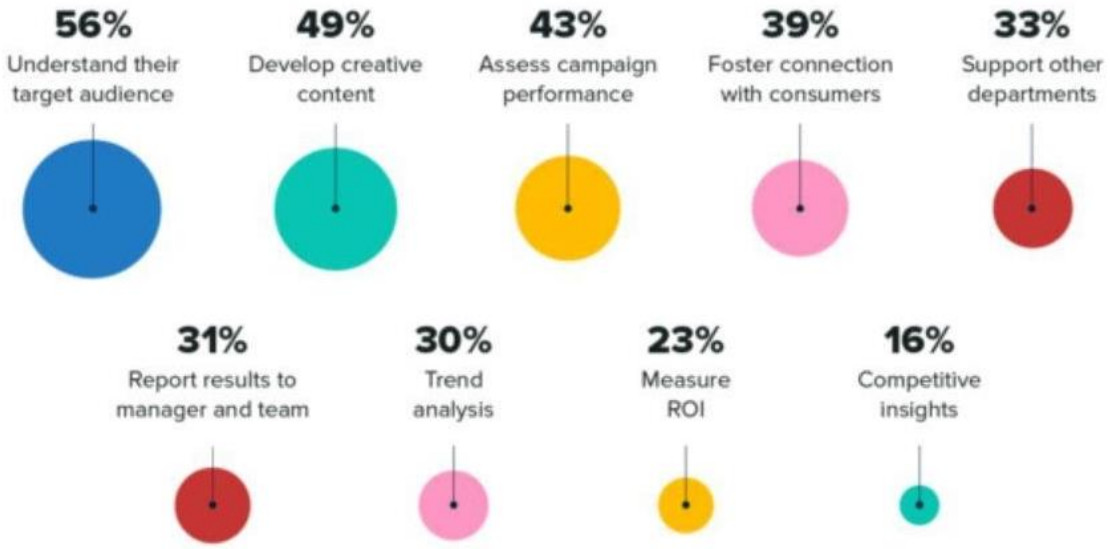
Data Story Telling (Day 6): Curious Data Minds

How is data science used in media and entertainment industry?

Read Blogs: https://hevodata.com/learn/data-analytics-in-media/#:~:text=Here%20are%20a%20couple%20of,in%20listening%20or%20viewing%20habits.

https://www.polestarllp.com/blog/big-data-analytics-media-and-the-entertainment-industry

How We use Social Data



Welcome to ML | Day 06

How We Understand the Data? ¶

Import Library and Dataset

In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv('train.csv')
```

In [3]:
```
df
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	E
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	

891 rows × 12 columns

Data Size:

In [5]:
```
df.shape
```

Out[5]: (891, 12)

In [7]:
```
df.head()
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emb
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	

Data Visualization:

In [10]:
```
df.sample(5)
```

Out[10]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emba
461	462	0	3	Morley, Mr. William	male	34.0	0	0	364506	8.0500	NaN	
826	827	0	3	Lam, Mr. Len	male	NaN	0	0	1601	56.4958	NaN	
848	849	0	2	Harper, Rev. John	male	28.0	0	1	248727	33.0000	NaN	
874	875	1	2	Abelson, Mrs. Samuel (Hannah Wizosky)	female	28.0	1	0	P/PP 3381	24.0000	NaN	
678	679	0	3	Goodwin, Mrs. Frederick (Augusta Tyler)	female	43.0	1	6	CA 2144	46.9000	NaN	

Column Data Varieties:

In [12]:
```
df.info()
```

```
<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
```

Existence of Missing Values:

In [14]:
```
df.isnull().sum()
```

Out[14]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

Mathematical Representation:

In [15]:
```
df.describe()
```

Out[15]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Correlation Examination:

In [16]:
```
df.corr(numeric_only = True)
```

Out[16]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

In [ ]:

EDA in Univariate Analysis

Import Library

import pandas as pd

In [7]:
```
import pandas as pd
import seaborn as sns
```

In [8]:
```
df = pd.read_csv('train.csv')
```

In [10]:
```
df.sample(5)
```

Out[10]:

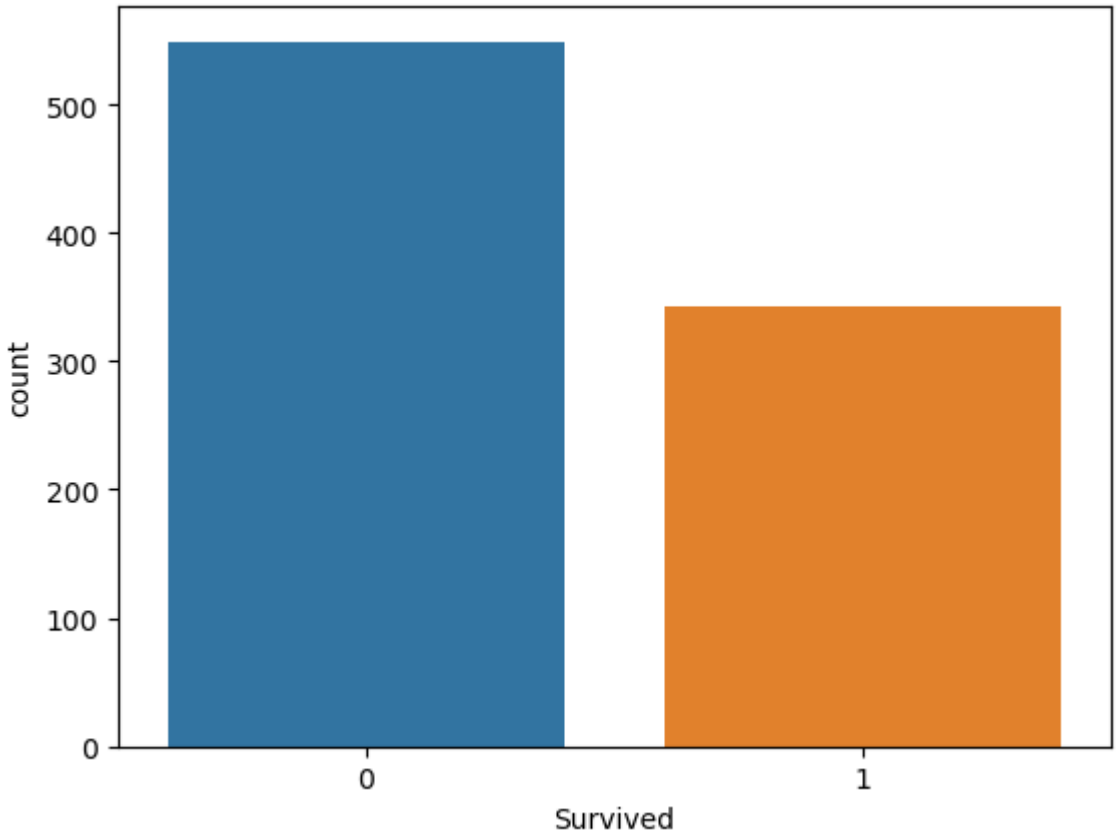
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	E
628	629	0	3	Bostandyeff, Mr. Guentcho	male	26.0	0	0	349224	7.8958	NaN	
847	848	0	3	Markoff, Mr. Marin	male	35.0	0	0	349213	7.8958	NaN	
204	205	1	3	Cohen, Mr. Gurshon "Gus"	male	18.0	0	0	A/5 3540	8.0500	NaN	
146	147	1	3	Andersson, Mr. August Edvard ("Wennerstrom")	male	27.0	0	0	350043	7.7958	NaN	
655	656	0	2	Hickman, Mr. Leonard Mark	male	24.0	2	0	S.O.C. 14879	73.5000	NaN	

1. Categorical Data

a. Countplot

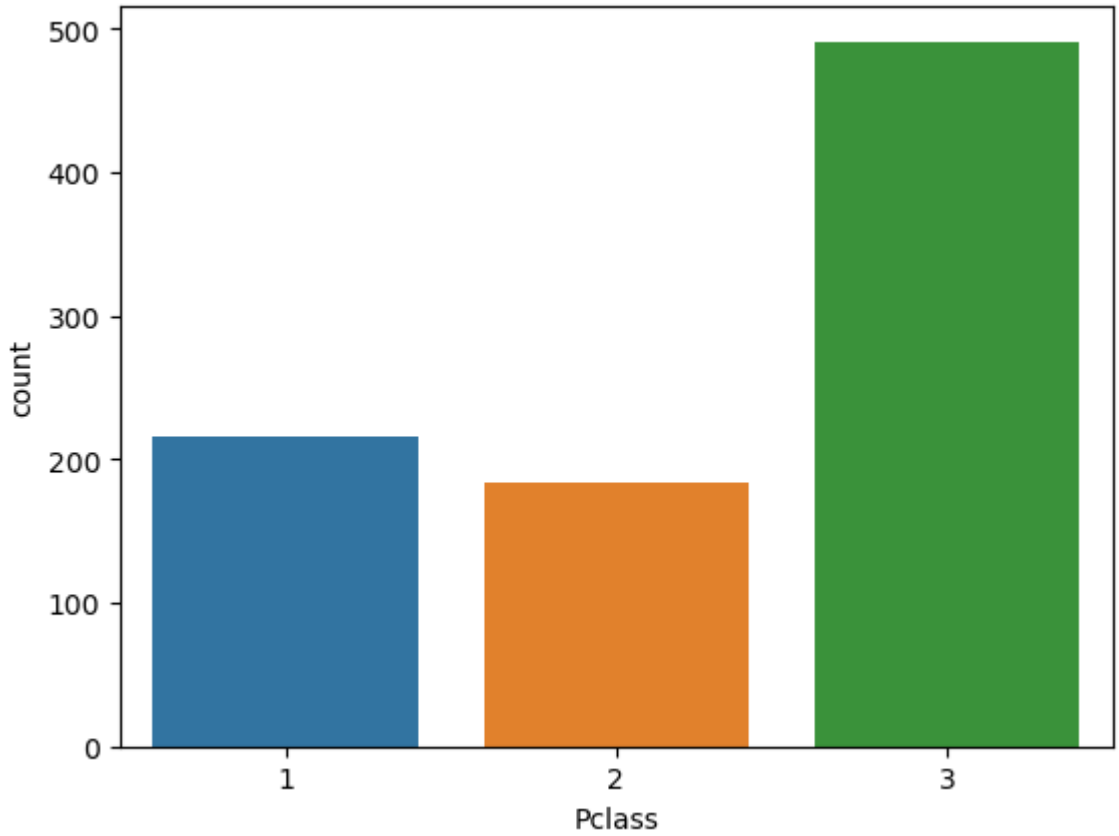
In [12]:
```
sns.countplot(x='Survived', data=df)
df['Survived'].value_counts()
```

Out[12]:
```
Survived
0    549
1    342
Name: count, dtype: int64
```



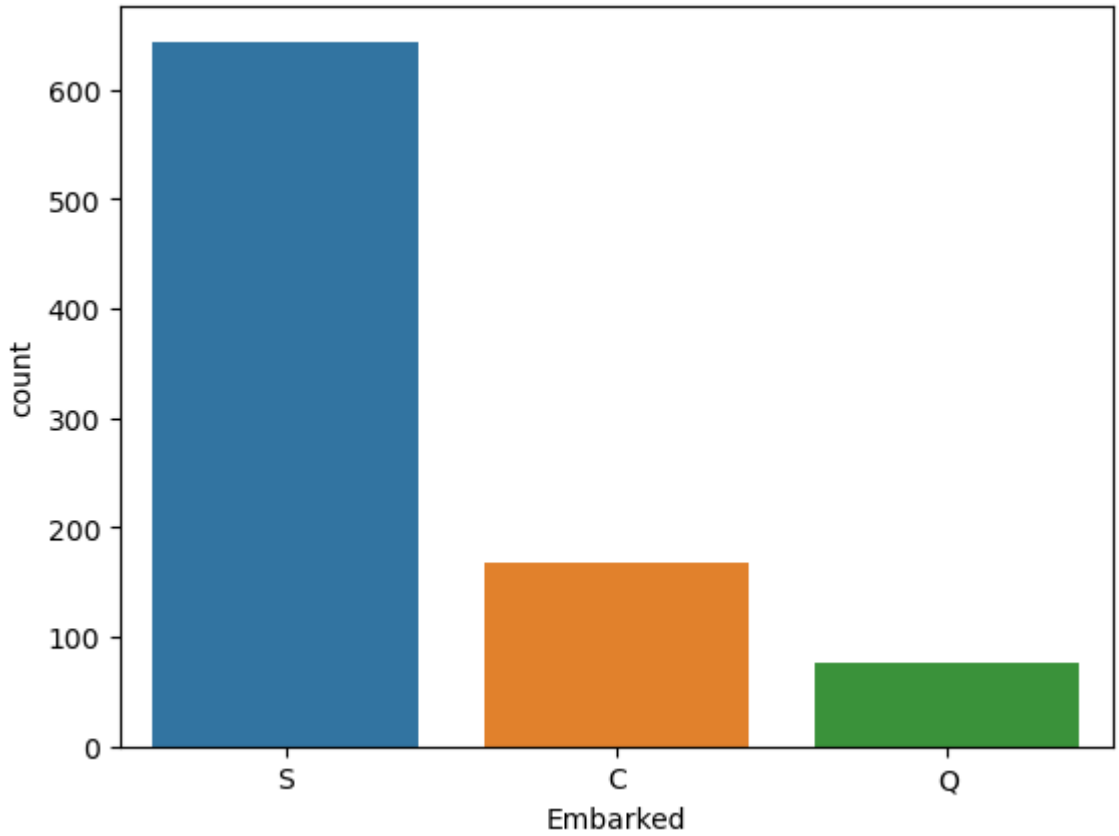
In [13]:
```
sns.countplot(x='Pclass', data=df)
df['Pclass'].value_counts()
```

Out[13]:
```
Pclass
3    491
1    216
2    184
Name: count, dtype: int64
```



In [14]:
```
sns.countplot(x='Embarked', data=df)
df['Embarked'].value_counts()
```

Out[14]:
```
Embarked
S    644
C    168
Q     77
Name: count, dtype: int64
```

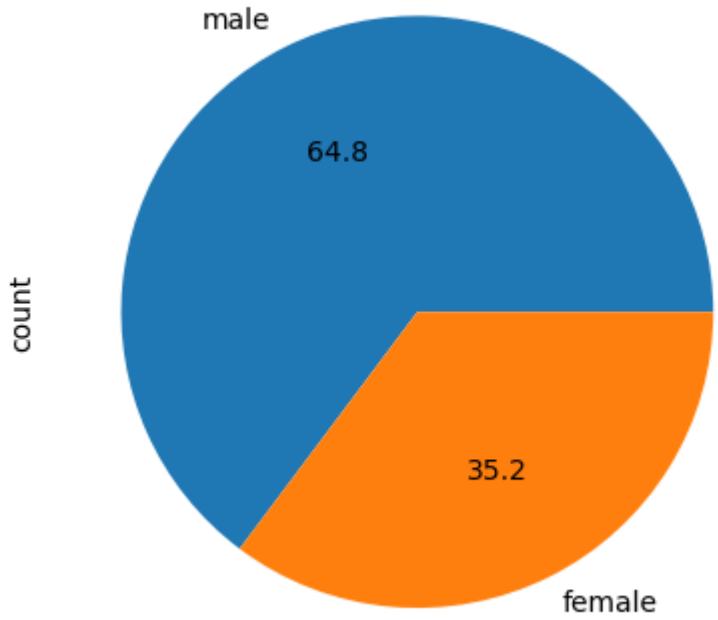


Type *Markdown* and LaTeX: α^2

b. Piechart

In [19]:
```
df['Sex'].value_counts().plot(kind='pie',autopct='%.1f')
```

Out[19]:
```
<Axes: ylabel='count'>
```



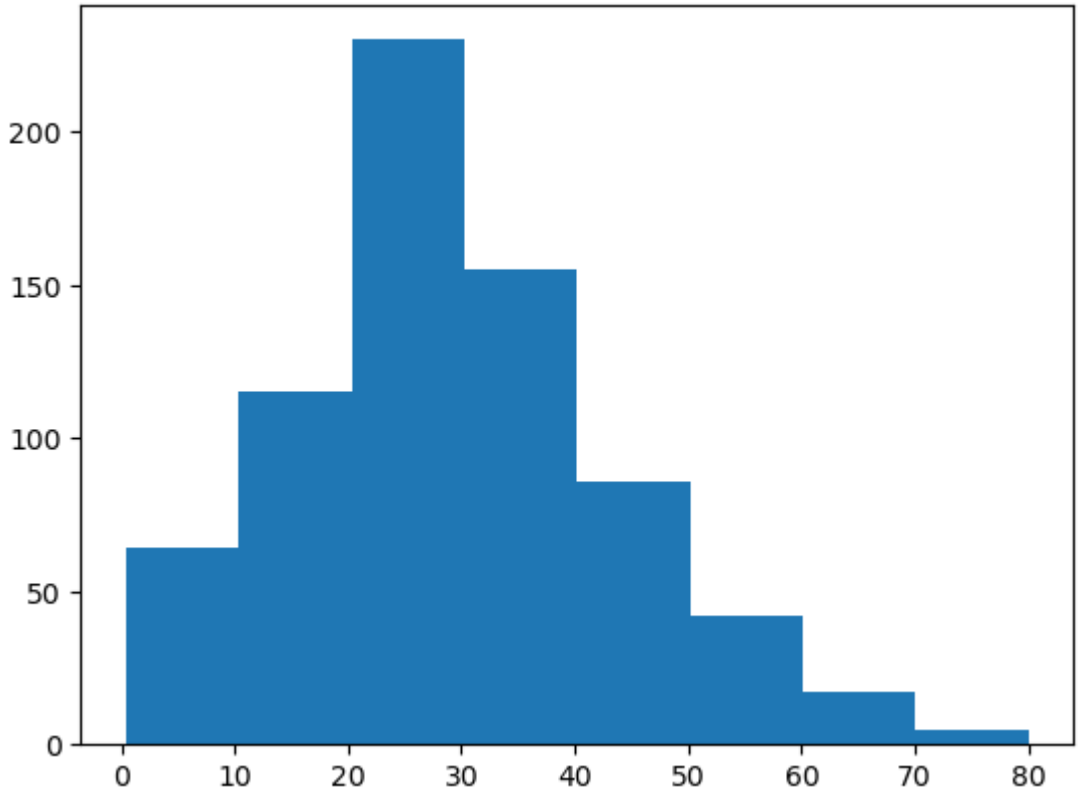
2. NUMERICAL DATA

a. Histogram

In [20]:
```
import matplotlib.pyplot as plt
```

In [29]:
```
plt.hist(df['Age'],bins=8)
```

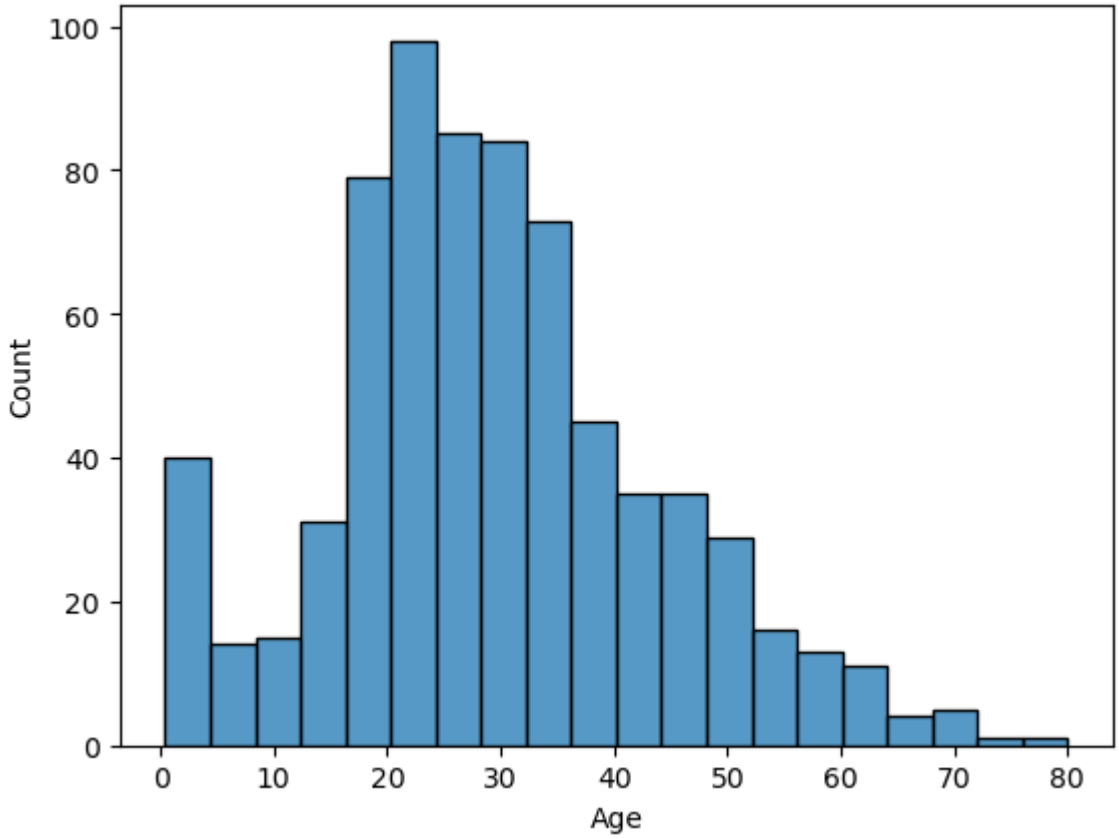
Out[29]:
```
(array([ 64., 115., 230., 155.,  86.,  42.,  17.,   5.]),
 array([ 0.42  , 10.3675, 20.315 , 30.2625, 40.21  , 50.1575, 60.105 ,
        70.0525, 80.    ]),
 <BarContainer object of 8 artists>)
```



b. Distplot / Histplot

In [33]:
```
sns.histplot(df['Age'])
```

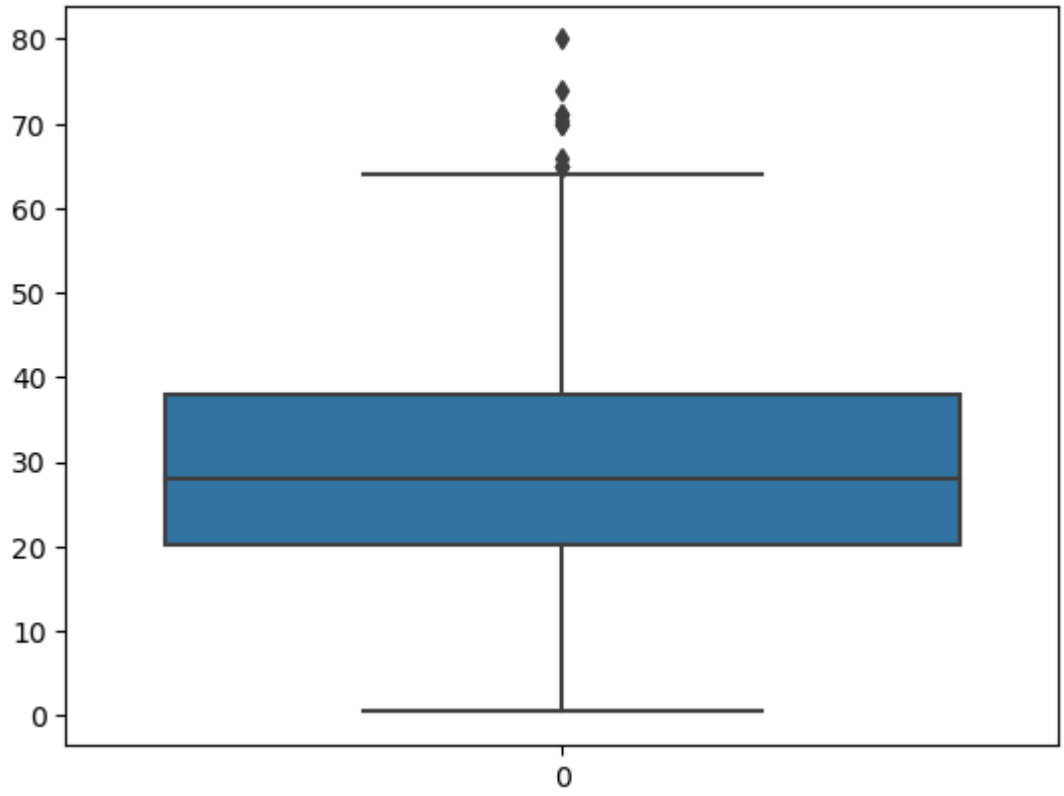
Out[33]:
```
<Axes: xlabel='Age', ylabel='Count'>
```



c. Boxplot

In [34]:
```
sns.boxplot(df['Age'])
```

Out[34]: <Axes: >



In [35]:
```
df['Age'].min()
```

Out[35]: 0.42

In [36]:
```
df['Age'].max()
```

Out[36]: 80.0

In [37]:
```
df['Age'].mean()
```

Out[37]: 29.69911764705882

In [38]:
```
df['Age'].skew()
```

Out[38]: 0.38910778230082704

In [ ]:

Welcome to 30 Day ML | Day 7

Import Library

In [1]:
```
import pandas as pd
import seaborn as sns
```

Import Datasets 1: Titanic

In [2]:
```
titanic = pd.read_csv('train.csv')
```

In [4]:
```
titanic.head(3)
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emb
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	

Use Datasets 2: Hotel Bill & Tips

In [18]:
```
bill = sns.load_dataset('tips')
```

In [19]:
```
bill
```

Out[19]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

Use Datasets 3: USA Flights

In [8]:
```
flights = sns.load_dataset('flights')
```

In [10]:
```
flights
```

Out[10]:

	year	month	passengers
0	1949	Jan	112
1	1949	Feb	118
2	1949	Mar	132
3	1949	Apr	129
4	1949	May	121
...	...	...	...
139	1960	Aug	606
140	1960	Sep	508
141	1960	Oct	461
142	1960	Nov	390
143	1960	Dec	432

144 rows × 3 columns

Use Datasets 4: Iris Flower

In [11]:
```
irisflwr = sns.load_dataset('iris')
```

In [12]:
```
irisflwr
```

Out[12]:

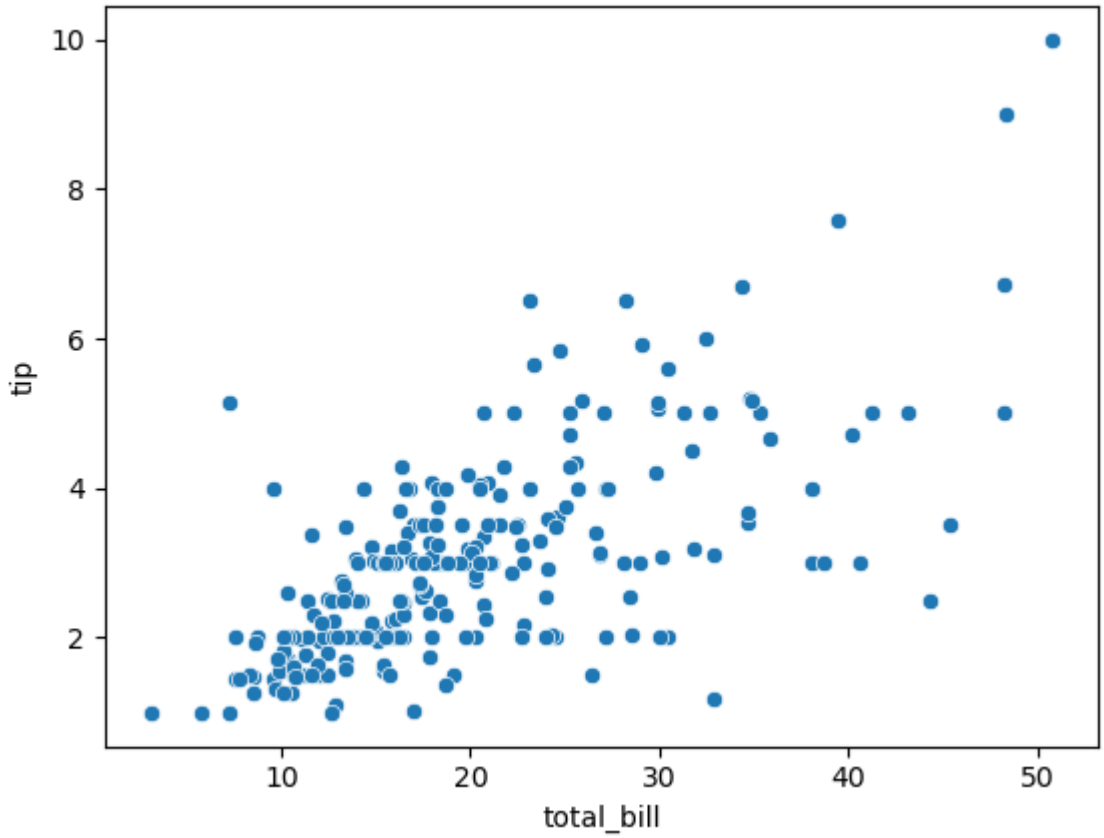
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

1<Numerical – Numerical> | Scatterplot

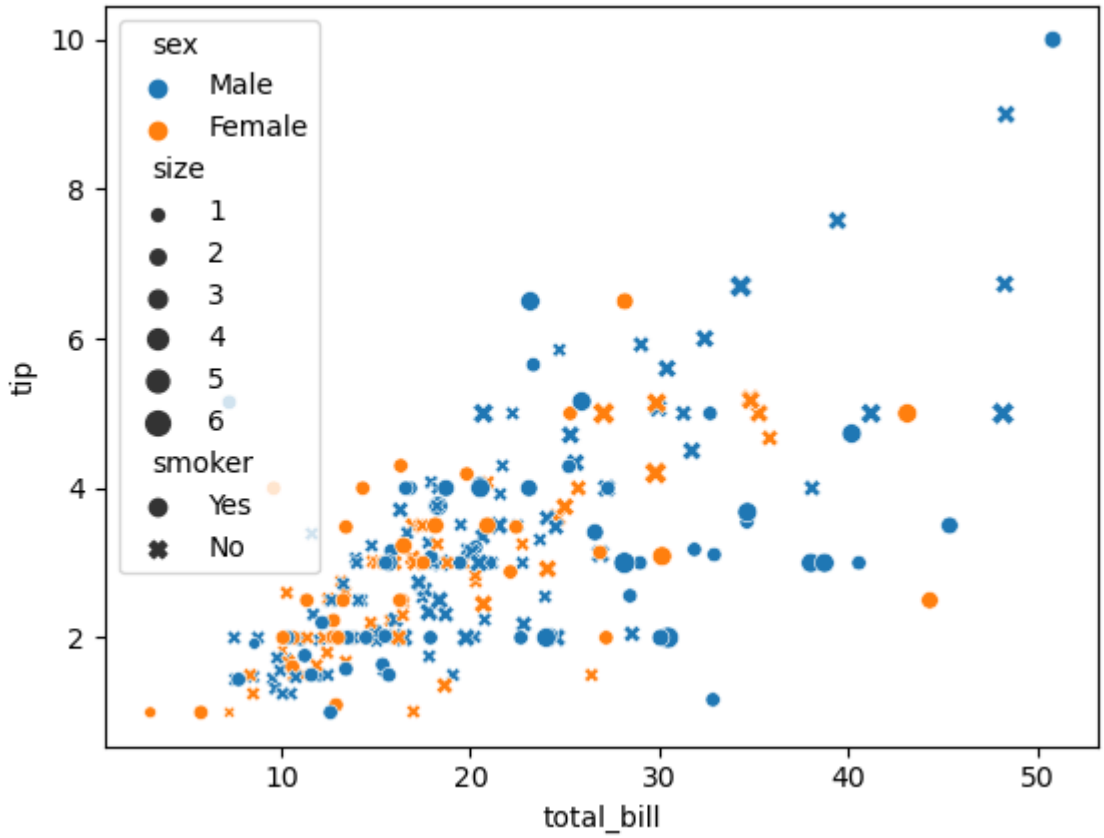
In [20]:
```
sns.scatterplot(x=bill['total_bill'],y=bill['tip'])
```

Out[20]:
```
<Axes: xlabel='total_bill', ylabel='tip'>
```



In [26]:
```
sns.scatterplot(x=bill['total_bill'],y=bill['tip'],hue=bill['sex'],style=bill['smoker
```

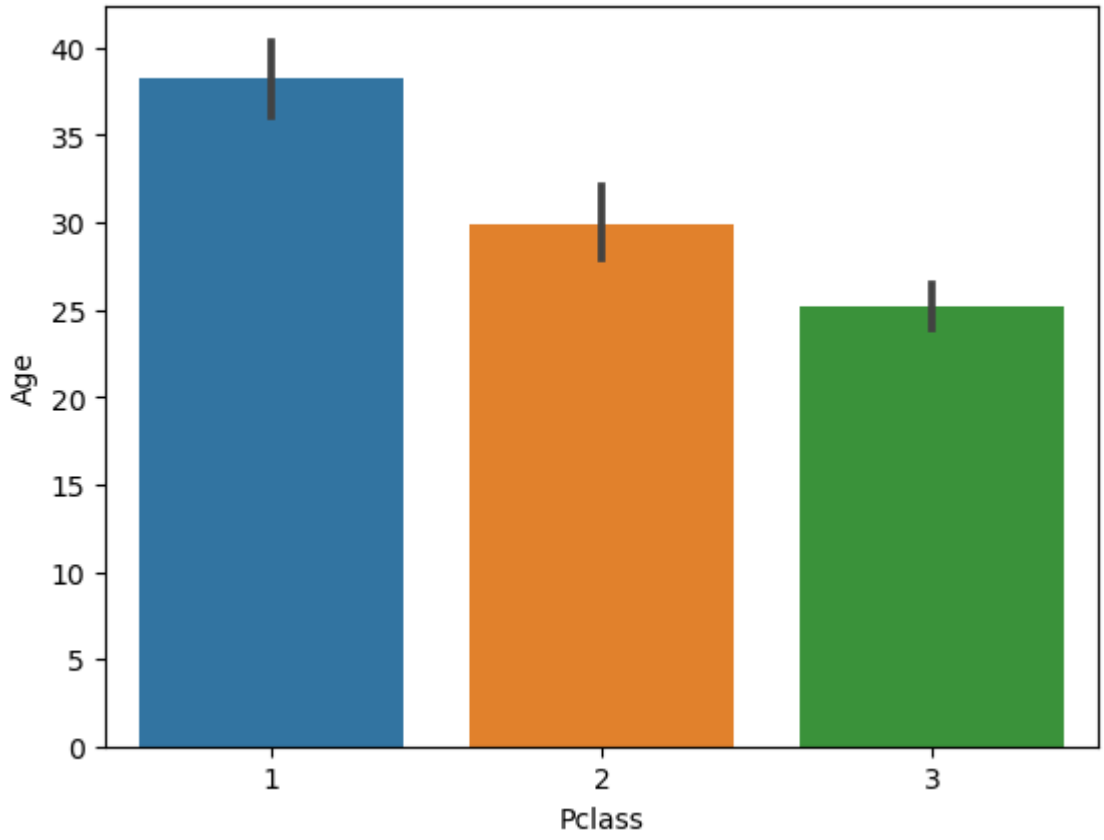
Out[26]:
```
<Axes: xlabel='total_bill', ylabel='tip'>
```



2<Numerical –Categorical> | Bar plot

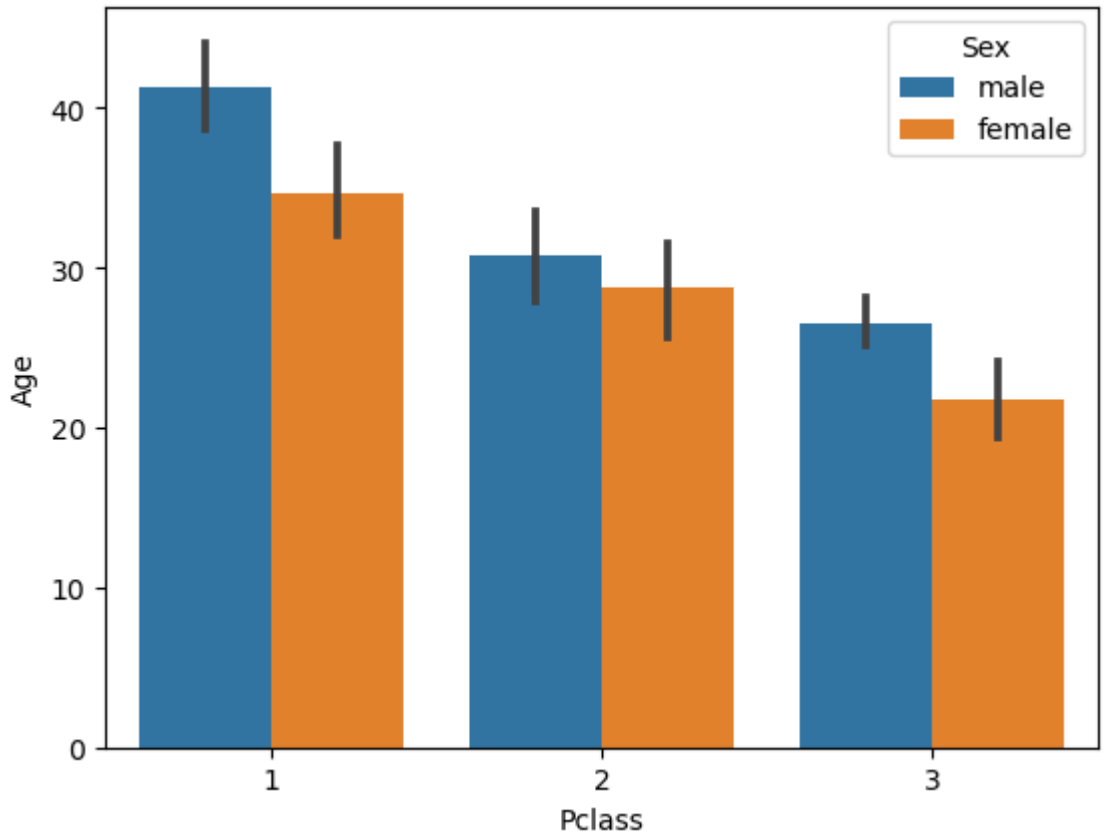
In [28]:
```
sns.barplot(x=titanic['Pclass'],y=titanic['Age'])
```

Out[28]: <Axes: xlabel='Pclass', ylabel='Age'>



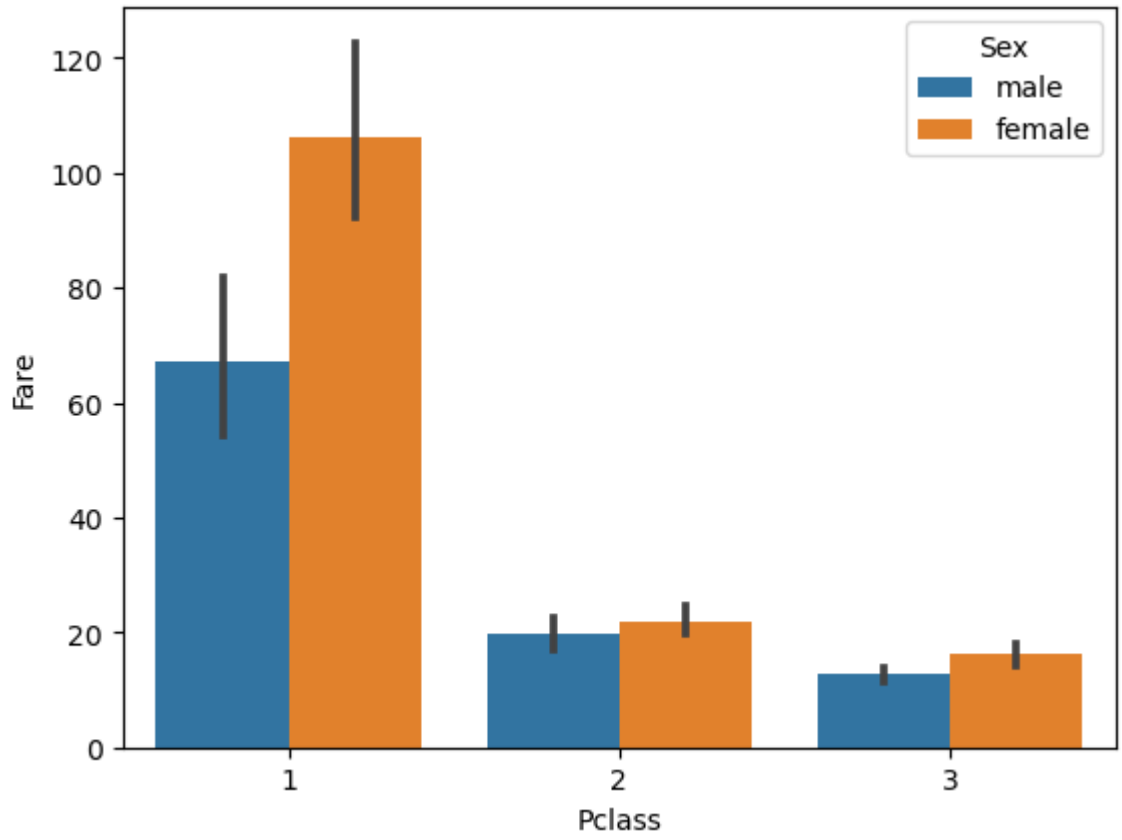
In [29]:
```
sns.barplot(x=titanic['Pclass'],y=titanic['Age'],hue=titanic['Sex'])
```

Out[29]: <Axes: xlabel='Pclass', ylabel='Age'>



In [30]:
```
sns.barplot(x=titanic['Pclass'],y=titanic['Fare'],hue=titanic['Sex'])
```

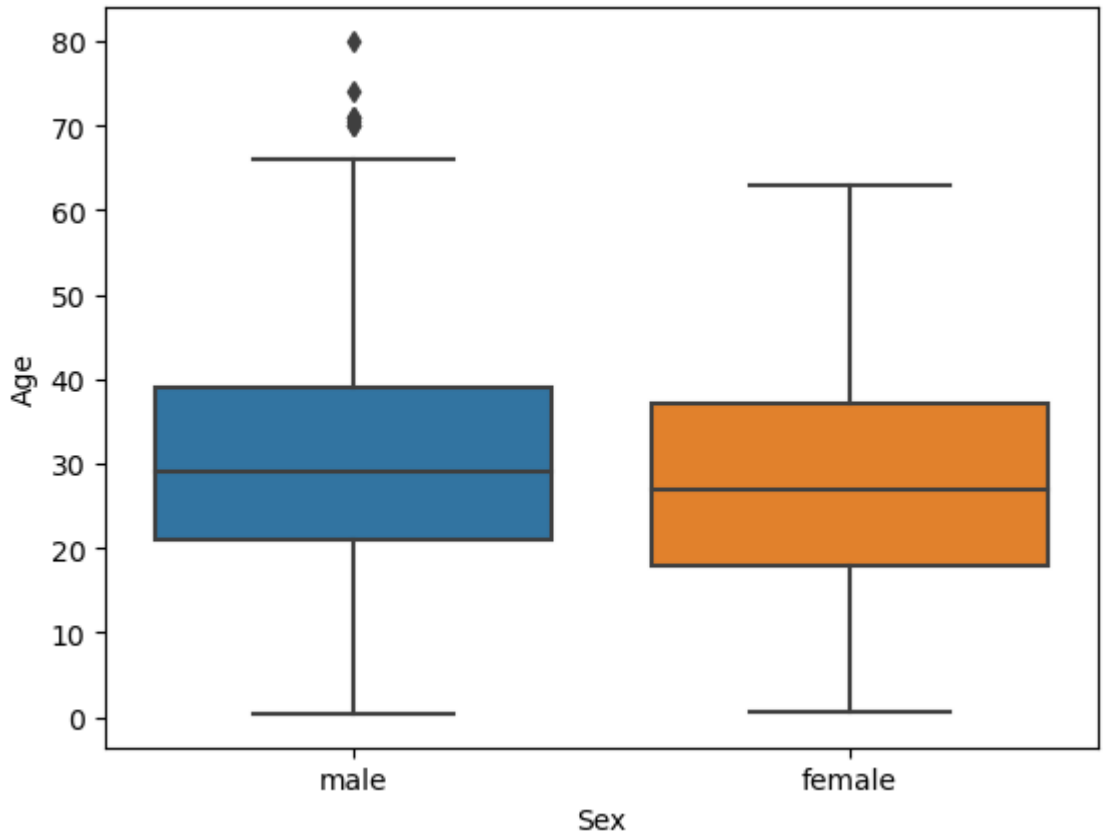
Out[30]:
```
<Axes: xlabel='Pclass', ylabel='Fare'>
```



3<Numerical –Categorical> | Box Plot

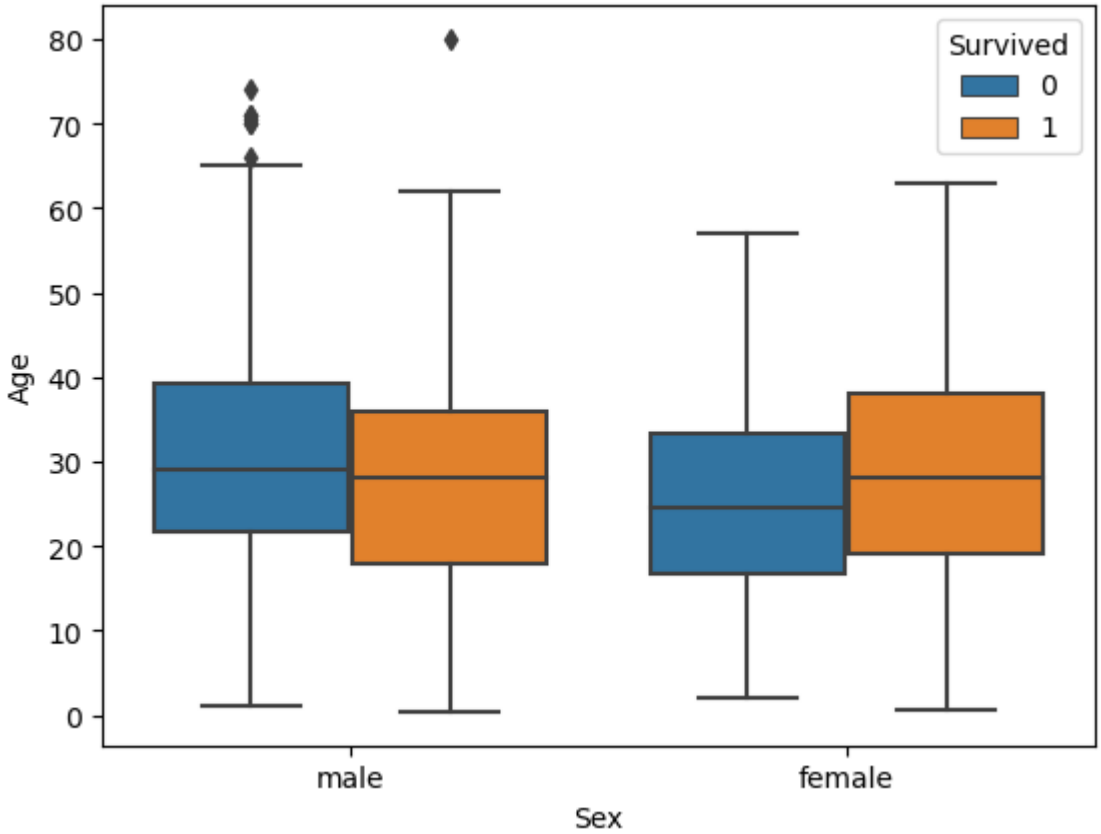
In [31]:
```
sns.boxplot(x=titanic['Sex'],y=titanic['Age'])
```

Out[31]:
```
<Axes: xlabel='Sex', ylabel='Age'>
```



In [32]:
```
sns.boxplot(x=titanic['Sex'],y=titanic['Age'],hue=titanic['Survived'])
```

Out[32]: <Axes: xlabel='Sex', ylabel='Age'>

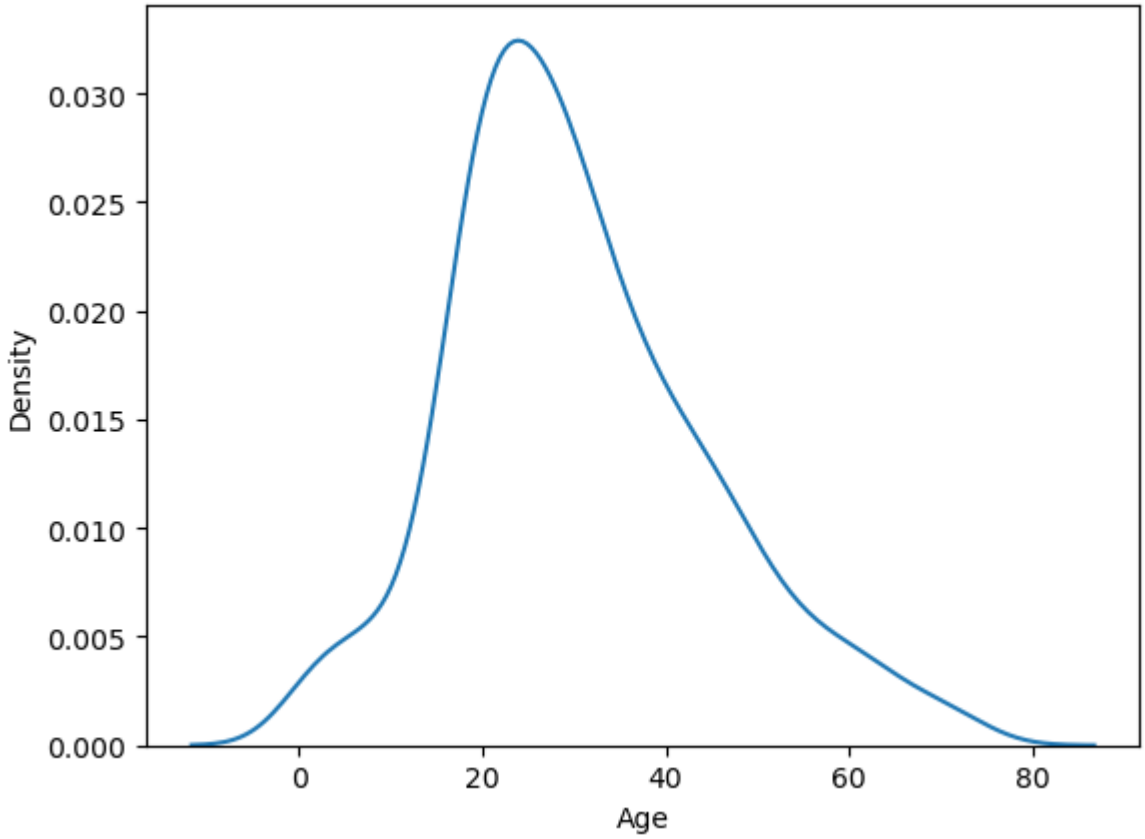


4<Numerical –Categorical> | Dist Plot

In [34]:
```
sns.distplot(titanic[titanic['Survived']==0]['Age'],hist=False)
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_4560\2677591329.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.githu
b.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(titanic[titanic['Survived']==0]['Age'],hist=False)
```

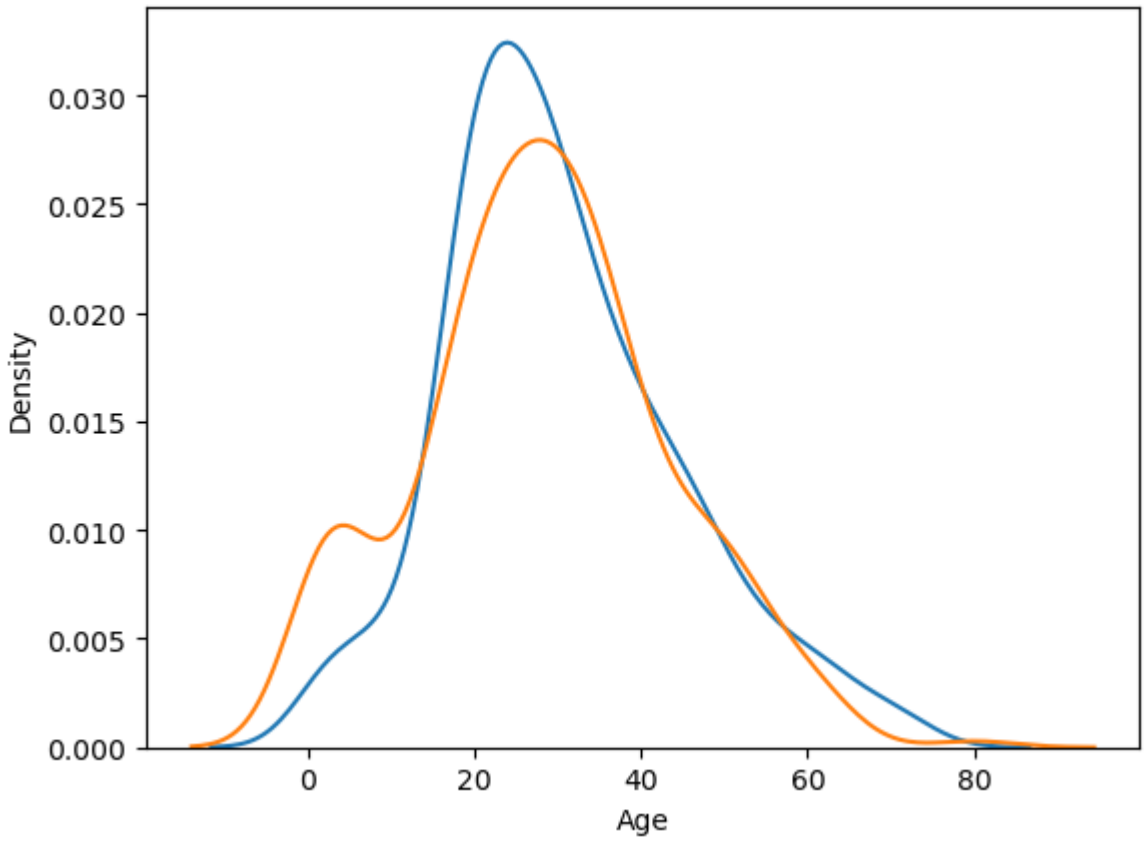
Out[34]:
```
<Axes: xlabel='Age', ylabel='Density'>
```



In [35]:
```
sns.distplot(titanic[titanic['Survived']==0]['Age'],hist=False)
sns.distplot(titanic[titanic['Survived']==1]['Age'],hist=False)
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_4560\1261424998.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.githu
b.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(titanic[titanic['Survived']==0]['Age'],hist=False)
C:\Users\ASUS\AppData\Local\Temp\ipykernel_4560\1261424998.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.githu
b.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(titanic[titanic['Survived']==1]['Age'],hist=False)
```

Out[35]:
```
<Axes: xlabel='Age', ylabel='Density'>
```



5<Categorical – Categorical> | HeatMap

In [45]:
```
titanic
```

Out[45]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	E
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	

891 rows × 12 columns

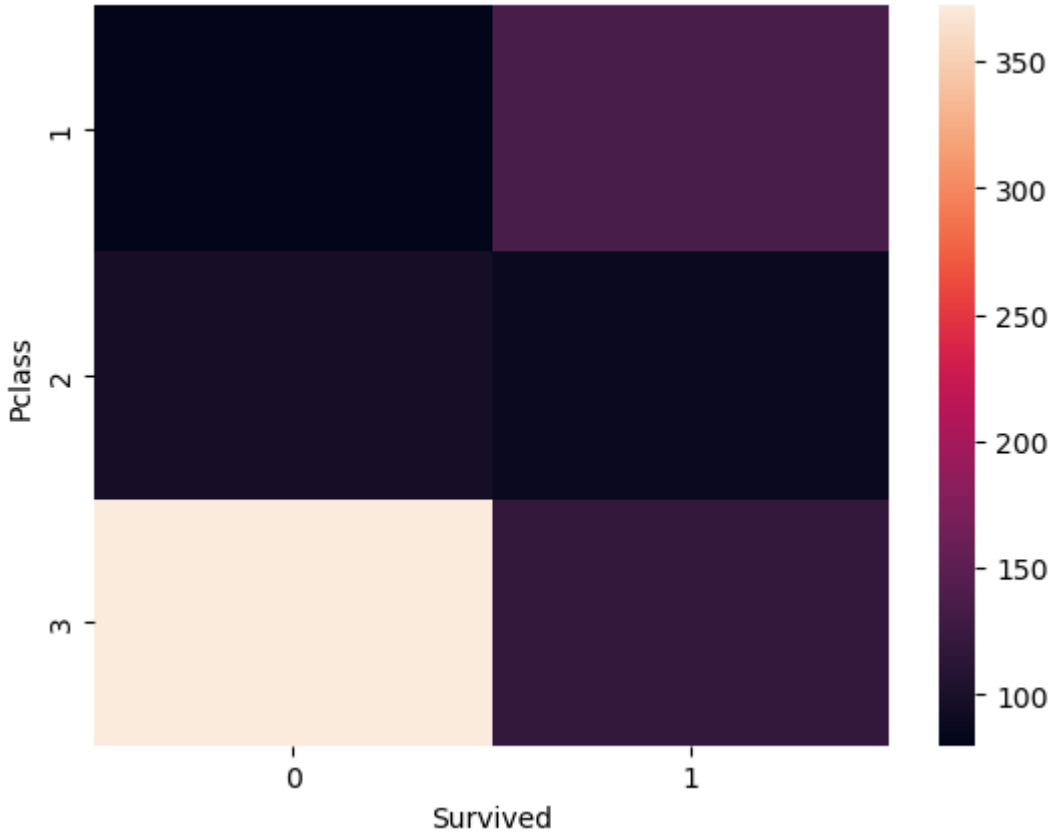
In [37]:
```
pd.crosstab(titanic['Pclass'],titanic['Survived'])
```

Out[37]:

Survived	0	1
Pclass		
1	80	136
2	97	87
3	372	119

In [38]:
```
sns.heatmap(pd.crosstab(titanic['Pclass'],titanic
['Survived']))
```

Out[38]:
```
<Axes: xlabel='Survived', ylabel='Pclass'>
```



6<Categorical – Categorical> | ClusterMap

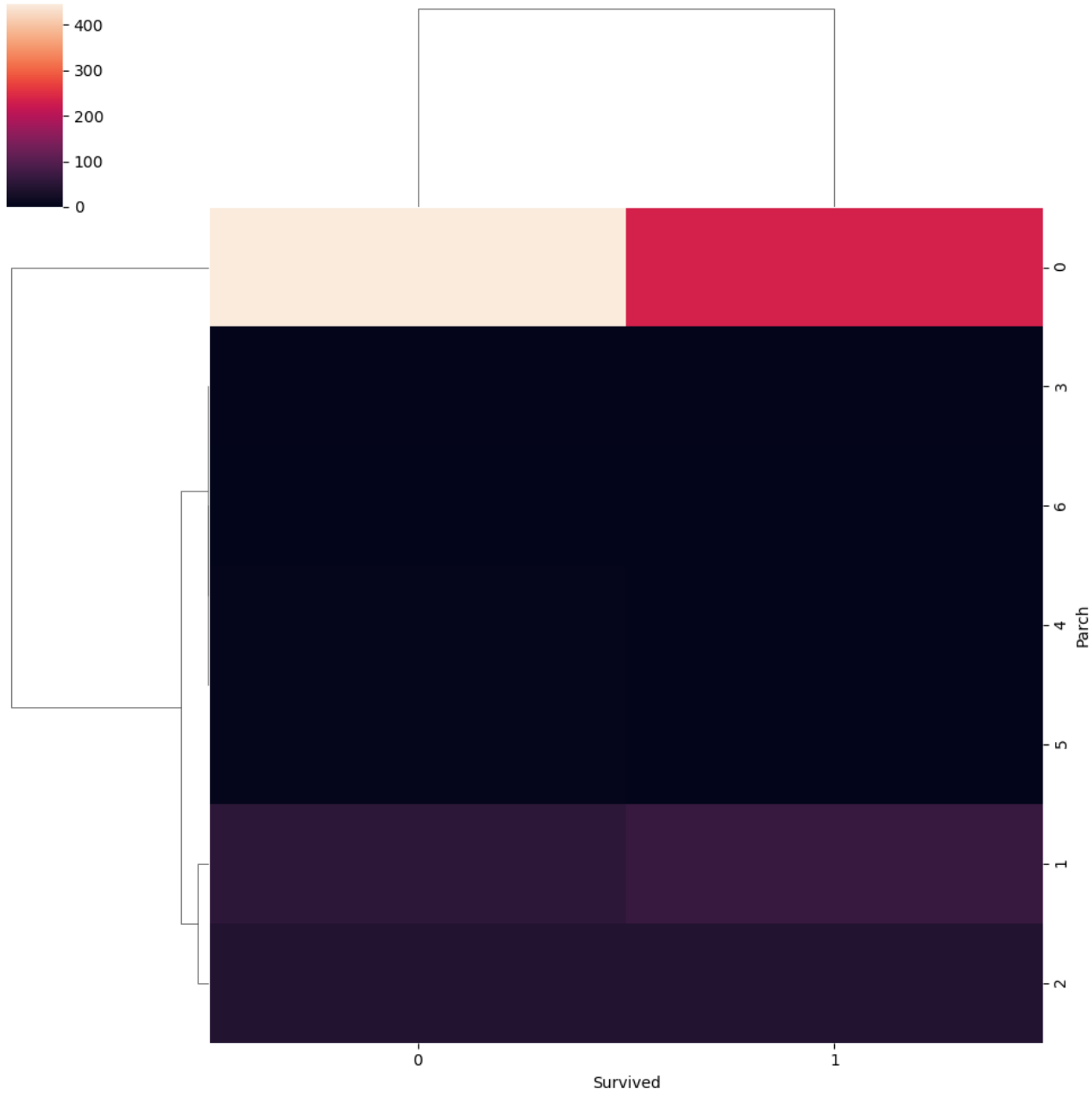
In [46]:
```
pd.crosstab(titanic['SibSp'],titanic['Survived'])
```

Out[46]:

Survived	0	1
SibSp		
0	398	210
1	97	112
2	15	13
3	12	4
4	15	3
5	5	0
8	7	0

In [43]:
```
sns.clustermap(pd.crosstab(titanic['Parch'],titanic['Survived']))
```

Out[43]:
```
<seaborn.matrix.ClusterGrid at 0x2bc24596710>
```



7<Numerical – Numerical -Categorical> | Pare Plot

In [44]:
```
irisflwr.head()
```

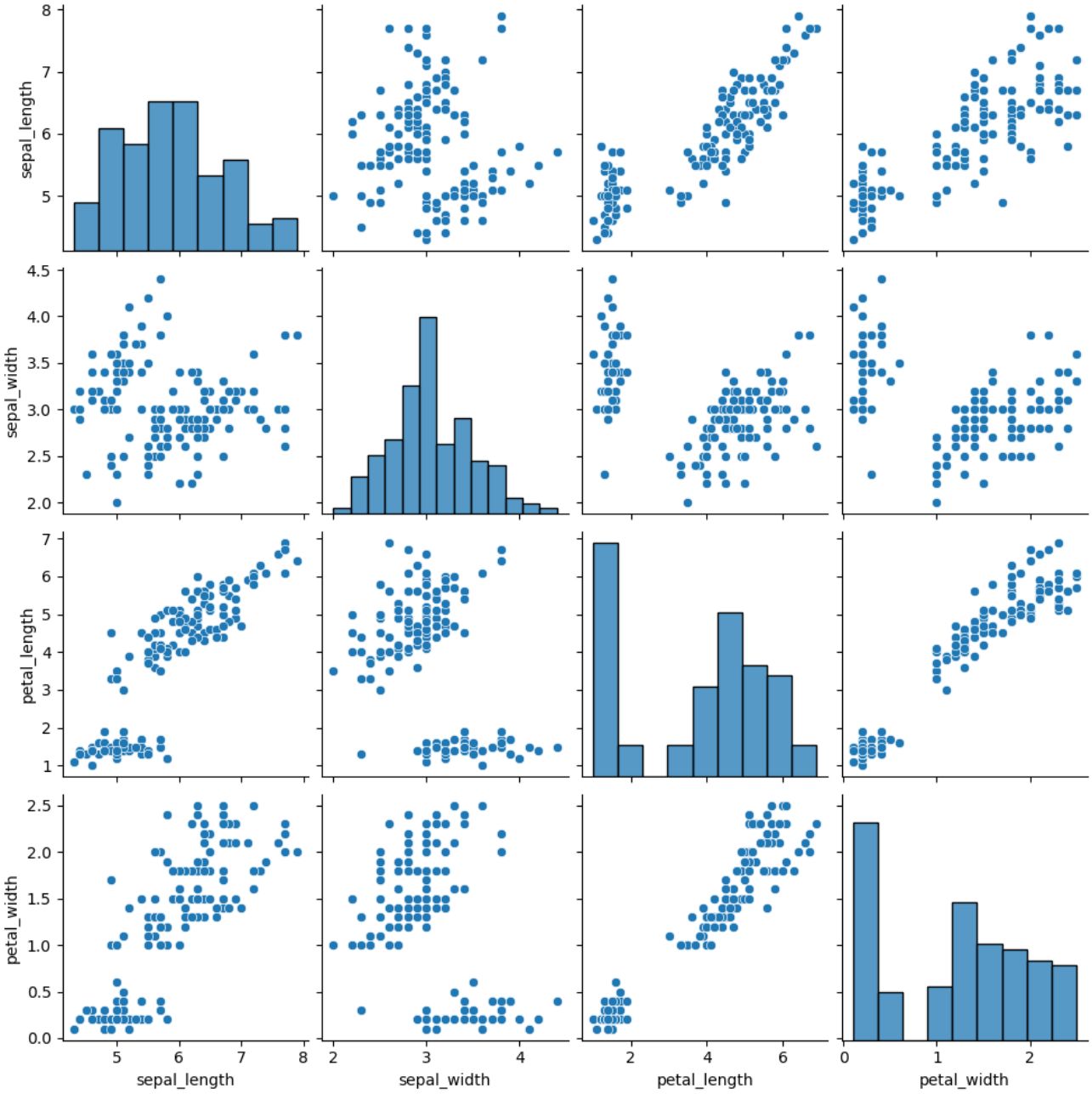
Out[44]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [47]:
```
sns.pairplot(irisflwr)
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The
figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

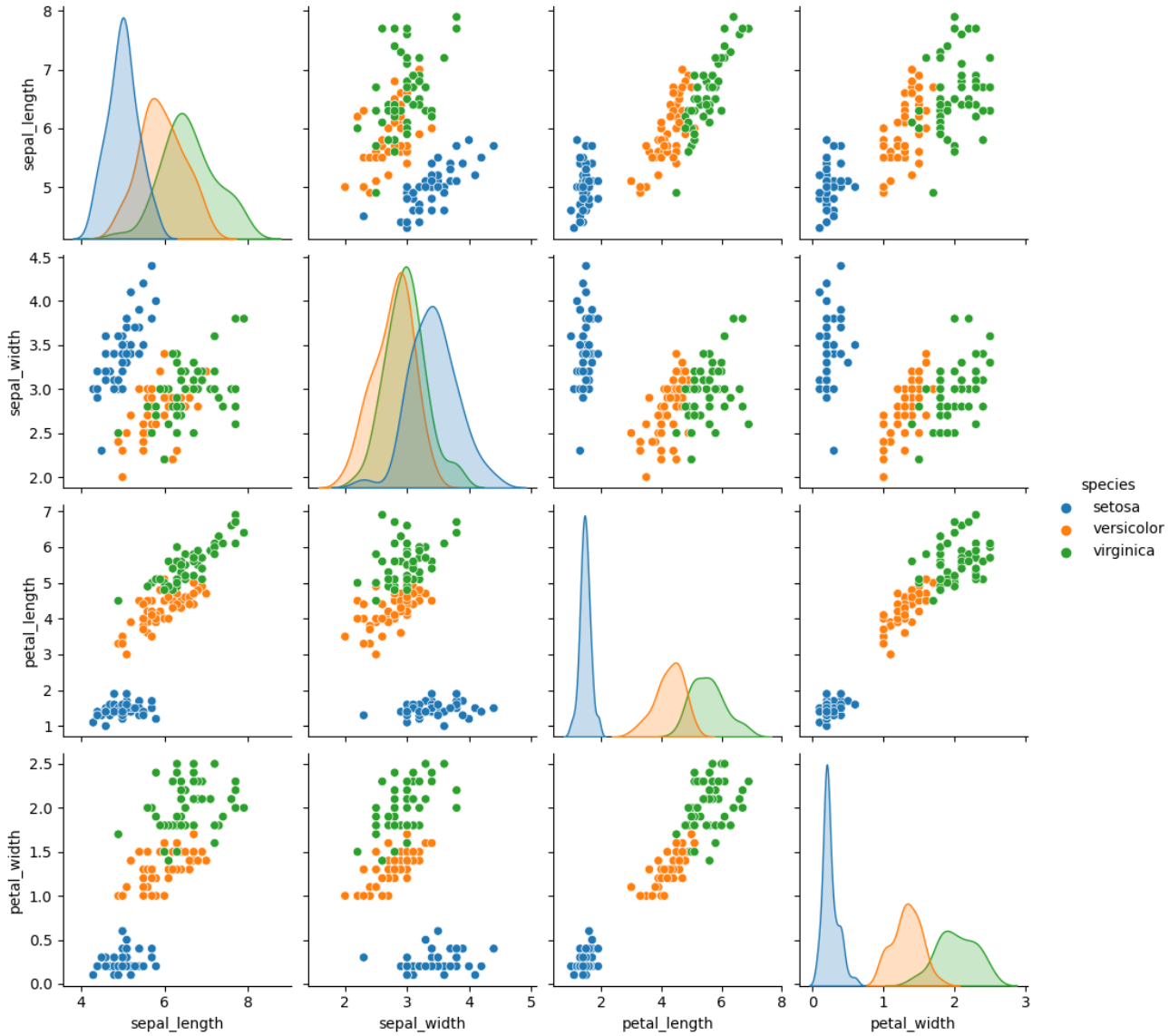
Out[47]:
```
<seaborn.axisgrid.PairGrid at 0x2bc2464bad0>
```



In [48]:
```
sns.pairplot(irisflwr,hue='species')
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The
figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[48]:
```
<seaborn.axisgrid.PairGrid at 0x2bc2592f850>
```



In [ ]:

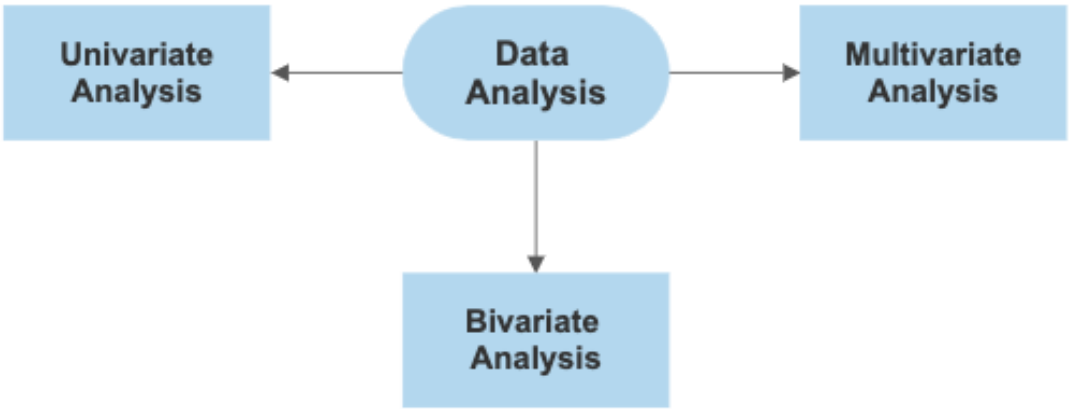
Data Science | 30 Days of Machine Learning | Day - 7

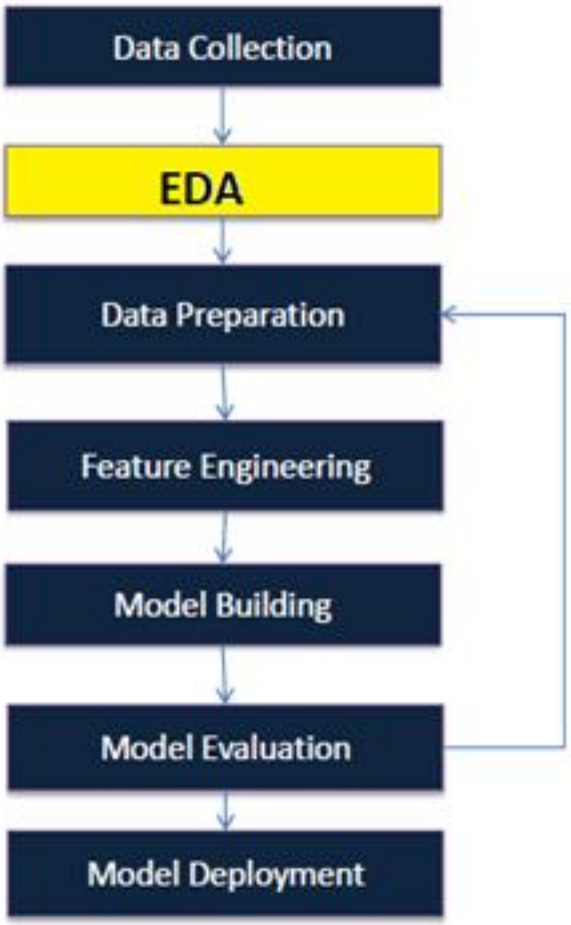
Educator Name: Nishant Dhote
Support Team: **+91-7880-113-112**

----Today Topics | Day 07----

- **EDA: Exploratory Data Analysis**

- **EDA Univariate Analysis (Day 06)**
- **EDA Bivariate Analysis**
- **EDA Multivariate Analysis**





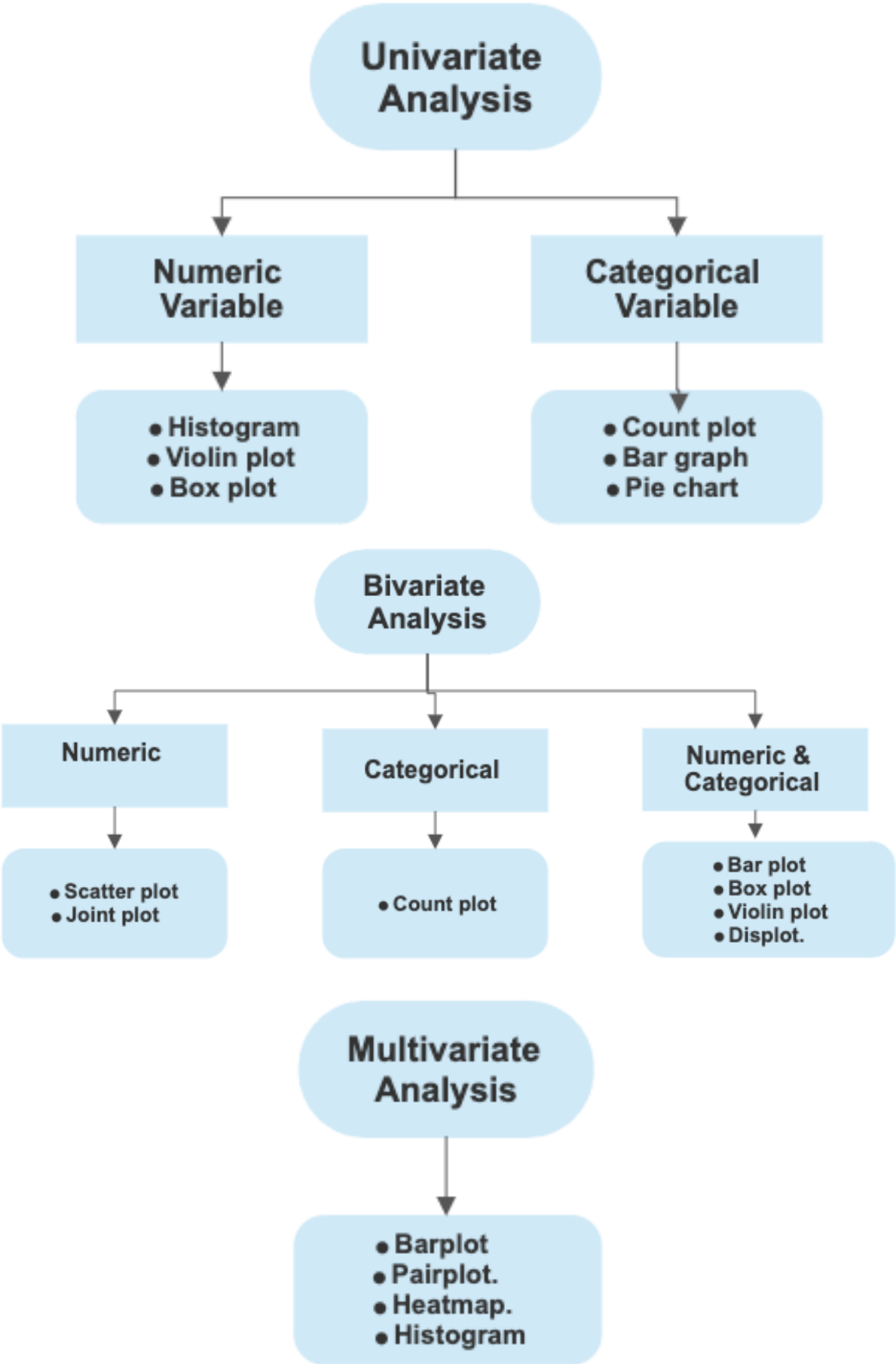
TYPES OF EXPLORATORY DATA ANALYSIS:

1. Univariate Analysis
2. Bivariate Analysis
3. Multivariate Analysis

Univariate EDA involves looking at a single variable at a time. Univariate EDA can help you understand the data distribution and identify any outliers.

Bivariate EDA involves looking at two variables at a time. Bivariate EDA can help you understand the relationship between two variables and identify any patterns that might exist.

Multivariate EDA involves looking at three or more variables at a time. Multivariate EDA can help you understand the relationships between several variables and identify any complex patterns or outliers that might exist.


Univariate Analysis
Numeric Variable
Categorical Variable
• Histogram
• Violin plot
• Box plot
• Count plot
• Bar graph
• Pie chart
Bivariate Analysis
Numeric
Categorical
Numeric & Categorical
• Scatter plot
• Joint plot
• Count plot
• Bar plot
• Box plot
• Violin plot
• Displot.
Multivariate Analysis
• Barplot
• Pairplot.
• Heatmap.
• Histogram

Dataset Link Kaggle: https://www.kaggle.com/competitions/titanic

GitHub Link:
https://github.com/TheiScale/30_Days_Machine_Learning/tree/main/Day%206%20ML

#Import Library

```
import pandas as pd
import seaborn as sns
```

#Import Datasets 1: Titanic

```
titanic = pd.read_csv('train.csv')
```

#Use Datasets 2: Hotel Bill & Tips

```
bill = sns.load_dataset('tips')
```

#Use Datasets 3: USA Flights

```
flights = sns.load_dataset('flights')
```

#Use Datasets 4: Iris Flower

```
irisflwr = sns.load_dataset('iris')
```

#1<Numerical – Numerical> | Scatterplot

```
sns.scatterplot(x=bill['total_bill'],y=bill['tip'])

----

sns.scatterplot(x=bill['total_bill'],y=bill['tip'],hue=bil
l['sex'],style=bill['smoker'],size=bill['size'])
```

#2<Numerical –Categorical> | Bar plot

```
sns.barplot(x=titanic['Pclass'],y=titanic['Age'])
----

sns.barplot(x=titanic['Pclass'],y=titanic['Fare'],
hue=titanic['Sex'])
```

#3<Numerical –Categorical> | Box Plot

```
sns.boxplot(x=titanic['Sex'],y=titanic['Age'])
---
sns.boxplot(x=titanic['Sex'],y=titanic['Age'],hue
 =titanic['Survived'])
```

#4<Numerical –Categorical> | Dist Plot

```
sns.distplot(titanic[titanic['Survived']==0]['Age'
],hist=False)

sns.distplot(titanic[titanic['Survived']==1]['Age'
],hist=False)
```

#5<Categorical – Categorical> | HeatMap

```
titanic.head(5)

---
pd.crosstab(titanic['Pclass'],titanic['Survived'])

---
 sns.heatmap(pd.crosstab(titanic['Pclass'],titanic
['Survived']))
```

#6<Categorical – Categorical> | ClusterMap

```
pd.crosstab(titanic['Parch'],titanic['Survived'])

---

sns.clustermap(pd.crosstab(titanic['Parch'],titani
c['Survived']))
```

#7<Numerical – Numerical -Categorical> | Pare Plot

```
irisflwr.head()

----
sns.pairplot(irisflwr)

----
sns.pairplot(irisflwr,hue='species')
```

Industries Helping Hands

EDA MCQ Question for Placement Practice:

https://www.examveda.com/data-science/practice-mcq-question-on-exploratory-data-analysis-(eda)/

Data Story Telling (Day 7): Curious Data Minds

Data science use cases in government?

How data science is used in national security?

Read Blog: https://activewizards.com/blog/top-12-data-science-use-cases-in-government/

Read Blog: https://www.iasparliament.com/article/data-analytics-and-national-security

Welcome to 30 ML | Day 8

Import Library | pandas

In [1]:
```
import pandas as pd
```

Import Datasets: Titanic

In [2]:
```
df = pd.read_csv('train.csv')
```

In [4]:
```
df.head()
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emb
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	

Pandas Profiling

In [7]:
```
!pip install ydata-profiling
```

```
Requirement already satisfied: ydata-profiling in c:\users\asus\anaconda3\lib\site-p
ackages (4.6.3)
Requirement already satisfied: scipy<1.12,>=1.4.1 in c:\users\asus\anaconda3\lib\sit
e-packages (from ydata-profiling) (1.11.1)
Requirement already satisfied: pandas!=1.4.0,<3,>1.1 in c:\users\asus\anaconda3\lib
\site-packages (from ydata-profiling) (2.0.3)
Requirement already satisfied: matplotlib<3.9,>=3.2 in c:\users\asus\anaconda3\lib\s
ite-packages (from ydata-profiling) (3.7.2)
Requirement already satisfied: pydantic>=2 in c:\users\asus\anaconda3\lib\site-packa
ges (from ydata-profiling) (2.5.2)
Requirement already satisfied: PyYAML<6.1,>=5.0.0 in c:\users\asus\anaconda3\lib\sit
e-packages (from ydata-profiling) (6.0)
Requirement already satisfied: jinja2<3.2,>=2.11.1 in c:\users\asus\anaconda3\lib\si
te-packages (from ydata-profiling) (3.1.2)
Requirement already satisfied: visions[type_image_path]==0.7.5 in c:\users\asus\anac
onda3\lib\site-packages (from ydata-profiling) (0.7.5)
Requirement already satisfied: numpy<1.26,>=1.16.0 in c:\users\asus\anaconda3\lib\si
te-packages (from ydata-profiling) (1.24.3)
Requirement already satisfied: htmlmin==0.1.12 in c:\users\asus\anaconda3\lib\site-p
ackages (from ydata-profiling) (0.1.12)
Requirement already satisfied: phik<0.13,>=0.11.1 in c:\users\asus\anaconda3\lib\sit
e-packages (from ydata-profiling) (0.12.3)
Requirement already satisfied: requests<3,>=2.24.0 in c:\users\asus\anaconda3\lib\si
te-packages (from ydata-profiling) (2.31.0)
Requirement already satisfied: tqdm<5,>=4.48.2 in c:\users\asus\anaconda3\lib\site-p
ackages (from ydata-profiling) (4.65.0)
Requirement already satisfied: seaborn<0.13,>=0.10.1 in c:\users\asus\anaconda3\lib
\site-packages (from ydata-profiling) (0.12.2)
Requirement already satisfied: multimethod<2,>=1.4 in c:\users\asus\anaconda3\lib\si
te-packages (from ydata-profiling) (1.10)
Requirement already satisfied: statsmodels<1,>=0.13.2 in c:\users\asus\anaconda3\lib
\site-packages (from ydata-profiling) (0.14.0)
Requirement already satisfied: typeguard<5,>=4.1.2 in c:\users\asus\anaconda3\lib\si
te-packages (from ydata-profiling) (4.1.5)
Requirement already satisfied: imagehash==4.3.1 in c:\users\asus\anaconda3\lib\site-
packages (from ydata-profiling) (4.3.1)
Requirement already satisfied: wordcloud>=1.9.1 in c:\users\asus\anaconda3\lib\site-
packages (from ydata-profiling) (1.9.3)
Requirement already satisfied: dacite>=1.8 in c:\users\asus\anaconda3\lib\site-packa
ges (from ydata-profiling) (1.8.1)
Requirement already satisfied: numba<0.59.0,>=0.56.0 in c:\users\asus\anaconda3\lib
\site-packages (from ydata-profiling) (0.57.1)
Requirement already satisfied: PyWavelets in c:\users\asus\anaconda3\lib\site-packag
es (from imagehash==4.3.1->ydata-profiling) (1.4.1)
Requirement already satisfied: pillow in c:\users\asus\anaconda3\lib\site-packages
(from imagehash==4.3.1->ydata-profiling) (9.4.0)
Requirement already satisfied: attrs>=19.3.0 in c:\users\asus\anaconda3\lib\site-pac
kages (from visions[type_image_path]==0.7.5->ydata-profiling) (22.1.0)
Requirement already satisfied: networkx>=2.4 in c:\users\asus\anaconda3\lib\site-pac
kages (from visions[type_image_path]==0.7.5->ydata-profiling) (3.1)
Requirement already satisfied: tangled-up-in-unicode>=0.0.4 in c:\users\asus\anacond
a3\lib\site-packages (from visions[type_image_path]==0.7.5->ydata-profiling) (0.2.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\asus\anaconda3\lib\site-p
ackages (from jinja2<3.2,>=2.11.1->ydata-profiling) (2.1.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\asus\anaconda3\lib\site-
packages (from matplotlib<3.9,>=3.2->ydata-profiling) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\asus\anaconda3\lib\site-pack
ages (from matplotlib<3.9,>=3.2->ydata-profiling) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\asus\anaconda3\lib\site
-packages (from matplotlib<3.9,>=3.2->ydata-profiling) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\asus\anaconda3\lib\site
-packages (from matplotlib<3.9,>=3.2->ydata-profiling) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\asus\anaconda3\lib\site-p
ackages (from matplotlib<3.9,>=3.2->ydata-profiling) (23.1)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\asus\anaconda3\lib
```

```
\site-packages (from matplotlib<3.9,>=3.2->ydata-profiling) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\asus\anaconda3\lib\s
ite-packages (from matplotlib<3.9,>=3.2->ydata-profiling) (2.8.2)
Requirement already satisfied: llvmlite<0.41,>=0.40.0dev0 in c:\users\asus\anaconda3
\lib\site-packages (from numba<0.59.0,>=0.56.0->ydata-profiling) (0.40.0)
Requirement already satisfied: pytz>=2020.1 in c:\users\asus\anaconda3\lib\site-pack
ages (from pandas!=1.4.0,<3,>1.1->ydata-profiling) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\asus\anaconda3\lib\site-pa
ckages (from pandas!=1.4.0,<3,>1.1->ydata-profiling) (2023.3)
Requirement already satisfied: joblib>=0.14.1 in c:\users\asus\anaconda3\lib\site-pa
ckages (from phik<0.13,>=0.11.1->ydata-profiling) (1.1.1)
Requirement already satisfied: annotated-types>=0.4.0 in c:\users\asus\anaconda3\lib
\site-packages (from pydantic>=2->ydata-profiling) (0.6.0)
Requirement already satisfied: pydantic-core==2.14.5 in c:\users\asus\anaconda3\lib
\site-packages (from pydantic>=2->ydata-profiling) (2.14.5)
Requirement already satisfied: typing-extensions>=4.6.1 in c:\users\asus\anaconda3\l
ib\site-packages (from pydantic>=2->ydata-profiling) (4.7.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\asus\anaconda3\l
ib\site-packages (from requests<3,>=2.24.0->ydata-profiling) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\asus\anaconda3\lib\site-pack
ages (from requests<3,>=2.24.0->ydata-profiling) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\asus\anaconda3\lib\sit
e-packages (from requests<3,>=2.24.0->ydata-profiling) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\asus\anaconda3\lib\sit
e-packages (from requests<3,>=2.24.0->ydata-profiling) (2023.7.22)
Requirement already satisfied: patsy>=0.5.2 in c:\users\asus\anaconda3\lib\site-pack
ages (from statsmodels<1,>=0.13.2->ydata-profiling) (0.5.3)
Requirement already satisfied: colorama in c:\users\asus\anaconda3\lib\site-packages
(from tqdm<5,>=4.48.2->ydata-profiling) (0.4.6)
Requirement already satisfied: six in c:\users\asus\anaconda3\lib\site-packages (fro
m patsy>=0.5.2->statsmodels<1,>=0.13.2->ydata-profiling) (1.16.0)
```

Import Profile Report

In [8]:
```
from ydata_profiling import ProfileReport
prof = ProfileReport(df)
prof.to_file(output_file='output.html')
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

Export report to file:   0%|          | 0/1 [00:00<?, ?it/s]
```

Import Datasets: Hotel

In [9]:
```
hotel = pd.read_csv('hotel.csv')
```

In [11]:
```
hotel
```

Out[11]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	a
0	Resort Hotel	0	342	2015	July	27	
1	Resort Hotel	0	737	2015	July	27	
2	Resort Hotel	0	7	2015	July	27	
3	Resort Hotel	0	13	2015	July	27	
4	Resort Hotel	0	14	2015	July	27	
...	...	...	...	...	...	...	
119385	City Hotel	0	23	2017	August	35	
119386	City Hotel	0	102	2017	August	35	
119387	City Hotel	0	34	2017	August	35	
119388	City Hotel	0	109	2017	August	35	
119389	City Hotel	0	205	2017	August	35	

119390 rows × 32 columns

In [12]:
```
pip install ydata-profiling
```

```
Requirement already satisfied: ydata-profiling in c:\users\asus\anaconda3\lib\site-p
ackages (4.6.3)Note: you may need to restart the kernel to use updated packages.

Requirement already satisfied: scipy<1.12,>=1.4.1 in c:\users\asus\anaconda3\lib\sit
e-packages (from ydata-profiling) (1.11.1)
Requirement already satisfied: pandas!=1.4.0,<3,>1.1 in c:\users\asus\anaconda3\lib
\site-packages (from ydata-profiling) (2.0.3)
Requirement already satisfied: matplotlib<3.9,>=3.2 in c:\users\asus\anaconda3\lib\s
ite-packages (from ydata-profiling) (3.7.2)
Requirement already satisfied: pydantic>=2 in c:\users\asus\anaconda3\lib\site-packa
ges (from ydata-profiling) (2.5.2)
Requirement already satisfied: PyYAML<6.1,>=5.0.0 in c:\users\asus\anaconda3\lib\sit
e-packages (from ydata-profiling) (6.0)
Requirement already satisfied: jinja2<3.2,>=2.11.1 in c:\users\asus\anaconda3\lib\si
te-packages (from ydata-profiling) (3.1.2)
Requirement already satisfied: visions[type_image_path]==0.7.5 in c:\users\asus\anac
onda3\lib\site-packages (from ydata-profiling) (0.7.5)
Requirement already satisfied: numpy<1.26,>=1.16.0 in c:\users\asus\anaconda3\lib\si
te-packages (from ydata-profiling) (1.24.3)
Requirement already satisfied: htmlmin==0.1.12 in c:\users\asus\anaconda3\lib\site-p
ackages (from ydata-profiling) (0.1.12)
Requirement already satisfied: phik<0.13,>=0.11.1 in c:\users\asus\anaconda3\lib\sit
e-packages (from ydata-profiling) (0.12.3)
Requirement already satisfied: requests<3,>=2.24.0 in c:\users\asus\anaconda3\lib\si
te-packages (from ydata-profiling) (2.31.0)
Requirement already satisfied: tqdm<5,>=4.48.2 in c:\users\asus\anaconda3\lib\site-p
ackages (from ydata-profiling) (4.65.0)
Requirement already satisfied: seaborn<0.13,>=0.10.1 in c:\users\asus\anaconda3\lib
\site-packages (from ydata-profiling) (0.12.2)
Requirement already satisfied: multimethod<2,>=1.4 in c:\users\asus\anaconda3\lib\si
te-packages (from ydata-profiling) (1.10)
Requirement already satisfied: statsmodels<1,>=0.13.2 in c:\users\asus\anaconda3\lib
\site-packages (from ydata-profiling) (0.14.0)
Requirement already satisfied: typeguard<5,>=4.1.2 in c:\users\asus\anaconda3\lib\si
te-packages (from ydata-profiling) (4.1.5)
Requirement already satisfied: imagehash==4.3.1 in c:\users\asus\anaconda3\lib\site-
packages (from ydata-profiling) (4.3.1)
Requirement already satisfied: wordcloud>=1.9.1 in c:\users\asus\anaconda3\lib\site-
packages (from ydata-profiling) (1.9.3)
Requirement already satisfied: dacite>=1.8 in c:\users\asus\anaconda3\lib\site-packa
ges (from ydata-profiling) (1.8.1)
Requirement already satisfied: numba<0.59.0,>=0.56.0 in c:\users\asus\anaconda3\lib
\site-packages (from ydata-profiling) (0.57.1)
Requirement already satisfied: PyWavelets in c:\users\asus\anaconda3\lib\site-packag
es (from imagehash==4.3.1->ydata-profiling) (1.4.1)
Requirement already satisfied: pillow in c:\users\asus\anaconda3\lib\site-packages
(from imagehash==4.3.1->ydata-profiling) (9.4.0)
Requirement already satisfied: attrs>=19.3.0 in c:\users\asus\anaconda3\lib\site-pac
kages (from visions[type_image_path]==0.7.5->ydata-profiling) (22.1.0)
Requirement already satisfied: networkx>=2.4 in c:\users\asus\anaconda3\lib\site-pac
kages (from visions[type_image_path]==0.7.5->ydata-profiling) (3.1)
Requirement already satisfied: tangled-up-in-unicode>=0.0.4 in c:\users\asus\anacond
a3\lib\site-packages (from visions[type_image_path]==0.7.5->ydata-profiling) (0.2.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\asus\anaconda3\lib\site-p
ackages (from jinja2<3.2,>=2.11.1->ydata-profiling) (2.1.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\asus\anaconda3\lib\site-
packages (from matplotlib<3.9,>=3.2->ydata-profiling) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\asus\anaconda3\lib\site-pack
ages (from matplotlib<3.9,>=3.2->ydata-profiling) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\asus\anaconda3\lib\site
-packages (from matplotlib<3.9,>=3.2->ydata-profiling) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\asus\anaconda3\lib\site
-packages (from matplotlib<3.9,>=3.2->ydata-profiling) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\asus\anaconda3\lib\site-p
ackages (from matplotlib<3.9,>=3.2->ydata-profiling) (23.1)
```

```
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\asus\anaconda3\lib
\site-packages (from matplotlib<3.9,>=3.2->ydata-profiling) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\asus\anaconda3\lib\s
ite-packages (from matplotlib<3.9,>=3.2->ydata-profiling) (2.8.2)
Requirement already satisfied: llvmlite<0.41,>=0.40.0dev0 in c:\users\asus\anaconda3
\lib\site-packages (from numba<0.59.0,>=0.56.0->ydata-profiling) (0.40.0)
Requirement already satisfied: pytz>=2020.1 in c:\users\asus\anaconda3\lib\site-pack
ages (from pandas!=1.4.0,<3,>1.1->ydata-profiling) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\asus\anaconda3\lib\site-pa
ckages (from pandas!=1.4.0,<3,>1.1->ydata-profiling) (2023.3)
Requirement already satisfied: joblib>=0.14.1 in c:\users\asus\anaconda3\lib\site-pa
ckages (from phik<0.13,>=0.11.1->ydata-profiling) (1.1.1)
Requirement already satisfied: annotated-types>=0.4.0 in c:\users\asus\anaconda3\lib
\site-packages (from pydantic>=2->ydata-profiling) (0.6.0)
Requirement already satisfied: pydantic-core==2.14.5 in c:\users\asus\anaconda3\lib
\site-packages (from pydantic>=2->ydata-profiling) (2.14.5)
Requirement already satisfied: typing-extensions>=4.6.1 in c:\users\asus\anaconda3\l
ib\site-packages (from pydantic>=2->ydata-profiling) (4.7.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\asus\anaconda3\l
ib\site-packages (from requests<3,>=2.24.0->ydata-profiling) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\asus\anaconda3\lib\site-pack
ages (from requests<3,>=2.24.0->ydata-profiling) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\asus\anaconda3\lib\sit
e-packages (from requests<3,>=2.24.0->ydata-profiling) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\asus\anaconda3\lib\sit
e-packages (from requests<3,>=2.24.0->ydata-profiling) (2023.7.22)
Requirement already satisfied: patsy>=0.5.2 in c:\users\asus\anaconda3\lib\site-pack
ages (from statsmodels<1,>=0.13.2->ydata-profiling) (0.5.3)
Requirement already satisfied: colorama in c:\users\asus\anaconda3\lib\site-packages
(from tqdm<5,>=4.48.2->ydata-profiling) (0.4.6)
Requirement already satisfied: six in c:\users\asus\anaconda3\lib\site-packages (fro
m patsy>=0.5.2->statsmodels<1,>=0.13.2->ydata-profiling) (1.16.0)
```

In [13]:

```python
from ydata_profiling import ProfileReport
prof=ProfileReport(hotel)
prof.to_file(output_file='output.html')
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

Export report to file:   0%|          | 0/1 [00:00<?, ?it/s]
```

In [ ]:

Data Science | 30 Days of Machine Learning | Day - 8

Educator Name: Nishant Dhote
Support Team: **+91-7880-113-112**

----Today Topics | Day 08----

- **Panda Profiling**
  How do we use the pandas profiling tool?
- **Feature Engineering**
- Feature Transformation
- Feature Construction
- Feature Selection
- Feature Selection

Dataset Link Kaggle: https://www.kaggle.com/competitions/titanic

GitHub Link: https://github.com/TheiScale/30_Days_Machine_Learning/

Github Link:

#Import Library | pandas

```
import pandas as pd
```

#Import Datasets: Titanic

```
df = pd.read_csv('train.csv')
```

#View Datasets

```
df.head()
```

#Pandas Profiling

```
pip install ydata-profiling
```

#Import Profile Report

```
from ydata_profiling import ProfileReport
prof = ProfileReport(df)
prof.to_file(output_file='output.html')
```


Generate and Analysis Report <output.html>

#Import Datasets: Hotel

```
hotel = pd.read_csv('hotel.csv')
```

#View Datasets

```
hotel.head()
```

#Pandas Profiling

```
pip install ydata-profiling
```

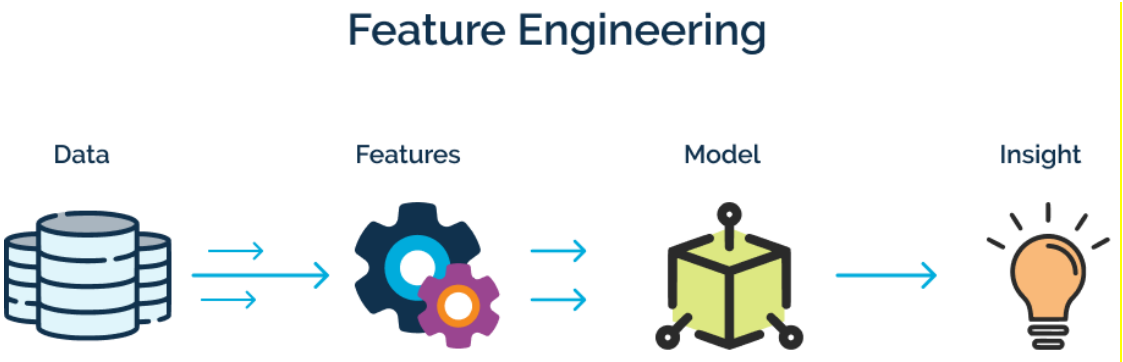
#Import Profile Report

```
from ydata_profiling import ProfileReport
prof=ProfileReport(hotel)
prof.to_file(output_file='output.html')
```


Generate and Analysis Report <output.html>

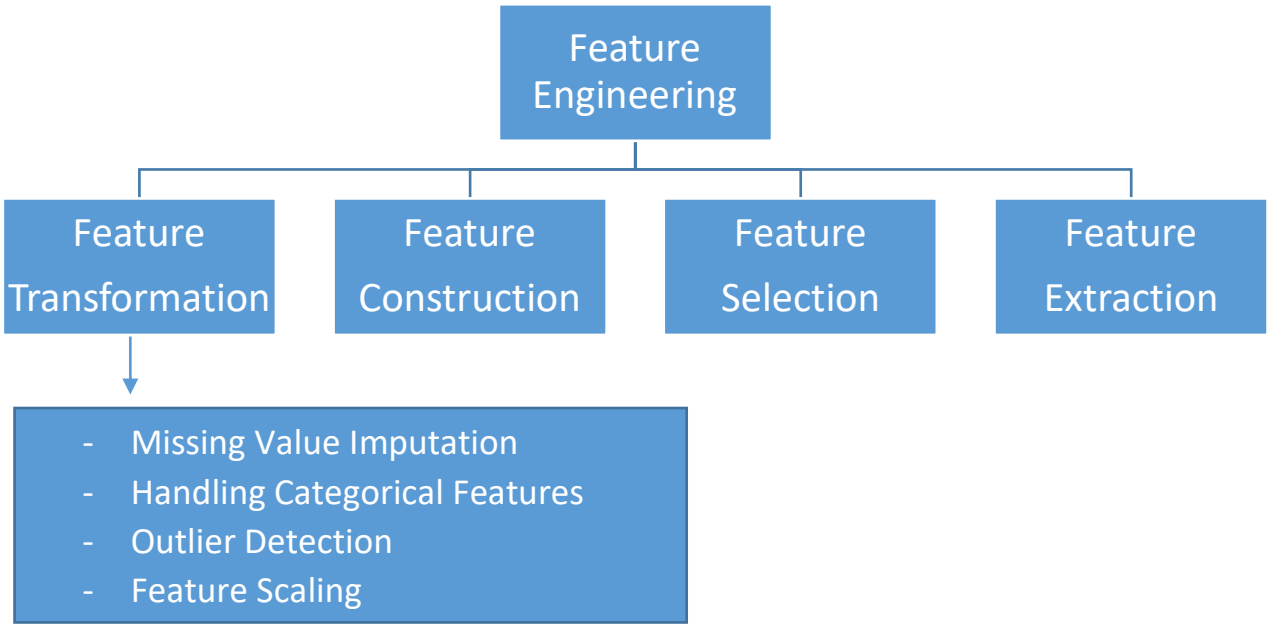
- Feature Engineering

- MLDLC
  - Frame the problem → Gathering data →Data pre-processing → Exploratory data analysis (EDA) → **Feature Engineering**



What is Feature Engineering:

Feature engineering is the process that takes raw data and transforms it into features that can be used to create a predictive model using machine learning.



- Missing Value Imputation:

What are imputed values?
Imputed value, also known as estimated imputation, is an assumed value given to an item when the actual value is not known or available. Imputed values are a logical or implicit value for an item or time set, wherein a "true" value has yet to be ascertained.

Missing values

PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	male	22	1	0	A/5 21171	7.25		S
2	1	1	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	female	26	0	0	STON/O2. 3101282	7.925		S
4	1	1	female	35	1	0	113803	53.1	C123	S
5	0	3	male	35	0	0	373450	8.05		S
6	0	3	male		0	0	330877	8.4583		Q

Average_Age = 26.0

ID	City	Age	Married ?
1	Lisbon	25	0
2	Berlin	25	1
3	Lisbon	30	1
4	Lisbon	30	1
5	Berlin	18	0
6	Lisbon	NaN	0
7	Berlin	30	1
8	Berlin	NaN	0
9	Berlin	25	1
10	Madrid	25	1

ID	City	Age	Married ?
1	Lisbon	25	0
2	Berlin	25	1
3	Lisbon	30	1
4	Lisbon	30	1
5	Berlin	18	0
6	Lisbon	26	0
7	Berlin	30	1
8	Berlin	26	0
9	Berlin	25	1
10	Madrid	25	1

- Handling Categorical Values

How do you handle categorical data?
One of the most common ways to deal with categorical data in machine learning is through a process called one-hot encoding. This technique involves converting categorical data into numerical data by creating a new binary feature for each category.

Index	Animal
0	Dog
1	Cat
2	Sheep
3	Horse
4	Lion

One-Hot code

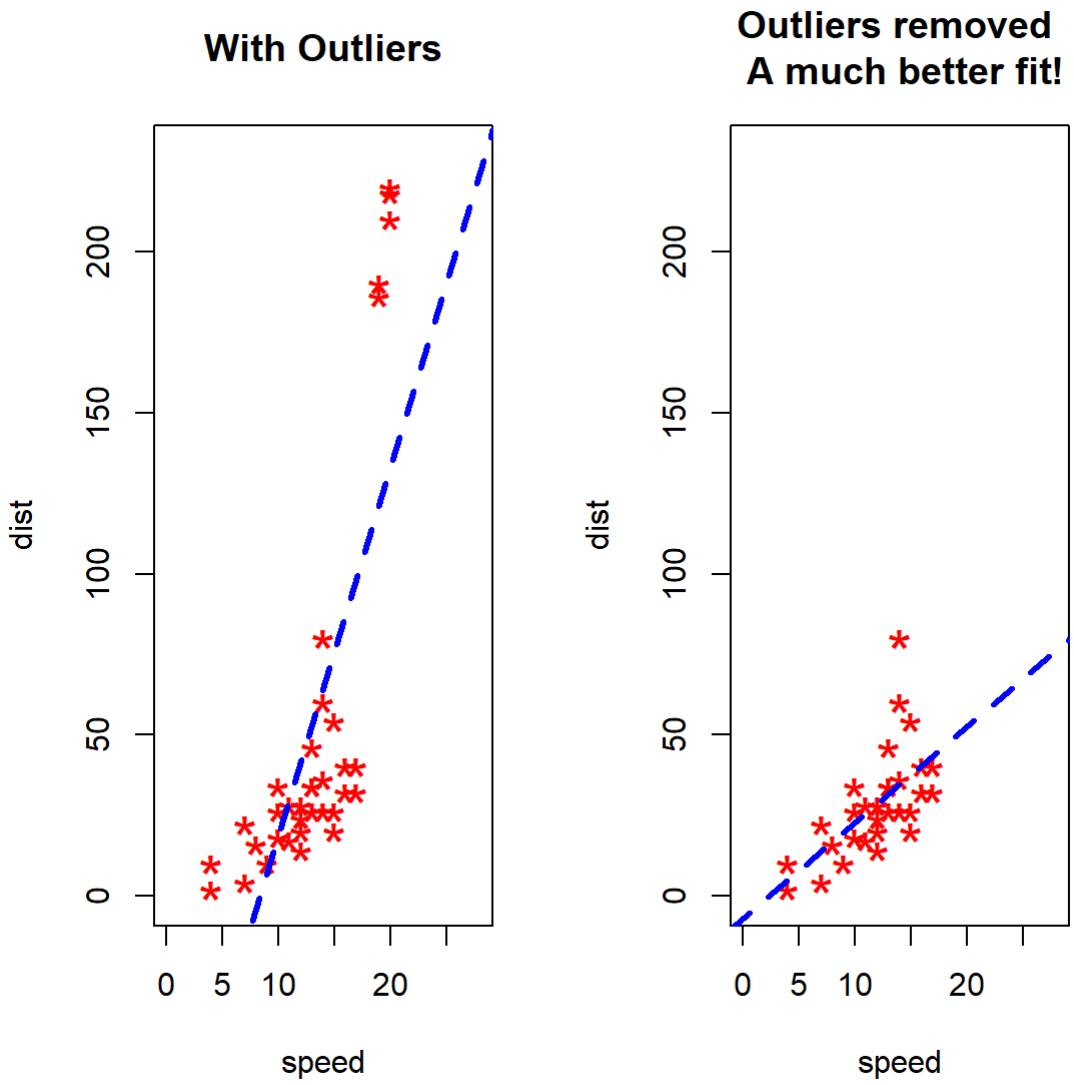
Index	Dog	Cat	Sheep	Lion	Horse
0	1	0	0	0	0
1	0	1	0	0	0
2	0	0	1	0	0
3	0	0	0	0	1
4	0	0	0	1	0

Helping Hands

- Outlier Detection

Outlier detection is the process of detecting outliers, or a data point that is far away from the average, and depending on what you are trying to accomplish, potentially removing or resolving them from the analysis to prevent any potential skewing.

How do you find outliers in linear regression?
We can do this visually in the scatter plot by drawing an extra pair of lines that are two standard deviations above and below the best-fit line. Any data points outside this extra pair of lines are flagged as potential outliers.

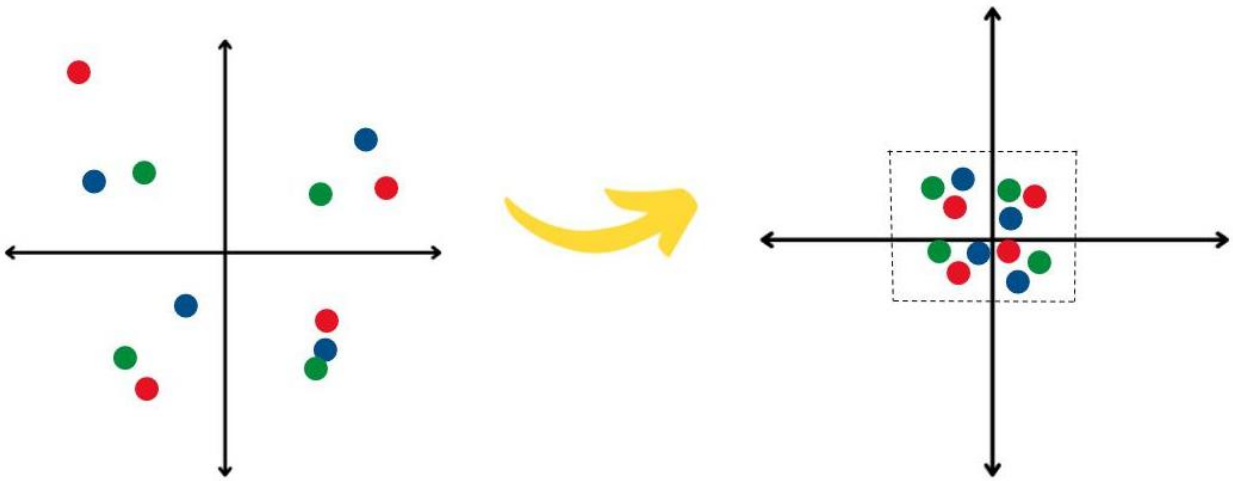


- Feature Scaling

What is Feature Scaling?

Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data pre-processing step.

	A	B	C	D
1	Country	Age	Salary	Purchased
2	France	44	72000	0
3	Spain	27	48000	1
4	Germany	30	54000	0
5	Spain	38	61000	0
6	Germany	40	1000	1
7	France	35	58000	1
8	Spain	78	52000	0
9	France	48	79000	1
10	Germany	50	83000	0
11	France	37	67000	1



- **Feature Construction**

  In the case of the Titanic dataset, two columns are available: “sibsp” (number of siblings/spouses aboard) and “parent” (number of parents/children aboard). To create a new feature called “family type,” you can combine these columns and assign a specific value to indicate the family size.

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

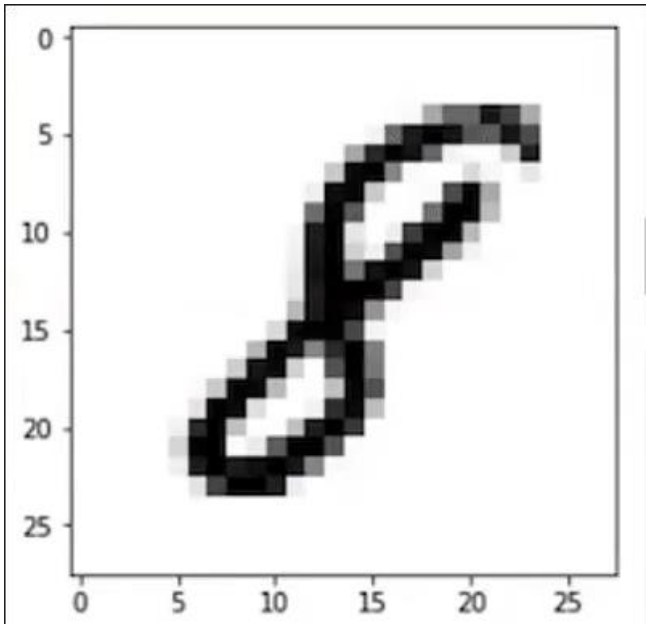
Helping Hands

- **Feature selection**

  Feature selection is the process of reducing the number of input variables when developing a predictive model.

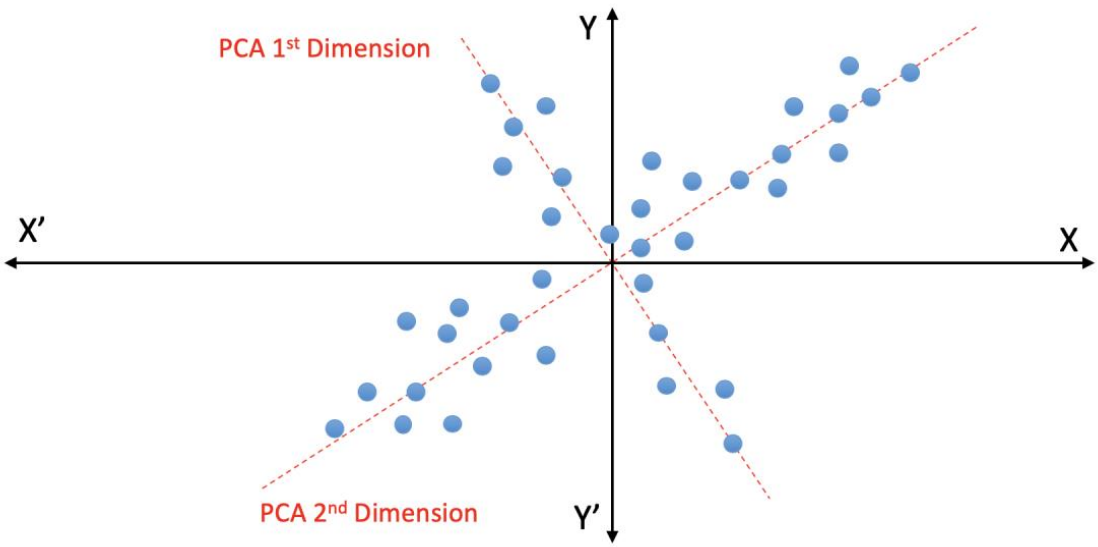
  It is desirable to reduce the number of input variables to both reduce the computational cost of modelling and, in some cases, to improve the performance of the model.

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0



- **Feature Extraction**

What Is Feature Extraction?
Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. It yields better results than applying machine learning directly to the raw data.



Data Story Telling (Day 8): Curious Data Minds

How Uber uses data science to reinvent transportation?

Read Blog: https://www.projectpro.io/article/how-uber-uses-data-science-to-reinvent-transportation/290

Read Blog: https://jagan-singhh.medium.com/data-science-at-uber-4380bf8f6aca

Movie Trailer Link (Super Pumped) : https://youtu.be/VMP21LO0Guc

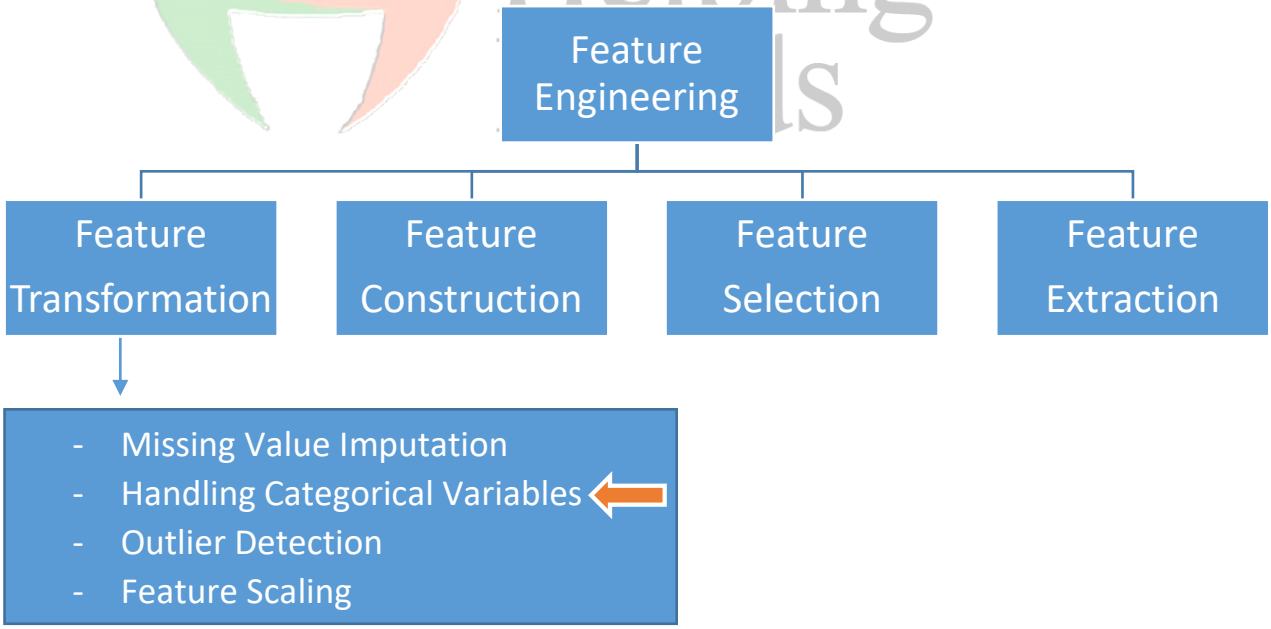
Data Science | 30 Days of Machine Learning | Day - 9

Educator Name: Nishant Dhote
Support Team: **+91-7880-113-112**

----Today Topics | Day 09----

- **How to Encode "Categorical Variables"?**
- What is categorical data?
- **Type of categorical data?**
- What is Ordinal Data?
- Ordinal Encoding
- Label Encoding
- What is Nominal Data?
- One Hot Encoding

Dataset Link GitHub: https://github.com/TheiScale/30_Days_Machine_Learning/



- What is categorical data?

Categorical data refers to a form of information that can be stored and identified based on their names or labels. It is a type of qualitative data that can be grouped into categories instead of being measured numerically.

This data type is made up of categorical variables that show things like a person's gender, hometown, and so on. Categorical measurements are not given in numbers but rather in natural language descriptions.

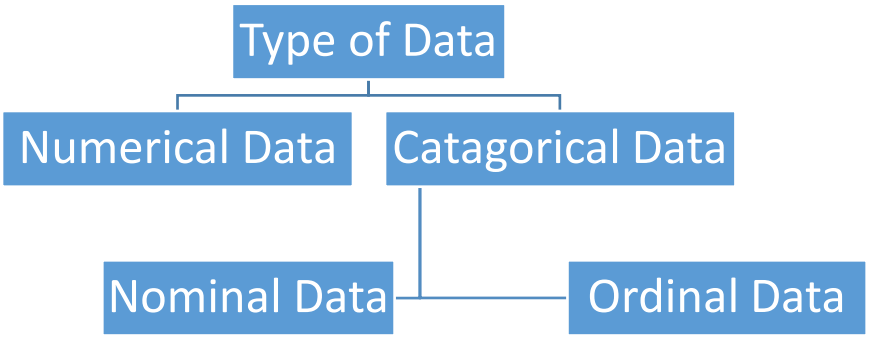
Numbers can sometimes represent it, but those numbers don't mean anything mathematically. The following are some examples of this data type:

- Birthdate
- Favourite sport
- Hair Colour
- Height

In the above example, both the birthdate and the postcode are made up of numbers. It is regarded as categorical data even though it includes numbers. Calculating the average is a simple way to determine if the provided data is categorical or numerical.

If you can figure out the average, it is considered numerical data. If you can't figure out the average, then it's considered categorical data.

- **Type of categorical data?**



- **Nominal Data**

Nominal data is a type of data that consists of categories that can't be ordered or ranked. It is also called a nominal scale. Nominal data can't be ranked or measured in any way. Still, nominal data can be both qualitative and quantitative at times.

Some examples of nominal data are symbols, words, letters, and the gender of a person, state and your engineering branch.

- **Ordinal Data**

Ordinal data is a category of data that has a natural order. It is often used in surveys, questionnaires, and the fields of finance and economics. Ordinal data stands out since it is impossible to differentiate between data values.

Clothing sizes are one example of this type of data (small, medium, and large are not measurable differences, but they are clearly ordered to show size comparisons). Exam Grade or Division.

- **What is Encoding?**

Data encoding involves converting a sequence of text characters into binary code so computers, which operate using binary numbers, can process, store, or transmit that textual information. Decoding occurs when the information is then translated from binary form into a readable version.

"Ordinal Encoding"

When we have a feature where variables have some order/rank.

Original Encoding	Ordinal Encoding
Poor	1
Good	2
Very Good	3
Excellent	4

For example: Student's performance, Customer's review, Education of person, etc…

In the above example, we have orders/ranks/sequences. We can assign ranks based on student's performance, based on feedback given by customers, based on the highest education of the person. Those features are called Ordinal features.

"Nominal Encoding"

When we have a feature where variables are just names and there is no order or rank to this variable's feature.

Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1
0	1	0

For example: City of person lives in, Gender of person, Marital Status, etc…

In the above example, we do not have any order or rank, or sequence. All the variables in the respective feature are equal. We can't give them any orders or ranks. Those features are called Nominal features.

Why ordinal encoding is referred as a Label Encoding?

Ordinal encoding is a technique that is used to transform categorical variables into a numerical format by assigning a unique value to each of its categories. It is also referred to as Label Encoding.

For example, we have customer feedback data based on a survey or online feedback mechanism. It contains categories - very dissatisfied, dissatisfied, neutral, satisfied, and very satisfied. To encode this variable using ordinal encoding, we can assign numerical values as mentioned below –

very dissatisfied - 1
dissatisfied - 2
neutral - 3
satisfied - 4
very satisfied – 5

Ordinal encoding assumes that categories in categorical variables have clear, natural, and intrinsic ordering to their categories. It does not work

for nominal categorical variables as no relationship exists between categories of a nominal variable. In our previous example, we encoded the categorical variable by assigning the lowest numerical value of 1 to the very dissatisfied category and the highest value of 5 to the very satisfied category.

.

This way, we were able to preserve the natural ordering of the categories - very dissatisfied < dissatisfied < neutral < satisfied < very satisfied was retained in 1 < 2 < 3 < 4 < 5. Suppose we have another categorical variable, which contains red, blue, and green categories. We can encode this variable using ordinal encoding by assigning 1 to red, 2 to blue, and 3 to green, but it may lead to incorrect results. As encoded values have a natural ordering between them - 1 < 2 < 3 will be there, but red < blue < green does not exist.

---Start Coding---

#Import Library | pandas | numpy

```
import numpy as np
import pandas as pd
```

#Import Datasets

```
df = pd.read_csv('customer.csv')
```

#Review Sample Data

```
df.sample(5)
```

#Eliminate Unnecessary Columns

```
df = df.iloc[:,2:]
---
df
df.head()
```

#Train_Test_Split

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test =
train_test_split(df.iloc[:,0:2],df.iloc[:,-
1:],test_size=0.2)
```

#Import_OrdinalEncoder

```
from sklearn.preprocessing import OrdinalEncoder
```

```
x_train
```

#Define Categories

```
oe =
OrdinalEncoder(categories=[['Poor','Average','Good'],['Sch
ool','UG','PG']])
```

```
oe.fit(x_train)
```

#Transform Ordinal Encoding

```
x_train = oe.transform(x_train)
x_test = oe.transform(x_test)
```

```
x_train
```

```
oe.categories_
```

#Label Encoding

```
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
```

```
---
le.fit(y_train)

---
le.classes_

---
y_train = le.transform(y_train)
y_test = le.transform(y_test)

---
y_train
```

- One Hot Encoding for Nominal Data

	brand	km_driven	fuel	owner	selling_price
0	Maruti	145500	Diesel	First Owner	450000
1	Skoda	120000	Diesel	Second Owner	370000
2	Honda	140000	Petrol	Third Owner	158000
3	Hyundai	127000	Diesel	First Owner	225000
4	Maruti	120000	Petrol	First Owner	130000

Difference Between Label Encoding VS OHE

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

→

One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

---Start Coding---

#Import Library | pandas | numpy

```
import numpy as np
import pandas as pd
```

#Import Datasets

```
df = pd.read_csv('cars.csv')
```

#Review Sample Data

```
df.sample(5)
```

```
---
df['owner'].value_counts()
```

```
---
df['brand'].value_counts()
df['brand'].nunique()
```

```
---
df['fuel'].value_counts()
df['fuel'].nunique()
```

#1. One Hot Encoding Using Pandas

```
pd.get_dummies(df,columns=['fuel','owner'])
```

#2. K-1 One Hot Encoding

```
pd.get_dummies(df,columns=['fuel','owner'],drop_first=True)
```

#3. One hot encoding Using Sklearn

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test =
train_test_split(df.iloc[:,0:4],df.iloc[:,-
 1],test_size=0.2,random_state=2)
```

```
---
df.head()
---
X_train.head()
---
X_test.head()
```

Import OHE: SK Learn

```
from sklearn.preprocessing import OneHotEncoder
---
ohe = OneHotEncoder
---
X_train_new =
ohe.fit_transform(X_train[['fuel','owner']]).toarray()
---
X_test_new =
ohe.transform(X_test[['fuel','owner']]).toarray()
---
X_train_new
---
X_train_new.shape
---
```

```
np.hstack((X_train[['brand','km_driven']].values,X_train_new))
```

MCQ Question:

https://www.aionlinecourse.com/ai-quiz-questions/machine-learning/data-pre-processing

Data Story Telling (Day 9): Curious Data Minds

Suggest Topic in Comment Box?

Industries Helping Hands

Import Library | pandas | numpy ¶

In [71]:
```
import numpy as np
import pandas as pd
```

Import Datasets

In [72]:
```
df = pd.read_csv('cars.csv')
```

In [73]:
```
df.sample(5)
```

Out[73]:

	brand	km_driven	fuel	owner	selling_price
2230	Mahindra	50000	Diesel	Second Owner	250000
5904	Honda	50000	Diesel	First Owner	1050000
2114	Maruti	46000	Petrol	Fourth & Above Owner	75000
6940	Mahindra	90000	Diesel	First Owner	550000
4665	Honda	85000	Petrol	Third Owner	476999

In [74]:
```
df['owner'].value_counts()
```

Out[74]:
```
owner
First Owner             5289
Second Owner            2105
Third Owner              555
Fourth & Above Owner     174
Test Drive Car             5
Name: count, dtype: int64
```

In [75]:
```
df['brand'].value_counts()
df['brand'].nunique()
```

Out[75]:
```
32
```

In [76]:
```
df['fuel'].value_counts()
```

Out[76]:
```
fuel
Diesel    4402
Petrol    3631
CNG         57
LPG         38
Name: count, dtype: int64
```

1. One Hot Encoding Using Pandas

In [117]:
```
pd.get_dummies(df,columns=['fuel','owner'])
```

Out[117]:

	brand	km_driven	selling_price	fuel_CNG	fuel_Diesel	fuel_LPG	fuel_Petrol	owner_First Owner	owner_ &
0	Maruti	145500	450000	False	True	False	False	True	
1	Skoda	120000	370000	False	True	False	False	False	
2	Honda	140000	158000	False	False	False	True	False	
3	Hyundai	127000	225000	False	True	False	False	True	
4	Maruti	120000	130000	False	False	False	True	True	
...	...	...	...	...	...	...	...	...	
8123	Hyundai	110000	320000	False	False	False	True	True	
8124	Hyundai	119000	135000	False	True	False	False	False	
8125	Maruti	120000	382000	False	True	False	False	True	
8126	Tata	25000	290000	False	True	False	False	True	
8127	Tata	25000	290000	False	True	False	False	True	

8128 rows × 12 columns

2. K-1 One Hot Encoding

In [118]:
```
pd.get_dummies(df,columns=['fuel','owner'],drop_first=True)
```

Out[118]:

	brand	km_driven	selling_price	fuel_Diesel	fuel_LPG	fuel_Petrol	owner_Fourth & Above Owner	owner_Second Owner
0	Maruti	145500	450000	True	False	False	False	False
1	Skoda	120000	370000	True	False	False	False	True
2	Honda	140000	158000	False	False	True	False	False
3	Hyundai	127000	225000	True	False	False	False	False
4	Maruti	120000	130000	False	False	True	False	False
...	...	...	...	...	...	...	...	...
8123	Hyundai	110000	320000	False	False	True	False	False
8124	Hyundai	119000	135000	True	False	False	True	False
8125	Maruti	120000	382000	True	False	False	False	False
8126	Tata	25000	290000	True	False	False	False	False
8127	Tata	25000	290000	True	False	False	False	False

8128 rows × 10 columns

3. One hot encoding Using Sklearn

In [79]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(df.iloc[:,0:4],df.iloc[:,-1],test_si
```

In [80]:
```
df.head()
```

Out[80]:

	brand	km_driven	fuel	owner	selling_price
0	Maruti	145500	Diesel	First Owner	450000
1	Skoda	120000	Diesel	Second Owner	370000
2	Honda	140000	Petrol	Third Owner	158000
3	Hyundai	127000	Diesel	First Owner	225000
4	Maruti	120000	Petrol	First Owner	130000

In [81]:
```
X_train.head()
```

Out[81]:

	brand	km_driven	fuel	owner
5571	Hyundai	35000	Diesel	First Owner
2038	Jeep	60000	Diesel	First Owner
2957	Hyundai	25000	Petrol	First Owner
7618	Mahindra	130000	Diesel	Second Owner
6684	Hyundai	155000	Diesel	First Owner

In [82]:
```
X_test.head()
```

Out[82]:

	brand	km_driven	fuel	owner
606	Hyundai	80000	Petrol	First Owner
7575	Mahindra	70000	Diesel	Second Owner
7705	Toyota	68089	Petrol	First Owner
4305	Hyundai	70000	Petrol	Second Owner
2685	Mahindra	97000	Diesel	Second Owner

Import OHE: SK Learn

In [109]:
```
from sklearn.preprocessing import OneHotEncoder
```

In [110]:
```
ohe = OneHotEncoder(drop='first',sparse=False,dtype=np.int32)
```

In [111]:
```
X_train_new = ohe.fit_transform(X_train[['fuel','owner']]).toarray()
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\sklearn\preprocessing\_encoders.py:972: Fu
tureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be remo
ved in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default valu
e.
  warnings.warn(

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[111], line 1
----> 1 X_train_new = ohe.fit_transform(X_train[['fuel','owner']]).toarray()

AttributeError: 'numpy.ndarray' object has no attribute 'toarray'
```

In [112]:
```
X_test_new = ohe.transform(X_test[['fuel','owner']]).toarray()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[112], line 1
----> 1 X_test_new = ohe.transform(X_test[['fuel','owner']]).toarray()

AttributeError: 'numpy.ndarray' object has no attribute 'toarray'
```

In [113]:
```
X_train_new
```

Out[113]:
```
array([[1, 0, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       [0, 0, 1, ..., 0, 0, 0],
       ...,
       [0, 0, 1, ..., 0, 0, 0],
       [1, 0, 0, ..., 1, 0, 0],
       [1, 0, 0, ..., 0, 0, 0]])
```

In [114]:
```
X_train_new.shape
```

Out[114]:
```
(6502, 7)
```

In [116]:
```
np.hstack((X_train[['brand','km_driven']].values,X_train_new))
```

Out[116]:
```
array([['Hyundai', 35000, 1, ..., 0, 0, 0],
       ['Jeep', 60000, 1, ..., 0, 0, 0],
       ['Hyundai', 25000, 0, ..., 0, 0, 0],
       ...,
       ['Tata', 15000, 0, ..., 0, 0, 0],
       ['Maruti', 32500, 1, ..., 1, 0, 0],
       ['Isuzu', 121000, 1, ..., 0, 0, 0]], dtype=object)
```

In [ ]:

Import Library | pandas | numpy

In [1]:
```
import numpy as np
import pandas as pd
```

Import Datasets

In [2]:
```
df = pd.read_csv('customer.csv')
```

In [5]:
```
df.sample(5)
```

Out[5]:

	age	gender	review	education	purchased
40	39	Male	Good	School	No
14	15	Male	Poor	PG	Yes
41	23	Male	Good	PG	Yes
34	86	Male	Average	School	No
39	76	Male	Poor	PG	No

Eliminate Unnecessary Columns

In [6]:
```
df = df.iloc[:,2:]
```

In [7]:
```
df.sample(5)
```

Out[7]:

	review	education	purchased
45	Poor	PG	Yes
29	Average	UG	Yes
21	Average	PG	No
46	Poor	PG	No
27	Poor	PG	No

Train_Test_Split

In [10]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(df.iloc[:,0:2],df.iloc[:,-1:],test_s
```

Import_OrdinalEncoder

In [11]:
```
from sklearn.preprocessing import OrdinalEncoder
```

In [12]:
```
x_train
```

Out[12]:

	review	education
42	Good	PG
7	Poor	School
36	Good	UG
23	Good	School
0	Average	School
43	Poor	PG
3	Good	PG
49	Good	UG
13	Average	School
12	Poor	School
24	Average	PG
5	Average	School
28	Poor	School
21	Average	PG
4	Average	UG
10	Good	UG
33	Good	PG
31	Poor	School
18	Good	School
19	Poor	PG
44	Average	UG
1	Poor	UG
2	Good	PG
17	Poor	UG
41	Good	PG
30	Average	UG
47	Good	PG
26	Poor	PG
20	Average	School
22	Poor	PG
40	Good	School
14	Poor	PG
8	Average	UG
38	Good	School
6	Good	School
48	Good	UG
11	Good	UG
25	Good	School
29	Average	UG
27	Poor	PG

Define Categories

In [13]:
```
oe = OrdinalEncoder(categories=[['Poor','Average','Good'],['School','UG','PG']])
```

In [14]:
```
oe.fit(x_train)
```

Out[14]:
```
OrdinalEncoder(categories=[['Poor', 'Average', 'Good'], ['School', 'UG', 'PG']])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Transform Ordinal Encoding

In [15]:
```
x_train = oe.transform(x_train)
x_test = oe.transform(x_test)
```

In [16]:
```
x_train
```

Out[16]:
```
array([[2., 2.],
       [0., 0.],
       [2., 1.],
       [2., 0.],
       [1., 0.],
       [0., 2.],
       [2., 2.],
       [2., 1.],
       [1., 0.],
       [0., 0.],
       [1., 2.],
       [1., 0.],
       [0., 0.],
       [1., 2.],
       [1., 1.],
       [2., 1.],
       [2., 2.],
       [0., 0.],
       [2., 0.],
       [0., 2.],
       [1., 1.],
       [0., 1.],
       [2., 2.],
       [0., 1.],
       [2., 2.],
       [1., 1.],
       [2., 2.],
       [0., 2.],
       [1., 0.],
       [0., 2.],
       [2., 0.],
       [0., 2.],
       [1., 1.],
       [2., 0.],
       [2., 0.],
       [2., 1.],
       [2., 1.],
       [2., 0.],
       [1., 1.],
       [0., 2.]])
```

In [17]:
```
oe.categories_
```

Out[17]:
```
[array(['Poor', 'Average', 'Good'], dtype=object),
 array(['School', 'UG', 'PG'], dtype=object)]
```

Label Encoding

In [18]:
```
from sklearn.preprocessing import LabelEncoder
```

In [19]:
```
le = LabelEncoder()
```

In [20]:
```
le.fit(y_train)
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.py:97: DataCo
nversionWarning: A column-vector y was passed when a 1d array was expected. Please c
hange the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[20]:
```
LabelEncoder()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [21]:
```
le.classes_
```

Out[21]:
```
array(['No', 'Yes'], dtype=object)
```

In [22]:
```
y_train = le.transform(y_train)
y_test = le.transform(y_test)
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.py:132: DataC
onversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, dtype=self.classes_.dtype, warn=True)
C:\Users\ASUS\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.py:132: DataC
onversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, dtype=self.classes_.dtype, warn=True)
```

In [23]:
```
y_train
```

Out[23]:
```
array([1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0])
```

In [ ]:

Welcome to ML | Day 10

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Import Dataset

In [2]:
```
df = pd.read_csv('data_science_job.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	educatio
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment	G
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment	G
2	11561	city_21	0.624	NaN	No relevent experience	Full time course	G
3	33241	city_115	0.789	NaN	No relevent experience	NaN	G
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment	

Finding Missing Data Column Wise

In [4]:
```
df.isnull().mean()*100
```

Out[4]:
```
enrollee_id                0.000000
city                       0.000000
city_development_index     2.500261
gender                    23.530640
relevent_experience        0.000000
enrolled_university        2.014824
education_level            2.401086
major_discipline          14.683161
experience                 0.339284
company_size              30.994885
company_type              32.049274
training_hours             3.998330
target                     0.000000
dtype: float64
```

In [5]:
```
df.shape
```

Out[5]:
```
(19158, 13)
```

Selected column name (Below 5%)

In [6]:
```
cols = [var for var in df.columns if df[var].isnull().mean() < 0.05 and df[var].isnul
cols
```

Out[6]:
```
['city_development_index',
 'enrolled_university',
 'education_level',
 'experience',
 'training_hours']
```

In [7]:
```
df[cols].sample(5)
```

Out[7]:

	city_development_index	enrolled_university	education_level	experience	training_hours
9468	0.624	no_enrollment	Graduate	7.0	25.0
5762	0.920	Part time course	High School	2.0	47.0
6185	0.920	no_enrollment	Graduate	1.0	8.0
18266	NaN	no_enrollment	Masters	20.0	33.0
10011	0.920	no_enrollment	Masters	2.0	80.0

Calculated drop rows

In [8]:
```
len(df[cols].dropna()) / len(df)
```

Out[8]:
```
0.8968577095730244
```

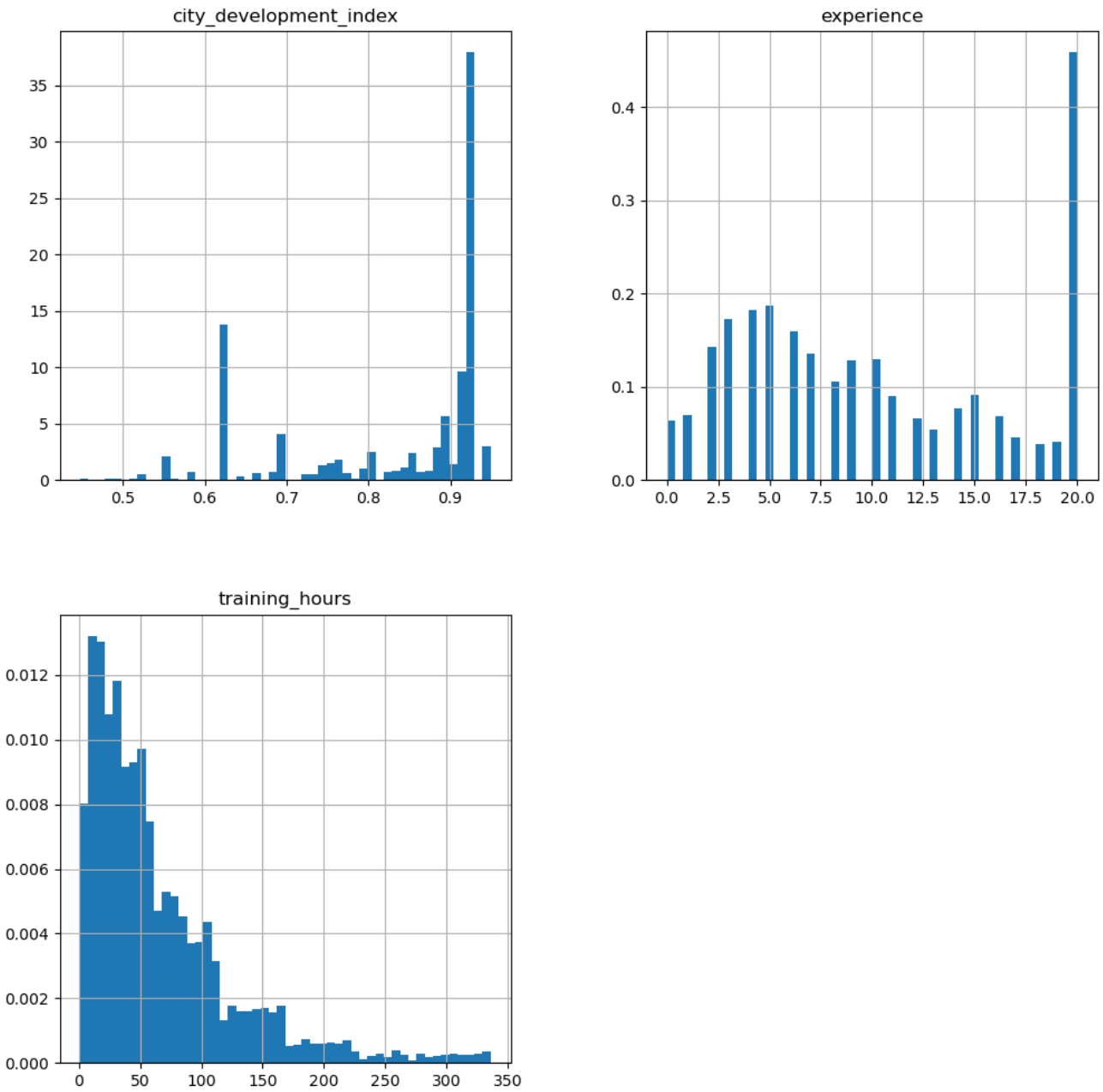
Create: New Data Frame

In [9]:
```
new_df = df[cols].dropna()
df.shape, new_df.shape
```

Out[9]:
```
((19158, 13), (17182, 5))
```

Plot Histogram (Applying CCA: Numerical Data)

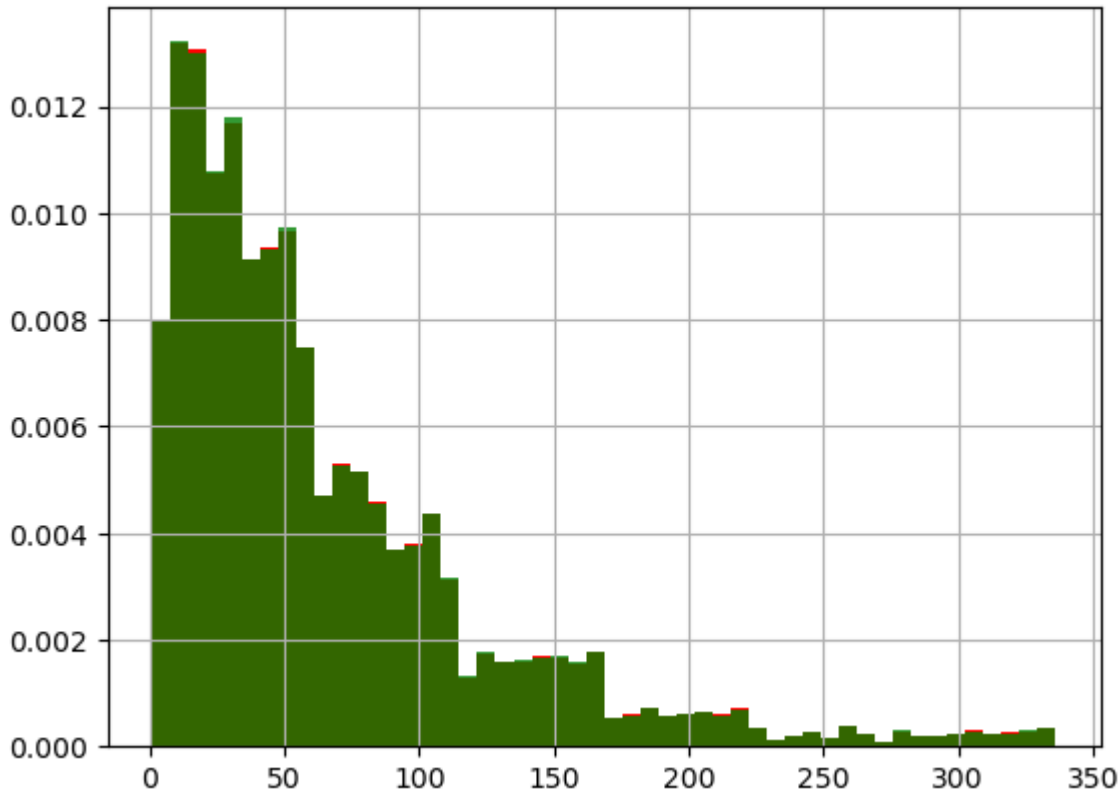
In [10]:
```
new_df.hist(bins=50, density=True, figsize=(12, 12))
plt.show()
```



Plot Histogram: Training Hours

In [12]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
# original data
df['training_hours'].hist(bins=50, ax=ax, density=True, color='red')
# data after cca, the argument alpha makes the color transparent, so we can
# see the overlay of the 2 distributions
new_df['training_hours'].hist(bins=50, ax=ax, color='green', density=True, alpha=0.8)
```

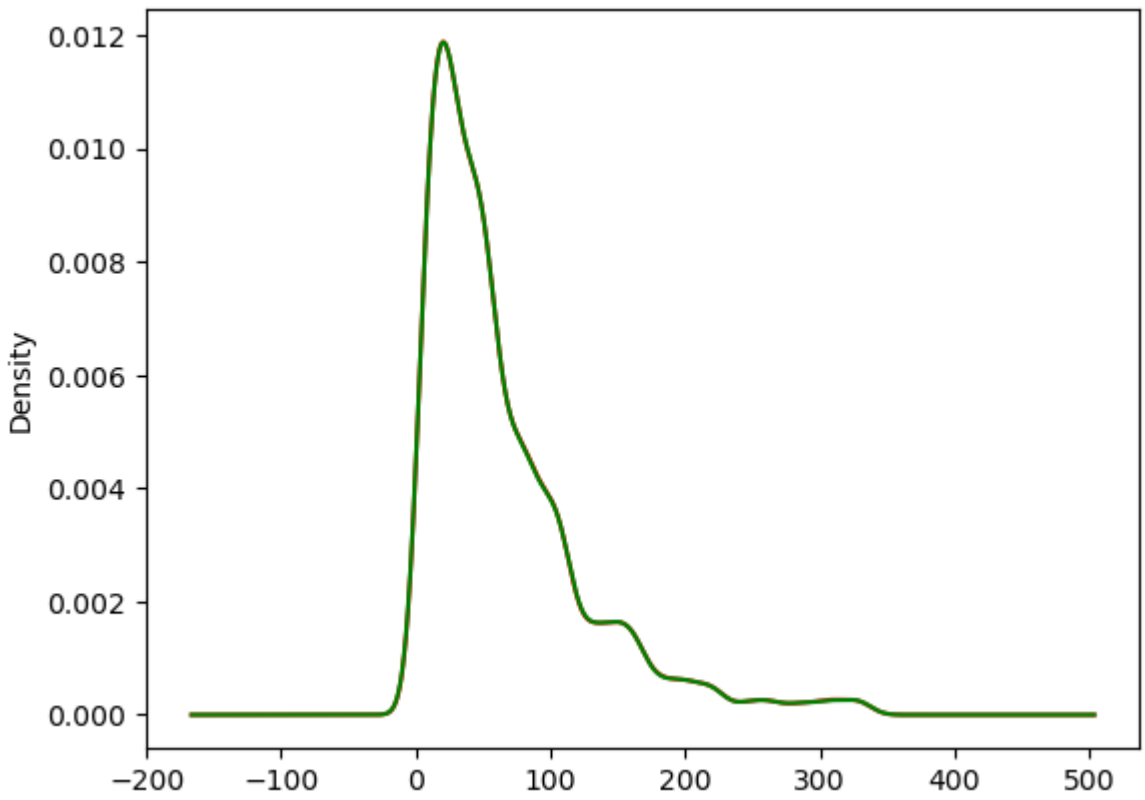
Out[12]: <Axes: >



Plot Probability Density Function (PDF): Training Hours

In [14]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
# original data
df['training_hours'].plot.density(color='red')
# data after cca
new_df['training_hours'].plot.density(color='green')
```

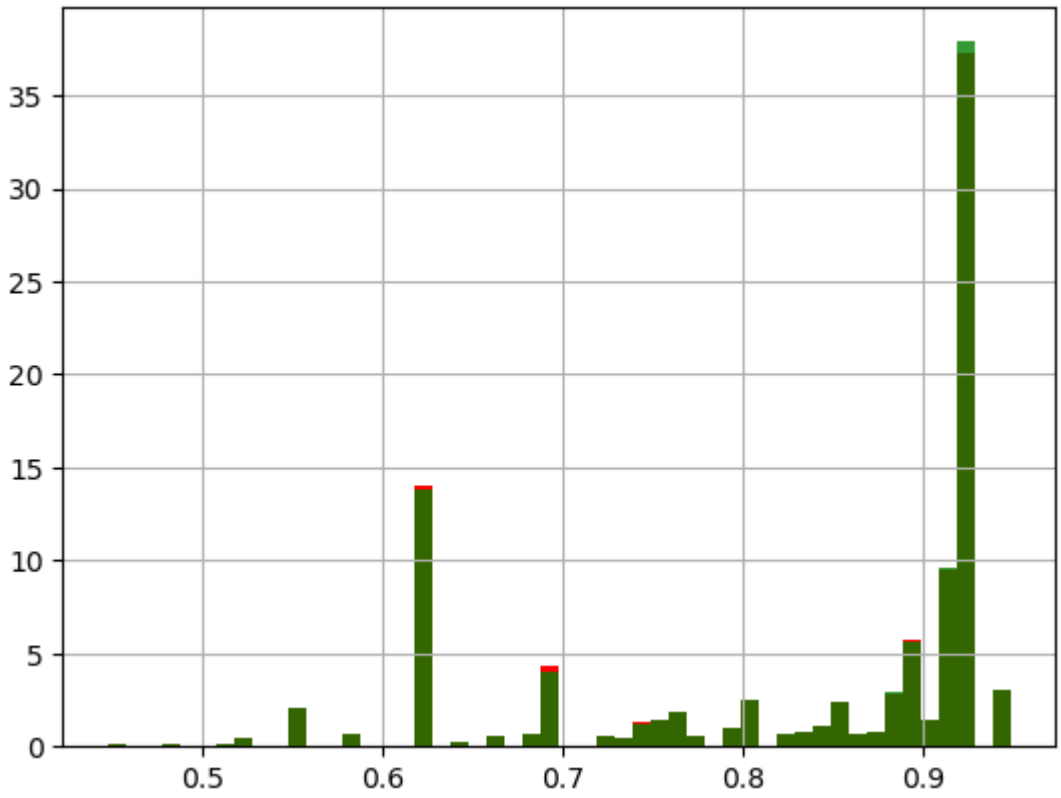
Out[14]: <Axes: ylabel='Density'>



Plot Histogram: City Development

In [15]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
# original data
df['city_development_index'].hist(bins=50, ax=ax, density=True, color='red')
# data after cca, the argument alpha makes the color transparent, so we can
# see the overlay of the 2 distributions
new_df['city_development_index'].hist(bins=50, ax=ax, color='green', density=True, al
```

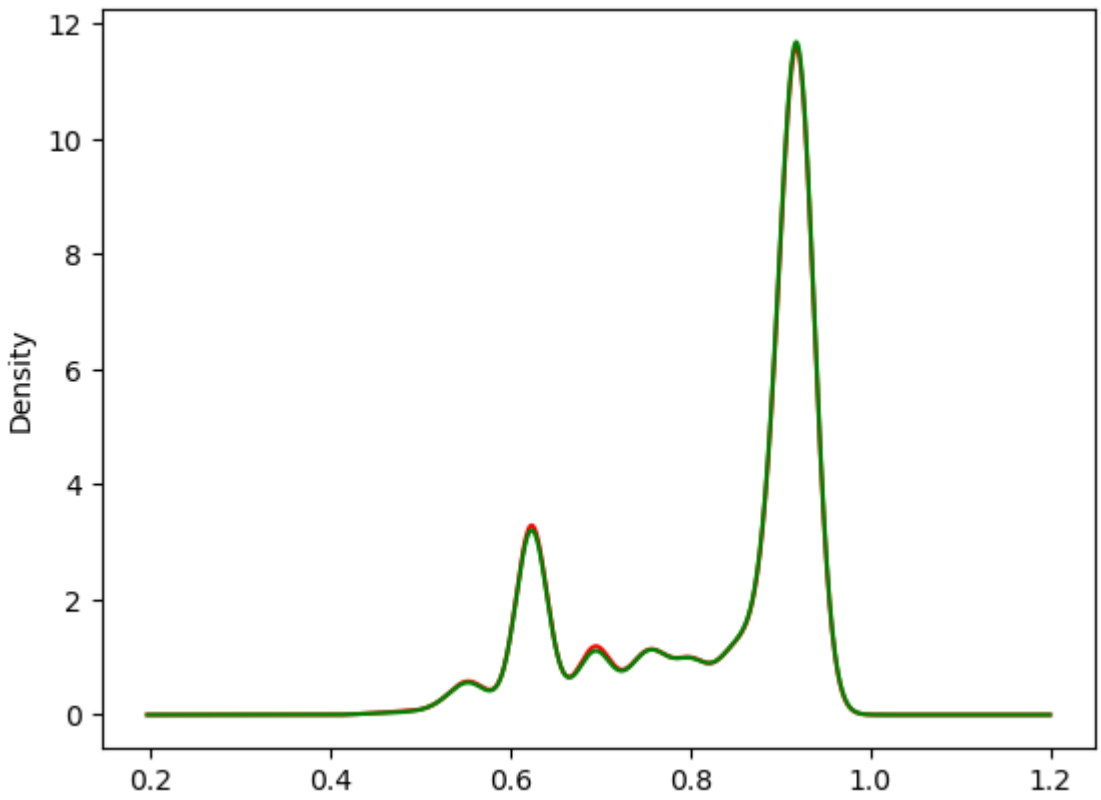
Out[15]: <Axes: >



Plot PDF: City Development

In [16]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
# original data
df['city_development_index'].plot.density(color='red')
# data after cca
new_df['city_development_index'].plot.density(color='green')
```

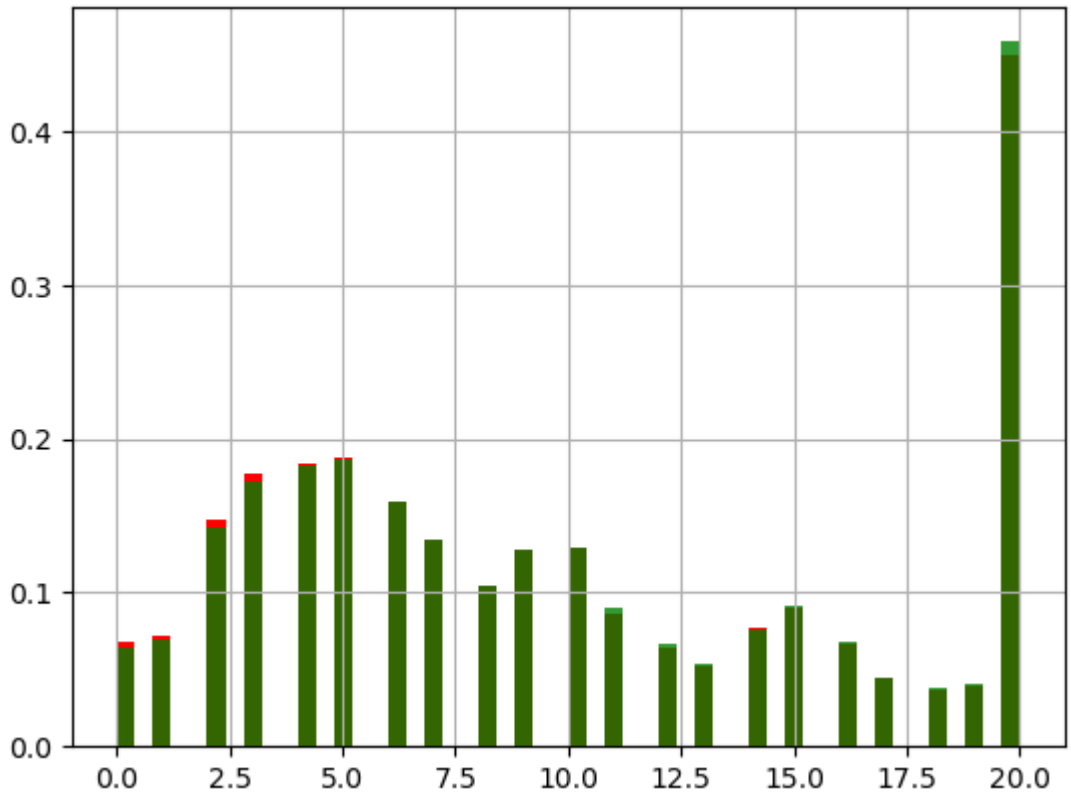
Out[16]: <Axes: ylabel='Density'>



Plot Histogram: Experience

In [17]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
# original data
df['experience'].hist(bins=50, ax=ax, density=True, color='red')
# data after cca, the argument alpha makes the color transparent, so we can
# see the overlay of the 2 distributions
new_df['experience'].hist(bins=50, ax=ax, color='green', density=True, alpha=0.8)
```

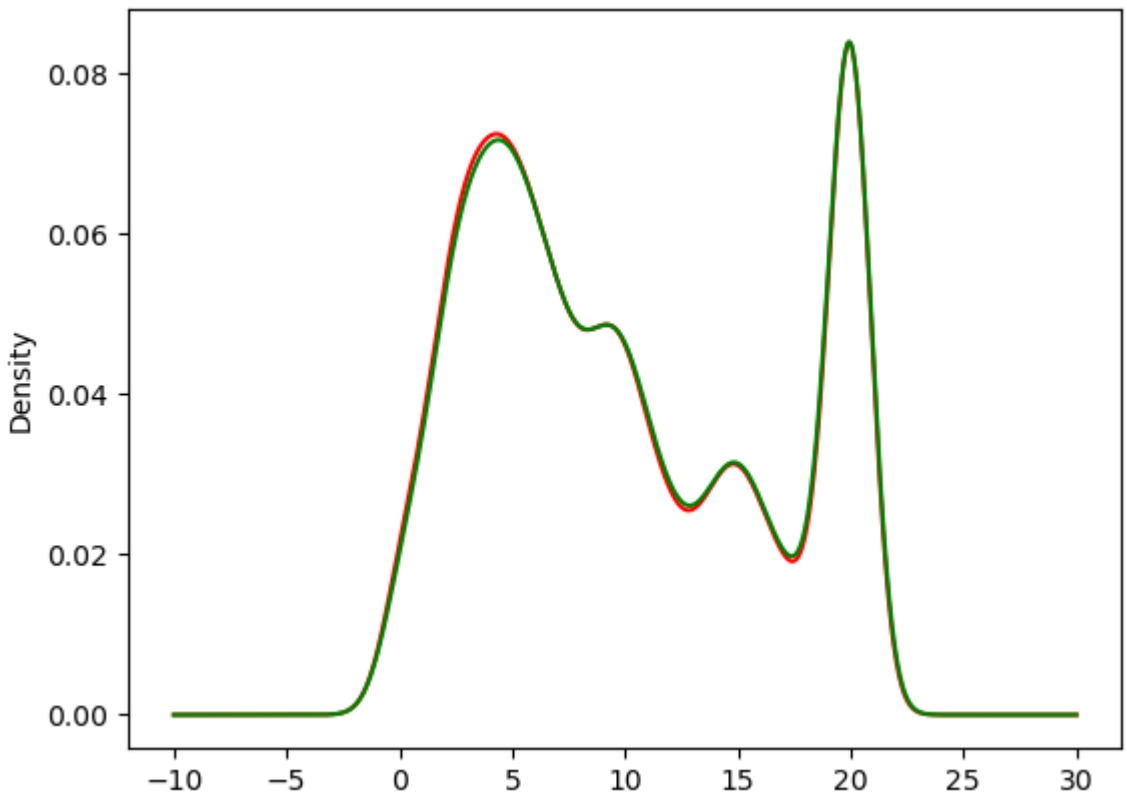
Out[17]: <Axes: >



Plot PDF: Experience

In [18]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
# original data
df['experience'].plot.density(color='red')
# data after cca
new_df['experience'].plot.density(color='green')
```

Out[18]:
```
<Axes: ylabel='Density'>
```



CCA in Categorical Data

In [19]:
```
df['education_level'].value_counts()
```

Out[19]:
```
education_level
Graduate          11598
Masters            4361
High School        2017
Phd                 414
Primary School      308
Name: count, dtype: int64
```

In [20]:
```
df['enrolled_university'].value_counts()
```

Out[20]:
```
enrolled_university
no_enrollment       13817
Full time course     3757
Part time course     1198
Name: count, dtype: int64
```

CCA Apply in Enrolled University

In [21]:
```
temp = pd.concat([
# percentage of observations per category, original data
df['enrolled_university'].value_counts() / len(df),
# percentage of observations per category, cca data
new_df['enrolled_university'].value_counts() / len(new_df)
],
axis=1)
# add column names
temp.columns = ['original', 'cca']
temp
```

Out[21]:

enrolled_university	original	cca
no_enrollment	0.721213	0.735188
Full time course	0.196106	0.200733
Part time course	0.062533	0.064079

CCA Apply in Education Level

In [22]:
```
temp = pd.concat([
# percentage of observations per category, original data
df['education_level'].value_counts() / len(df),
# percentage of observations per category, cca data
new_df['education_level'].value_counts() / len(new_df)
],
axis=1)
# add column names
temp.columns = ['original', 'cca']
temp
```

Out[22]:

education_level	original	cca
Graduate	0.605387	0.619835
Masters	0.227633	0.234082
High School	0.105282	0.107380
Phd	0.021610	0.022116
Primary School	0.016077	0.016587

In [ ]:

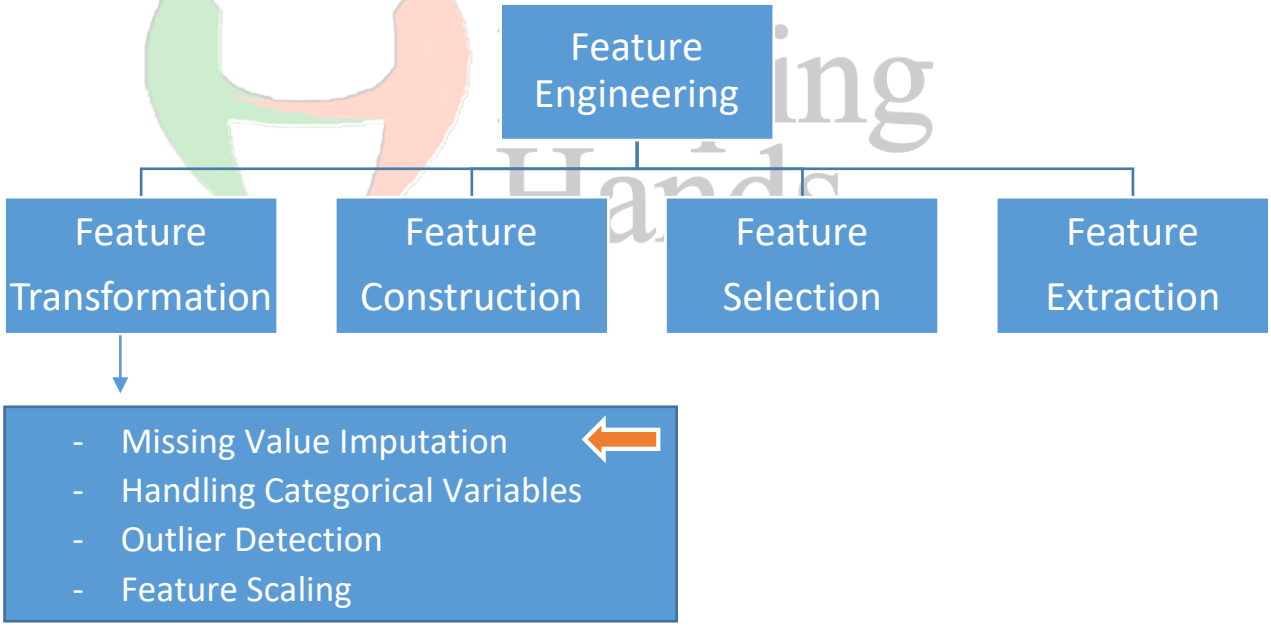
Data Science | 30 Days of Machine Learning | Day - 10

Educator Name: Nishant Dhote
Support Team: **+91-7880-113-112**

----Today Topics | Day 10----

- **Handling Missing Data**
- What are the problems with missing data?
- Remove Missing Values
- What is a missing completely at random (MCAR)?
- Pro and Corns CCA?
- When we use CCA?

Dataset Link GitHub: https://github.com/TheiScale/30_Days_Machine_Learning/



- What are the problems with missing data?

The real-world data often has a lot of missing values. The cause of missing values can be data corruption or failure to record data. The handling of

missing data is very important during the pre-processing of the dataset as many machine learning algorithms do not support missing values.

- Two Important ways to handle missing values in the dataset:

  1. Deleting Rows with missing values (Remove Missing Values)
  2. Impute missing values (Fill)
     - Univariate (Numerical & Categorical Removal)
     - Multivariate (KNN & Iterative Imputer)

- **Remove Missing Values**
  - CCA: Complete Case Analysis
    The standard treatment of missing data in most statistical packages is complete case analysis (CCA) done by case wise deletion. Any observation that has a missing value for any variable is automatically discarded and only complete observations are analysed.

    This means analysing only those observations for which all variables in the dataset have information.

  - Acceptance for CCA
    - What is a missing completely at random (MCAR)?
    - Pro and Corns CCA?
    - When we use CCA?

<Start Coding>

#Import Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

#Import Dataset

```
df = pd.read_csv('data_science_job.csv')
```

```
df.head()
```

#Finding Missing Data Column Wise

```
df.isnull().mean()*100
```

```
df.shape
```

#Selected column name (Below 5%)

```
cols = [var for var in df.columns if
df[var].isnull().mean() < 0.05 and
df[var].isnull().mean() > 0]
cols
```

```
df[cols].sample(5)
```

#Calculated drop rows

```
len(df[cols].dropna()) / len(df)
```

#Create: New Data Frame

```
new_df = df[cols].dropna()
df.shape, new_df.shape
```

#Plot Histogram (Before Applying CCA: Numerical Data)

```
new_df.hist(bins=50, density=True, figsize=(12,
12))
plt.show()
```

#Plot Histogram: Training Hours

```
fig = plt.figure()
ax = fig.add_subplot(111)

# original data
df['training_hours'].hist(bins=50, ax=ax,
density=True, color='red')

# data after cca, the argument alpha makes the
color transparent, so we can
# see the overlay of the 2 distributions
new_df['training_hours'].hist(bins=50, ax=ax,
color='green', density=True, alpha=0.8)
```

#Plot Probability Density Function (PDF): Training Hours

```
fig = plt.figure()
ax = fig.add_subplot(111)

# original data
df['training_hours'].plot.density(color='red')

# data after cca
new_df['training_hours'].plot.density(color='gree
n')
```

#Plot Histogram: City Development

```
fig = plt.figure()
ax = fig.add_subplot(111)

# original data
df['city_development_index'].hist(bins=50, ax=ax,
density=True, color='red')

# data after cca, the argument alpha makes the
color transparent, so we can
# see the overlay of the 2 distributions
new_df['city_development_index'].hist(bins=50,
ax=ax, color='green', density=True, alpha=0.8)
```

#Plot PDF: City Development

```
fig = plt.figure()
ax = fig.add_subplot(111)

# original data
df['city_development_index'].plot.density(color='r
ed')

# data after cca
new_df['city_development_index'].plot.density(colo
r='green')
```

#Plot Histogram: Experience

```
fig = plt.figure()
ax = fig.add_subplot(111)

# original data
df['experience'].hist(bins=50, ax=ax,
density=True, color='red')

# data after cca, the argument alpha makes the
color transparent, so we can
# see the overlay of the 2 distributions
new_df['experience'].hist(bins=50, ax=ax,
color='green', density=True, alpha=0.8)
```

#Plot PDF: Experience

```
fig = plt.figure()
ax = fig.add_subplot(111)

# original data
df['experience'].plot.density(color='red')

# data after cca
new_df['experience'].plot.density(color='green')
```

Industries Helping Hands

#CCA in Categorical Data

```
df['education_level'].value_counts()

---

df['enrolled_university'].value_counts()
```

#CCA Apply in Enrolled University

```
temp = pd.concat([
# percentage of observations per category,
original data

df['enrolled_university'].value_counts() /
len(df),

# percentage of observations per category, cca
data

new_df['enrolled_university'].value_counts() /
len(new_df)
        ],
        axis=1)

# add column names
temp.columns = ['original', 'cca']

temp
```

#CCA Apply in Education Level

```
temp = pd.concat([
 # percentage of observations per category,
original data
            df['education_level'].value_counts() /
len(df),

  # percentage of observations per category, cca
data

new_df['education_level'].value_counts() /
len(new_df)
        ],
        axis=1)

# add column names
temp.columns = ['original', 'cca']

temp
```

Day 10: Curious Data Minds

Dhanurjay "DJ" Patil

https://en.wikipedia.org/wiki/DJ_Patil

Dhanurjay "DJ" Patil (born August 3, 1974) is an American mathematician and computer scientist, White House announced Patil would be the first U.S. Chief Data Scientist

Dr. Patil public policy work includes being appointed by President Obama to be the first U.S. Chief Data Scientist where his efforts led to the establishment of nearly 40 Chief Data Officer roles across the Federal government.

Head of Data Products and Chief Scientist of LinkedIn

We're all products of failure.

While growing up in California, to simply say I was bad at Math would have been an understatement. My freshman year of high school, I was kicked out of my algebra class.

By the time high school graduation came around, I almost didn't graduate. For the record, I did actually graduate, but it was only because a very kind administrator took pity on me and changed my failing grade in chemistry to a passing one.

Industries Helping Hands

Day 11 | Mean & Median imputation

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Import sklearn Libraries

In [3]:
```
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
```

Import Dataset

In [4]:
```
df = pd.read_csv('titanic_toy.csv')
```

In [10]:
```
df
```

Out[10]:

	Age	Fare	Family	Survived
0	22.0	7.2500	1	0
1	38.0	71.2833	1	1
2	26.0	7.9250	0	1
3	35.0	53.1000	1	1
4	35.0	8.0500	0	0
...	...	...	...	...
886	27.0	13.0000	0	0
887	19.0	30.0000	0	1
888	NaN	23.4500	3	0
889	26.0	NaN	0	1
890	32.0	7.7500	0	0

891 rows × 4 columns

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Age       714 non-null    float64
 1   Fare      846 non-null    float64
 2   Family    891 non-null    int64
 3   Survived  891 non-null    int64
dtypes: float64(2), int64(2)
memory usage: 28.0 KB
```

Perform Train Test Split

In [11]:
```
X = df.drop(columns=['Survived'])
y = df['Survived']
```

In [12]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [13]:
```
X_train.shape, X_test.shape
```

Out[13]:
```
((712, 3), (179, 3))
```

In [14]:
```
X_train.isnull().mean()
```

Out[14]:
```
Age       0.207865
Fare      0.050562
Family    0.000000
dtype: float64
```

Calculate Mean and Median (Age | Fare)

In [15]:
```
mean_age = X_train['Age'].mean()
median_age = X_train['Age'].median()
mean_fare = X_train['Fare'].mean()
median_fare = X_train['Fare'].median()
```

Create new column & impute missing values

In [17]:
```
X_train['Age_median'] = X_train['Age'].fillna(median_age)
X_train['Age_mean'] = X_train['Age'].fillna(mean_age)
X_train['Fare_median'] = X_train['Fare'].fillna(median_fare)
X_train['Fare_mean'] = X_train['Fare'].fillna(mean_fare
```

In [18]:
```
X_train.sample(8)
```

Out[18]:

	Age	Fare	Family	Age_median	Age_mean	Fare_median	Fare_mean
255	29.0	15.2458	2	29.00	29.000000	15.2458	15.245800
420	NaN	7.8958	0	28.75	29.785904	7.8958	7.895800
673	31.0	NaN	0	31.00	31.000000	14.4583	32.617597
761	41.0	7.1250	0	41.00	41.000000	7.1250	7.125000
82	NaN	NaN	0	28.75	29.785904	14.4583	32.617597
269	35.0	135.6333	0	35.00	35.000000	135.6333	135.633300
254	41.0	20.2125	2	41.00	41.000000	20.2125	20.212500
440	45.0	26.2500	2	45.00	45.000000	26.2500	26.250000

Review Variance

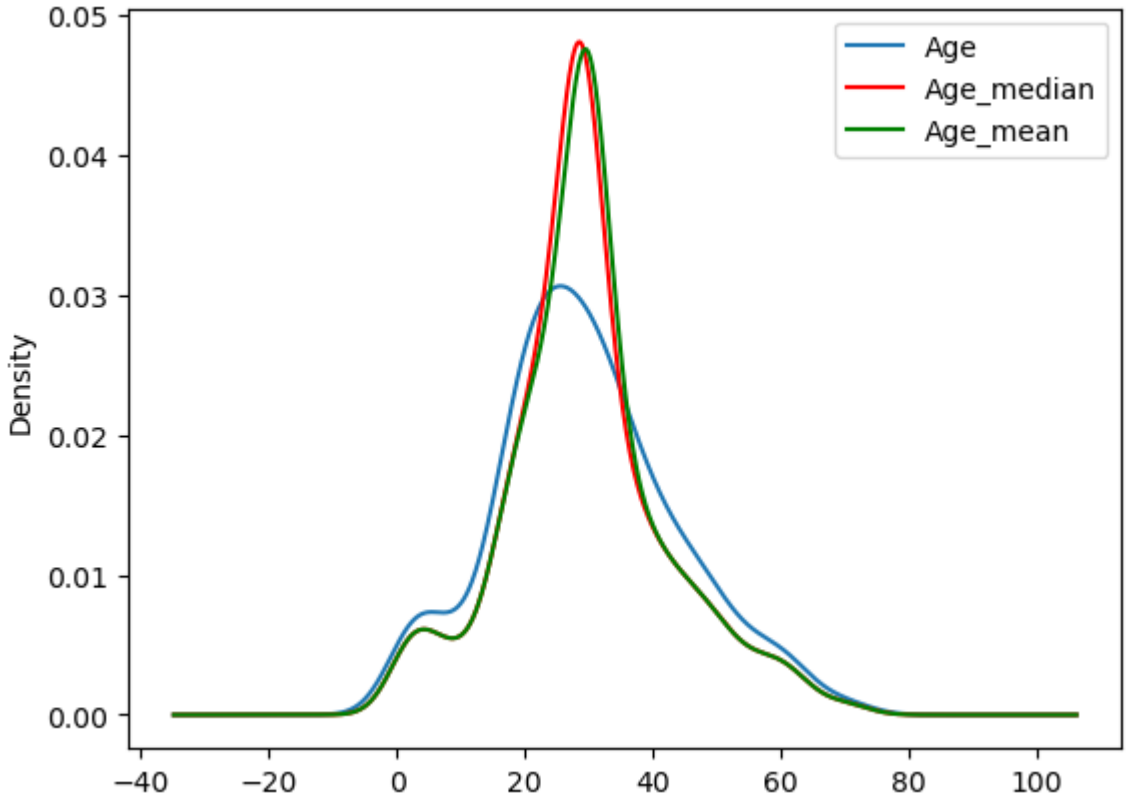
In [19]:
```
print('Original Age variable variance: ', X_train['Age'].var())
print('Age Variance after median imputation: ', X_train['Age_median'].var())
print('Age Variance after mean imputation: ', X_train['Age_mean'].var())

print('Original Fare variable variance: ', X_train['Fare'].var())
print('Fare Variance after median imputation: ', X_train['Fare_median'].var())
print('Fare Variance after mean imputation: ', X_train['Fare_mean'].var())
```

```
Original Age variable variance:  204.3495133904614
Age Variance after median imputation:  161.9895663346054
Age Variance after mean imputation:  161.81262452718673
Original Fare variable variance:  2448.197913706318
Fare Variance after median imputation:  2340.0910219753637
Fare Variance after mean imputation:  2324.2385256705547
```

Changes in Distribution in Age

In [20]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
# original variable distribution
X_train['Age'].plot(kind='kde', ax=ax)
# variable imputed with the median
X_train['Age_median'].plot(kind='kde', ax=ax, color='red')
# variable imputed with the mean
X_train['Age_mean'].plot(kind='kde', ax=ax, color='green')
# add legends
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best')
```

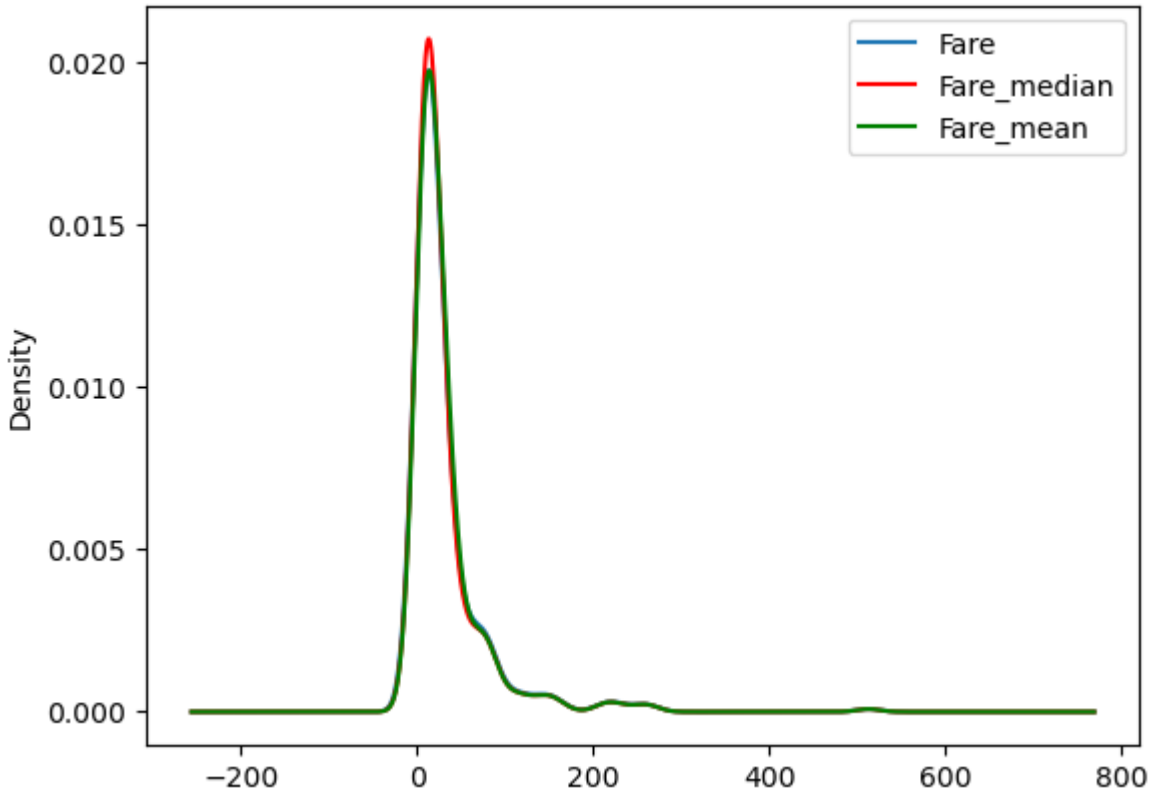
Out[20]:
```
<matplotlib.legend.Legend at 0x2bca7545e90>
```



Changes in Distribution in Fare

In [21]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
# original variable distribution
X_train['Fare'].plot(kind='kde', ax=ax)
# variable imputed with the median

X_train['Fare_median'].plot(kind='kde', ax=ax, color='red')
# variable imputed with the mean
X_train['Fare_mean'].plot(kind='kde', ax=ax, color='green')
# add legends
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best')
```

Out[21]:
```
<matplotlib.legend.Legend at 0x2bca45f4490>
```



Check Covariance

In [22]:
```
X_train.cov()
```

Out[22]:

	Age	Fare	Family	Age_median	Age_mean	Fare_median	Fare_mean
Age	204.349513	70.719262	-6.498901	204.349513	204.349513	64.858859	66.665205
Fare	70.719262	2448.197914	17.258917	57.957599	55.603719	2448.197914	2448.197914
Family	-6.498901	17.258917	2.735252	-5.112563	-5.146106	16.476305	16.385048
Age_median	204.349513	57.957599	-5.112563	161.989566	161.812625	53.553455	55.023037
Age_mean	204.349513	55.603719	-5.146106	161.812625	161.812625	51.358000	52.788341
Fare_median	64.858859	2448.197914	16.476305	53.553455	51.358000	2340.091022	2324.238526
Fare_mean	66.665205	2448.197914	16.385048	55.023037	52.788341	2324.238526	2324.238526

Check Correlation

In [24]:
```
X_train.corr()
```

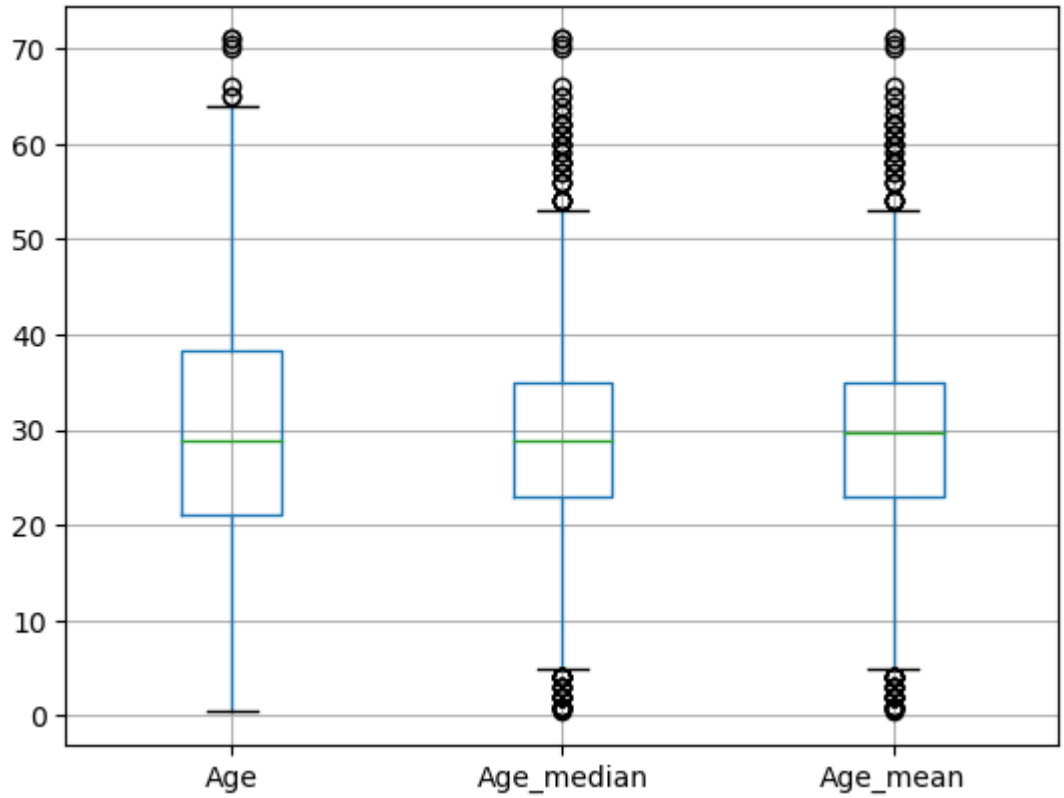
Out[24]:

	Age	Fare	Family	Age_median	Age_mean	Fare_median	Fare_mean
Age	1.000000	0.092644	-0.299113	1.000000	1.000000	0.087356	0.090156
Fare	0.092644	1.000000	0.208268	0.091757	0.088069	1.000000	1.000000
Family	-0.299113	0.208268	1.000000	-0.242883	-0.244610	0.205942	0.205499
Age_median	1.000000	0.091757	-0.242883	1.000000	0.999454	0.086982	0.089673
Age_mean	1.000000	0.088069	-0.244610	0.999454	1.000000	0.083461	0.086078
Fare_median	0.087356	1.000000	0.205942	0.086982	0.083461	1.000000	0.996607
Fare_mean	0.090156	1.000000	0.205499	0.089673	0.086078	0.996607	1.000000

Box Plot for Age

In [25]:
```
X_train[['Age', 'Age_median',
'Age_mean']].boxplot()
```

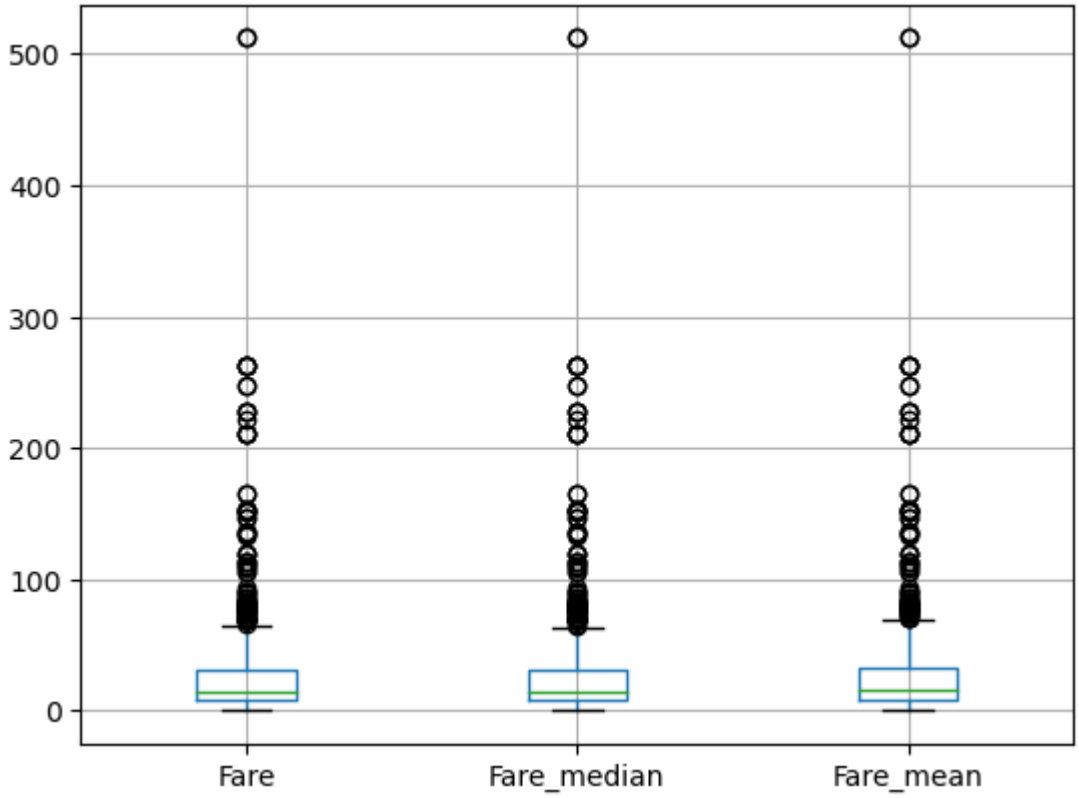
Out[25]: <Axes: >



Box Plot for Fare

In [26]:
```
X_train[['Fare', 'Fare_median', 'Fare_mean']].boxplot()
```

Out[26]: <Axes: >



In [ ]:

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Import SK learn

In [9]:
```
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
```

Import Dataset

In [10]:
```
df = pd.read_csv('titanic_toy.csv')
```

In [11]:
```
df
```

Out[11]:

	Age	Fare	Family	Survived
0	22.0	7.2500	1	0
1	38.0	71.2833	1	1
2	26.0	7.9250	0	1
3	35.0	53.1000	1	1
4	35.0	8.0500	0	0
...	...	...	...	...
886	27.0	13.0000	0	0
887	19.0	30.0000	0	1
888	NaN	23.4500	3	0
889	26.0	NaN	0	1
890	32.0	7.7500	0	0

891 rows × 4 columns

In [12]:
```
df.head()
```

Out[12]:

	Age	Fare	Family	Survived
0	22.0	7.2500	1	0
1	38.0	71.2833	1	1
2	26.0	7.9250	0	1
3	35.0	53.1000	1	1
4	35.0	8.0500	0	0

In [13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Age       714 non-null    float64
 1   Fare      846 non-null    float64
 2   Family    891 non-null    int64
 3   Survived  891 non-null    int64
dtypes: float64(2), int64(2)
memory usage: 28.0 KB
```

Perform Train Test Split

In [18]:
```
X = df.drop(columns=['Survived'])
y = df['Survived']
```

In [19]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [20]:
```
X_train['Age_99'] = X_train['Age'].fillna(99)
X_train['Age_minus1'] = X_train['Age'].fillna(-1)
X_train['Fare_999'] = X_train['Fare'].fillna(999)
X_train['Fare_minus1'] = X_train['Fare'].fillna(-
1)
```

Create New Column and Replace Value (Age-99 | Fare-999)

In [21]:
```
X_train['Age_99'] = X_train['Age'].fillna(99)
X_train['Age_minus1'] = X_train['Age'].fillna(-1)
X_train['Fare_999'] = X_train['Fare'].fillna(999)
X_train['Fare_minus1'] = X_train['Fare'].fillna(-1)
```

In [23]:
```
X_train.sample(10)
```

Out[23]:

	Age	Fare	Family	Age_99	Age_minus1	Fare_999	Fare_minus1
861	21.0	11.5000	1	21.0	21.0	11.5000	11.5000
428	NaN	7.7500	0	99.0	-1.0	7.7500	7.7500
97	23.0	63.3583	1	23.0	23.0	63.3583	63.3583
31	NaN	146.5208	1	99.0	-1.0	146.5208	146.5208
378	20.0	NaN	0	20.0	20.0	999.0000	-1.0000
358	NaN	7.8792	0	99.0	-1.0	7.8792	7.8792
323	22.0	29.0000	2	22.0	22.0	29.0000	29.0000
121	NaN	8.0500	0	99.0	-1.0	8.0500	8.0500
165	9.0	20.5250	2	9.0	9.0	20.5250	20.5250
494	21.0	8.0500	0	21.0	21.0	8.0500	8.0500

Review Variance

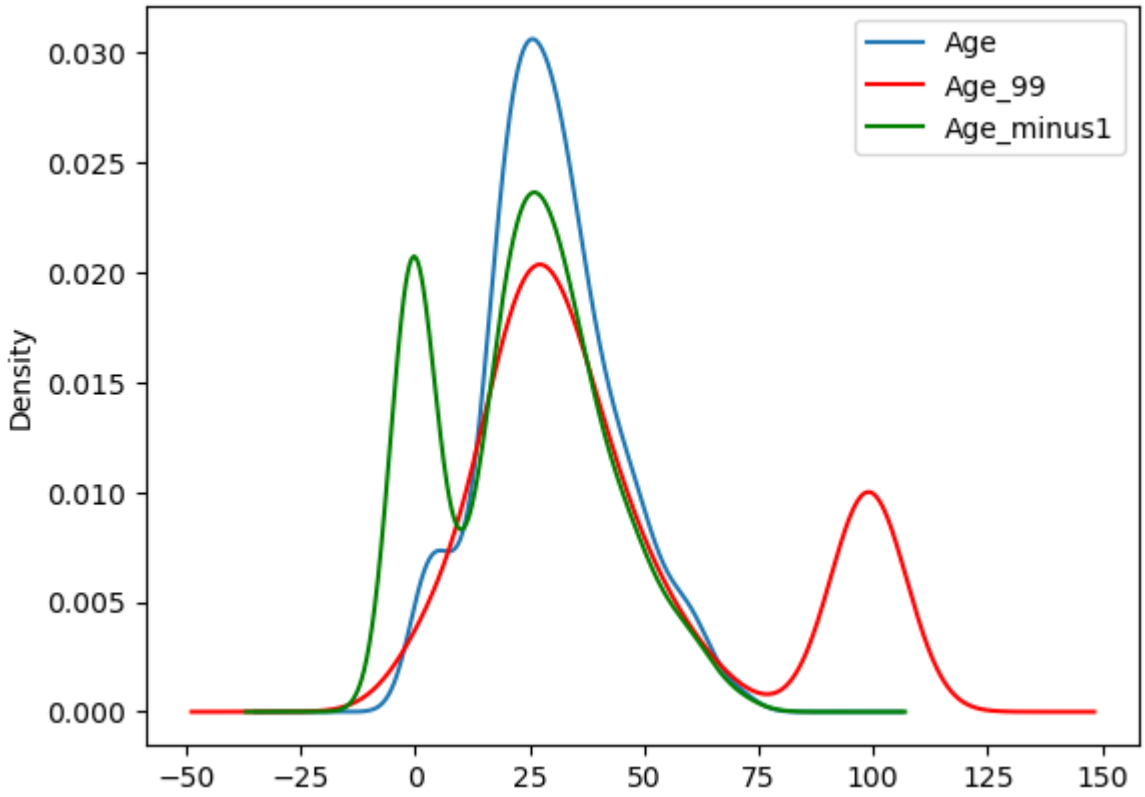
In [27]:
```
print('Original Age variable variance: ', X_train['Age'].var())
print('Age Variance after 99 use imputation: ', X_train['Age_99'].var())
print('Age Variance after -1 use imputation: ', X_train['Age_minus1'].var())
print('Original Fare variable variance: ', X_train['Fare'].var())
print('Fare Variance after 999 use imputation: ', X_train['Fare_999'].var())
print('Fare Variance after -1 use imputation: ', X_train['Fare_minus1'].var())
```

```
Original Age variable variance:  204.3495133904614
Age Variance after 99 use imputation:  951.7275570187172
Age Variance after -1 use imputation:  318.0896202624484
Original Fare variable variance:  2448.197913706318
Fare Variance after 999 use imputation:  47219.20265217623
Fare Variance after -1 use imputation:  2378.5676784883503
```

Review Distribution - Age:

In [29]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
# original variable distribution
X_train['Age'].plot(kind='kde', ax=ax)
# variable imputed with the median
X_train['Age_99'].plot(kind='kde', ax=ax, color='red')
# variable imputed with the mean
X_train['Age_minus1'].plot(kind='kde', ax=ax, color='green')
# add legends
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best')
```

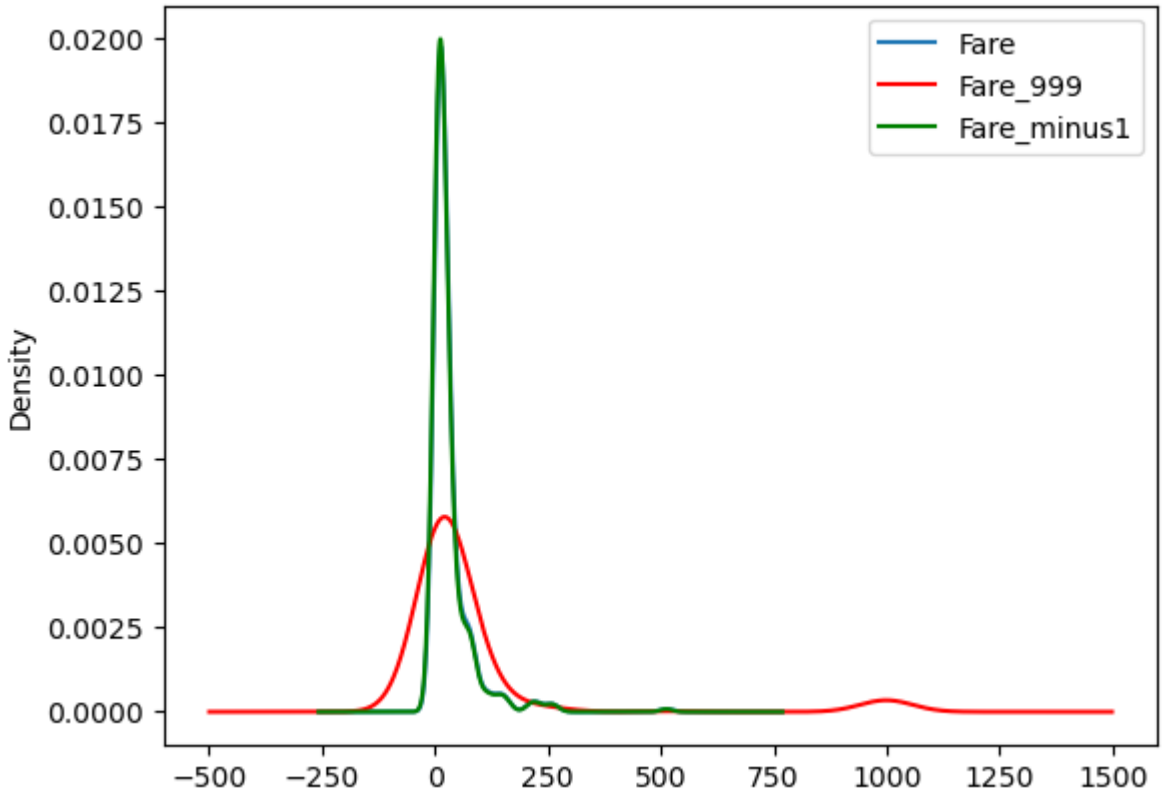
Out[29]:
```
<matplotlib.legend.Legend at 0x200c9e32ad0>
```



Review Distribution –Fare:

In [30]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
# original variable distribution
X_train['Fare'].plot(kind='kde', ax=ax)
# variable imputed with the median
X_train['Fare_999'].plot(kind='kde', ax=ax, color='red')
# variable imputed with the mean

X_train['Fare_minus1'].plot(kind='kde', ax=ax, color='green')
# add legends
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best')
```

Out[30]: <matplotlib.legend.Legend at 0x200c9147090>



Check Covariance

In [31]:
```
X_train.cov()
```

Out[31]:

	Age	Fare	Family	Age_99	Age_minus1	Fare_999	Fare_minus1
Age	204.349513	70.719262	-6.498901	204.349513	204.349513	162.793430	63.321188
Fare	70.719262	2448.197914	17.258917	-101.671097	125.558364	2448.197914	2448.197914
Family	-6.498901	17.258917	2.735252	-7.387287	-4.149246	11.528625	16.553989
Age_99	204.349513	-101.671097	-7.387287	951.727557	-189.535540	-159.931663	-94.317400
Age_minus1	204.349513	125.558364	-4.149246	-189.535540	318.089620	257.379887	114.394141
Fare_999	162.793430	2448.197914	11.528625	-159.931663	257.379887	47219.202652	762.474982
Fare_minus1	63.321188	2448.197914	16.553989	-94.317400	114.394141	762.474982	2378.567678

Check Correlation

In [32]:
```
X_train.corr()
```

Out[32]:

	Age	Fare	Family	Age_99	Age_minus1	Fare_999	Fare_minus1
Age	1.000000	0.092644	-0.299113	1.000000	1.000000	0.051179	0.084585
Fare	0.092644	1.000000	0.208268	-0.066273	0.142022	1.000000	1.000000
Family	-0.299113	0.208268	1.000000	-0.144787	-0.140668	0.032079	0.205233
Age_99	1.000000	-0.066273	-0.144787	1.000000	-0.344476	-0.023857	-0.062687
Age_minus1	1.000000	0.142022	-0.140668	-0.344476	1.000000	0.066411	0.131514
Fare_999	0.051179	1.000000	0.032079	-0.023857	0.066411	1.000000	0.071946
Fare_minus1	0.084585	1.000000	0.205233	-0.062687	0.131514	0.071946	1.000000

In [ ]:

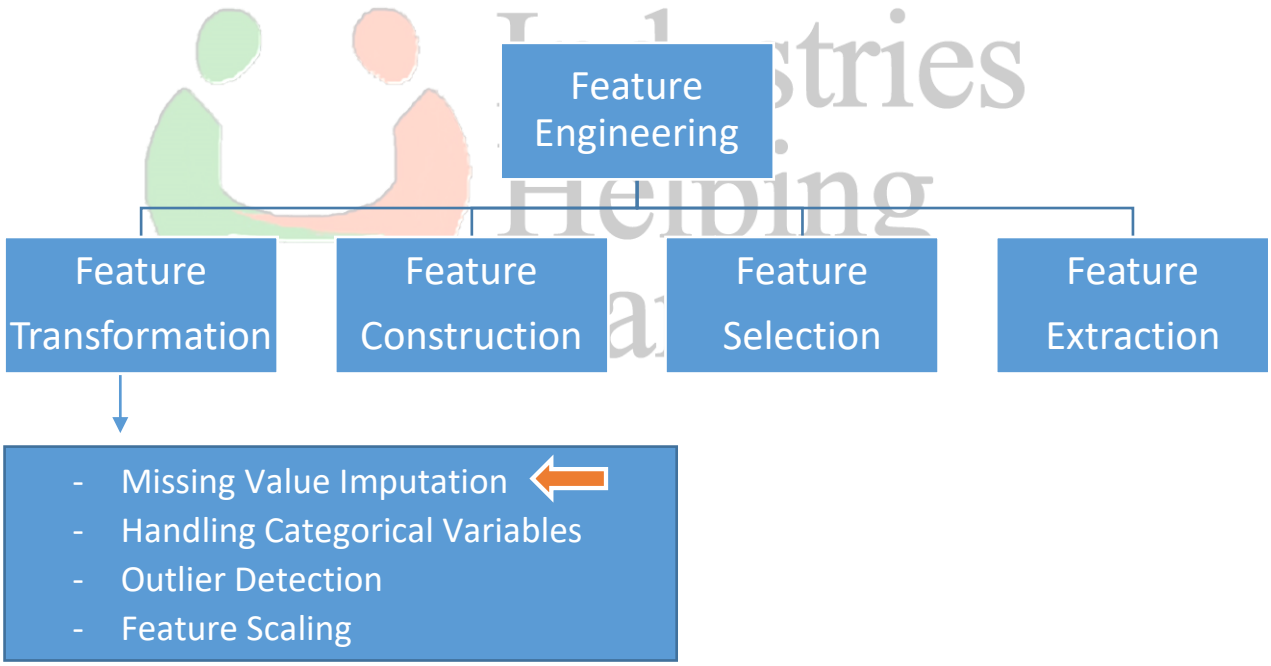
Data Science | 30 Days of Machine Learning | Day - 11

Educator Name: Nishant Dhote
Support Team: **+91-7880-113-112**

----Today Topics | Day 11----

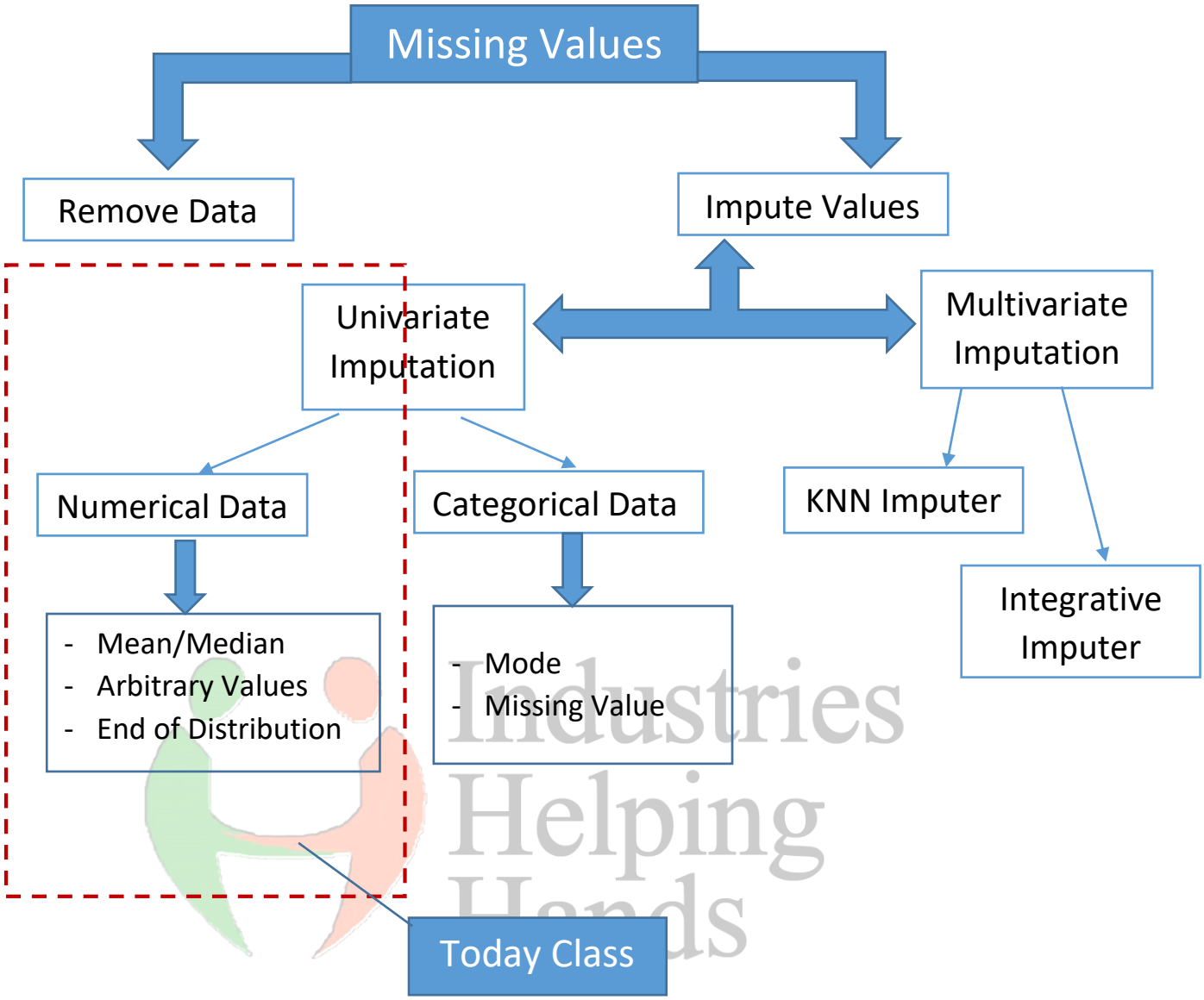
- Univariate Imputation in Numerical Data
- Difference between univariate imputation and multivariate imputation
- Mean or Median Imputation
- Arbitrary Value Imputation
- End of Distribution
- Random value Imputation

Dataset Link GitHub: https://github.com/TheiScale/30_Days_Machine_Learning/



- Two Important ways to handle missing values in the dataset:

  1. Deleting rows with missing values (Remove Missing Values)
  2. Impute missing values (Fill)
     - Univariate (Numerical & Categorical Removal)
     - Multivariate (KNN & Iterative Imputer)



Univariate Imputation in Numerical Data

What is the difference between univariate imputation and multivariate imputation?

Univariate imputation implies that we are only considering the values of a single column when performing imputation. Multivariate imputation, on the other hand, involves taking into account other features in the dataset when performing imputation.

Univariate Imputation

C1	C2	C3	C4	C5
	--			

Multivariate Imputation

C1	C2	C3	C4	C5
	--			

We cover below topics

- Mean/Median
- Arbitrary Values
- End of Distribution

Mean or Median Imputation-

Mean imputation (MI) is one such method in which the mean of the observed values for each variable is computed and the missing values for that variable are imputed by this mean.

Mean / Median imputation: example

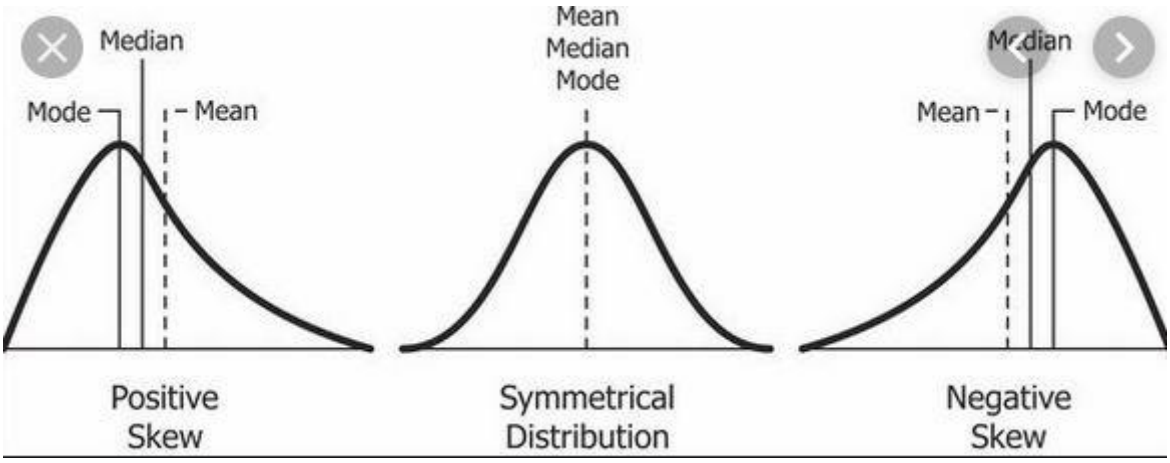
Price
100
90
50
40
20
100
60
120
200

Mean = 86.66

Median = 90

Price
100
90
50
40
20
100
86.66
60
120
86.66
200

When will we use mean value and median value?



Average	Advantages	Disadvantages
Mean	All the data is used to find the answer	Very large or very small numbers can distort the answer
Median	Very big and very small values don't affect it	Takes a long time to calculate for a very large set of data
Mode or modal class	The only average we can use when the data is not numerical	1. There may be more than one mode 2. There may be no mode at all if none of the data is the same 3. It may not accurately represent the data

<Start Coding | Mean – Median - imputation>

#Import Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

#Import sklearn Libraries

```
from sklearn.model_selection import
train_test_split
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
```

#Import Dataset

```
df = pd.read_csv('titanic_toy.csv')
```

```
df.head()
```

```
df.info(
```

#Perform Train Test Split

```
X = df.drop(columns=['Survived'])
y = df['Survived']
```

```
X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.2,random_state=2
)
```

```
X_train.shape, X_test.shape
```

```
X_train.isnull().mean()
```

#Calculate Mean and Median (Age | Fare)

```
mean_age = X_train['Age'].mean()
median_age = X_train['Age'].median()

mean_fare = X_train['Fare'].mean()
median_fare = X_train['Fare'].median()
```

#Create new column & impute missing values

```
X_train['Age_median'] =
X_train['Age'].fillna(median_age)
X_train['Age_mean'] =
X_train['Age'].fillna(mean_age)

X_train['Fare_median'] =
X_train['Fare'].fillna(median_fare)
X_train['Fare_mean'] =
X_train['Fare'].fillna(mean_fare

---
X_train.sample(8)
```

#Review Variance

```
print('Original Age variable variance: ',
X_train['Age'].var())
print('Age Variance after median imputation: ',
X_train['Age_median'].var())
print('Age Variance after mean imputation: ',
X_train['Age_mean'].var())
```

```
print('Original Fare variable variance: ',
X_train['Fare'].var())
print('Fare Variance after median imputation: ',
X_train['Fare_median'].var())
print('Fare Variance after mean imputation: ',
X_train['Fare_mean'].var())
```

#Changes in Distribution in Age

```
fig = plt.figure()
ax = fig.add_subplot(111)

# original variable distribution
X_train['Age'].plot(kind='kde', ax=ax)

# variable imputed with the median
X_train['Age_median'].plot(kind='kde', ax=ax,
color='red')

# variable imputed with the mean
X_train['Age_mean'].plot(kind='kde', ax=ax,
color='green')

# add legends
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best')
```

#Changes in Distribution in Fare

```
fig = plt.figure()
ax = fig.add_subplot(111)

# original variable distribution
X_train['Fare'].plot(kind='kde', ax=ax)

# variable imputed with the median
```

```
X_train['Fare_median'].plot(kind='kde', ax=ax,
color='red')

# variable imputed with the mean
X_train['Fare_mean'].plot(kind='kde', ax=ax,
color='green')

# add legends
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best')
```

#Check Covariance

```
X_train.cov()
```

#Check Correlation

```
X_train.corr()
```

#Box Plot for Age

```
X_train[['Age', 'Age_median',
'Age_mean']].boxplot()
```

#Box Plot for Fare

```
X_train[['Fare', 'Fare_median',
'Fare_mean']].boxplot()
```

Industries Helping Hands

- **Arbitrary Value Imputation:** Arbitrary value imputation consists of replacing all occurrences of missing values within a variable with an arbitrary value.

For Categorical Data-

Using the word 'Missing' as a category to replace the missing values

ID	City	Degree	Married ?
1	Lisbon	NaN	0
2	Berlin	Bachelor	1
3	Lisbon	NaN	1
4	Lisbon	Bachelor	1
5	Berlin	Bachelor	0
6	Lisbon	Bachelor	0
7	Berlin	Masters	1
8	Berlin	No Degree	0
9	Berlin	Masters	1
10	Madrid	Masters	1

ID	City	Degree	Married ?
1	Lisbon	Missing	0
2	Berlin	Bachelor	1
3	Lisbon	Missing	1
4	Lisbon	Bachelor	1
5	Berlin	Bachelor	0
6	Lisbon	Bachelor	0
7	Berlin	Masters	1
8	Berlin	No Degree	0
9	Berlin	Masters	1
10	Madrid	Masters	1

For Numerical Data-

Arbitrary Value = -1

ID	City	Age	Married ?
1	Lisbon	25	0
2	Berlin	25	1
3	Lisbon	30	1
4	Lisbon	30	1
5	Berlin	18	0
6	Lisbon	NaN	0
7	Berlin	30	1
8	Berlin	NaN	0
9	Berlin	25	1
10	Madrid	25	1

ID	City	Age	Married ?
1	Lisbon	25	0
2	Berlin	25	1
3	Lisbon	30	1
4	Lisbon	30	1
5	Berlin	18	0
6	Lisbon	-1	0
7	Berlin	30	1
8	Berlin	-1	0
9	Berlin	25	1
10	Madrid	25	1

<Start Coding | Arbitrary Value Imputation >

#Import Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

#Import Dataset

```
df = pd.read_csv('titanic_toy.csv')
```

```
df.head()
```

```
df.info(
```

#Perform Train Test Split

```
X = df.drop(columns=['Survived'])
y = df['Survived']
```

```
X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.2,random_state=2
)
```

```
X_train['Age_99'] = X_train['Age'].fillna(99)
X_train['Age_minus1'] = X_train['Age'].fillna(-1)

X_train['Fare_999'] = X_train['Fare'].fillna(999)
X_train['Fare_minus1'] = X_train['Fare'].fillna(-
1)
```

#Create New Column and Replace Value (Age-99 | Fare-999)

```
X_train['Age_99'] = X_train['Age'].fillna(99)
X_train['Age_minus1'] = X_train['Age'].fillna(-1)

X_train['Fare_999'] = X_train['Fare'].fillna(999)
X_train['Fare_minus1'] = X_train['Fare'].fillna(-
1)
```

#Review variance

```
print('Original Age variable variance: ',
X_train['Age'].var())
print('Age Variance after 99 use imputation: ',
X_train['Age_99'].var())
print('Age Variance after -1 use imputation: ',
X_train['Age_minus1'].var())

print('Original Fare variable variance: ',
X_train['Fare'].var())
print('Fare Variance after 999 use imputation: ',
X_train['Fare_999'].var())
print('Fare Variance after -1 use imputation: ',
X_train['Fare_minus1'].var())
```

Industries Helping Hands

#Review Distribution - Age:

```
fig = plt.figure()
ax = fig.add_subplot(111)

# original variable distribution
X_train['Age'].plot(kind='kde', ax=ax)

# variable imputed with the median
X_train['Age_99'].plot(kind='kde', ax=ax,
color='red')

# variable imputed with the mean
X_train['Age_minus1'].plot(kind='kde', ax=ax,
color='green')

# add legends
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best'
```

#Review Distribution –Fare:

```
fig = plt.figure()
ax = fig.add_subplot(111)

# original variable distribution
X_train['Fare'].plot(kind='kde', ax=ax)

# variable imputed with the median
X_train['Fare_999'].plot(kind='kde', ax=ax,
  color='red')

# variable imputed with the mean
```

```
X_train['Fare_minus1'].plot(kind='kde', ax=ax,
  color='green')

# add legends
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best')
```

#Check Covariance

```
X_train.cov()
```

#Check Correlation

```
X_train.corr()
```

Industries Helping Hands

Day 11: Curious Data Minds

How generative AI-ML and Voice Technology will transform?

https://www.fastcompany.com/90909940/speak-listen-do-how-generative-ai-and-voice-technology-will-transform-how-we-live-and-work

Indian Singer Arjit Singh patent voice

https://www.moneycontrol.com/news/trends/arijit-singh-aims-to-patent-his-voice-says-ai-is-a-tool-not-threat-video-12023771.html#google_vignette

https://www.hindustantimes.com/entertainment/bollywood/the-ethics-of-using-ai-to-generate-dead-artists-voices-industry-experts-weigh-in-on-copyright-and-privacy-issues-101687521121515.html

Day 12 | Mode - imputation

Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Import Dataset

In [3]:
```
df = pd.read_csv('train.csv',usecols=['GarageQual','FireplaceQu','SalePrice'])
```

In [4]:
```
df.head()
```

Out[4]:

	FireplaceQu	GarageQual	SalePrice
0	NaN	TA	208500
1	TA	TA	181500
2	TA	TA	223500
3	Gd	TA	140000
4	TA	TA	250000

In [6]:
```
df.sample(10)
```

Out[6]:

	FireplaceQu	GarageQual	SalePrice
822	Gd	TA	225000
958	NaN	TA	185000
486	NaN	TA	156000
658	Gd	TA	97500
1254	Gd	TA	165400
238	NaN	TA	318000
69	TA	TA	225000
921	NaN	NaN	145900
1405	Gd	TA	275000
32	NaN	TA	179900

Check missing (null) value

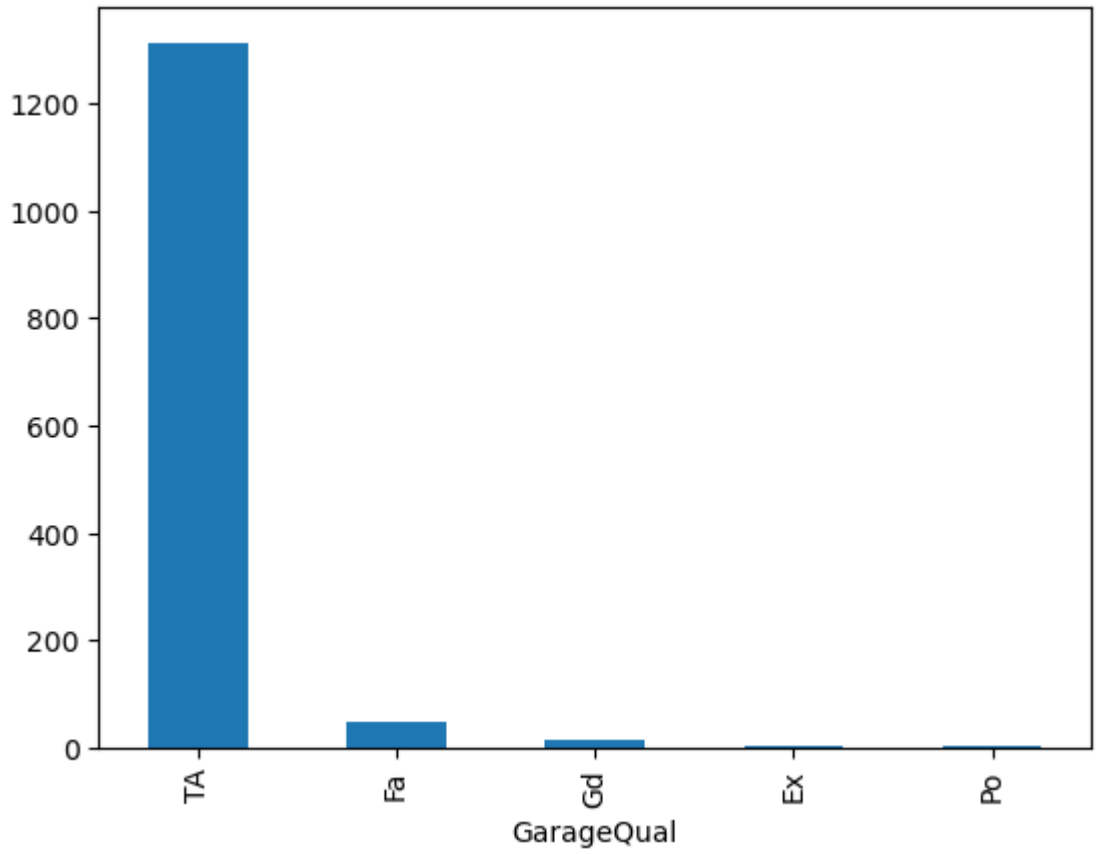
In [7]:
```
df.isnull().mean()*100
```

Out[7]:
```
FireplaceQu    47.260274
GarageQual      5.547945
SalePrice       0.000000
dtype: float64
```

Plot Bar Garage Value

In [8]:
```
df['GarageQual'].value_counts().plot(kind='bar')
```

Out[8]:
```
<Axes: xlabel='GarageQual'>
```



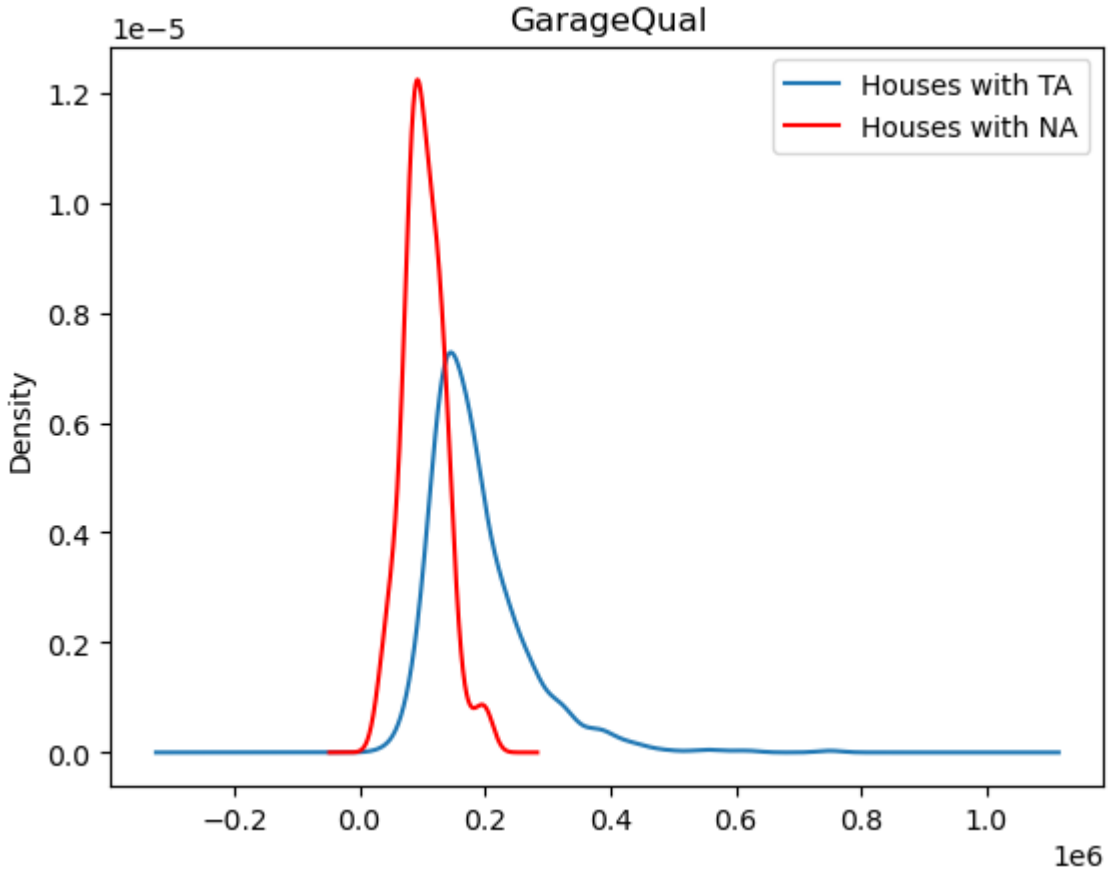
In [9]:
```
df['GarageQual'].mode()
```

Out[9]:
```
0    TA
Name: GarageQual, dtype: object
```

kde Plot | Compare Houses with TA | Null

In [10]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
df[df['GarageQual']=='TA']['SalePrice'].plot(kind='kde', ax=ax)
df[df['GarageQual'].isnull()]['SalePrice'].plot(kind='kde', ax=ax, color='red')
lines, labels = ax.get_legend_handles_labels()
labels = ['Houses with TA', 'Houses with NA']
ax.legend(lines, labels, loc='best')
plt.title('GarageQual')
```

Out[10]:
```
Text(0.5, 1.0, 'GarageQual')
```



Store variable TA in temp

In [11]:
```
temp = df[df['GarageQual']=='TA']['SalePrice']
```

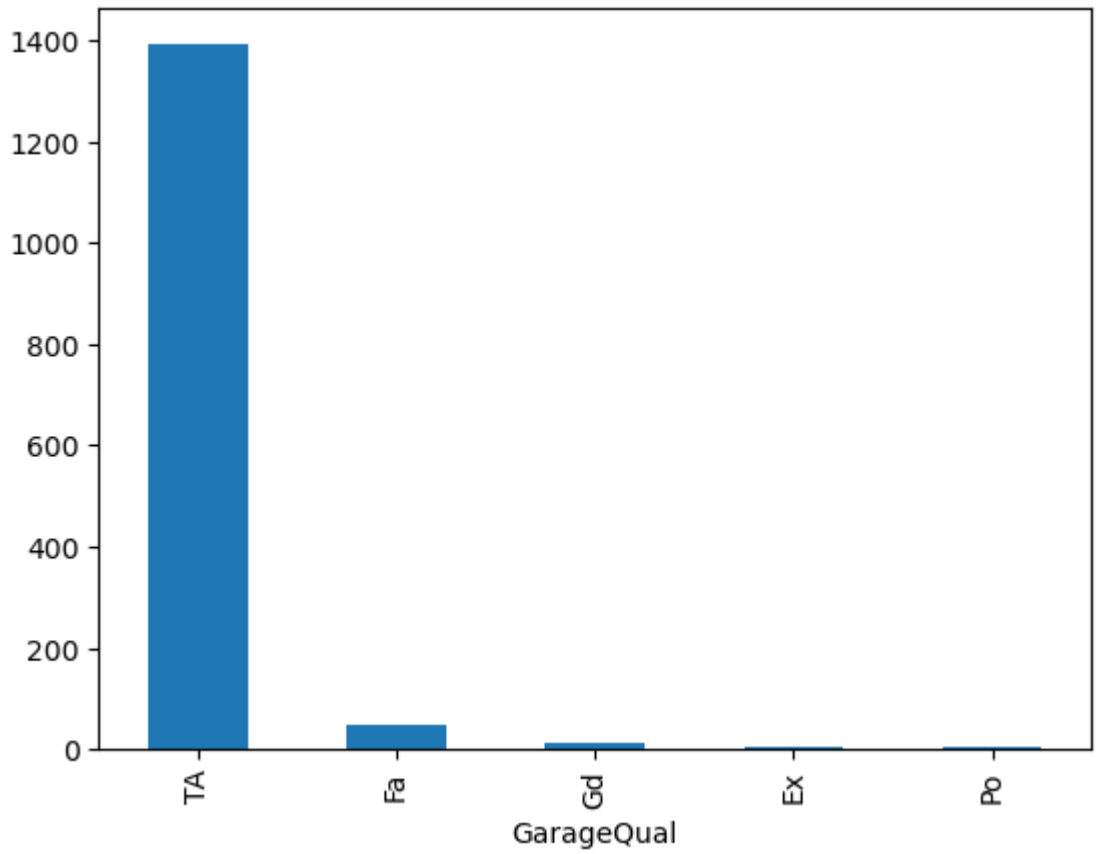
Replace missing value with TA

In [12]:
```
df['GarageQual'].fillna('TA', inplace=True)
```

Review Bar Plot Changes

In [13]:
```
df['GarageQual'].value_counts().plot(kind='bar')
```

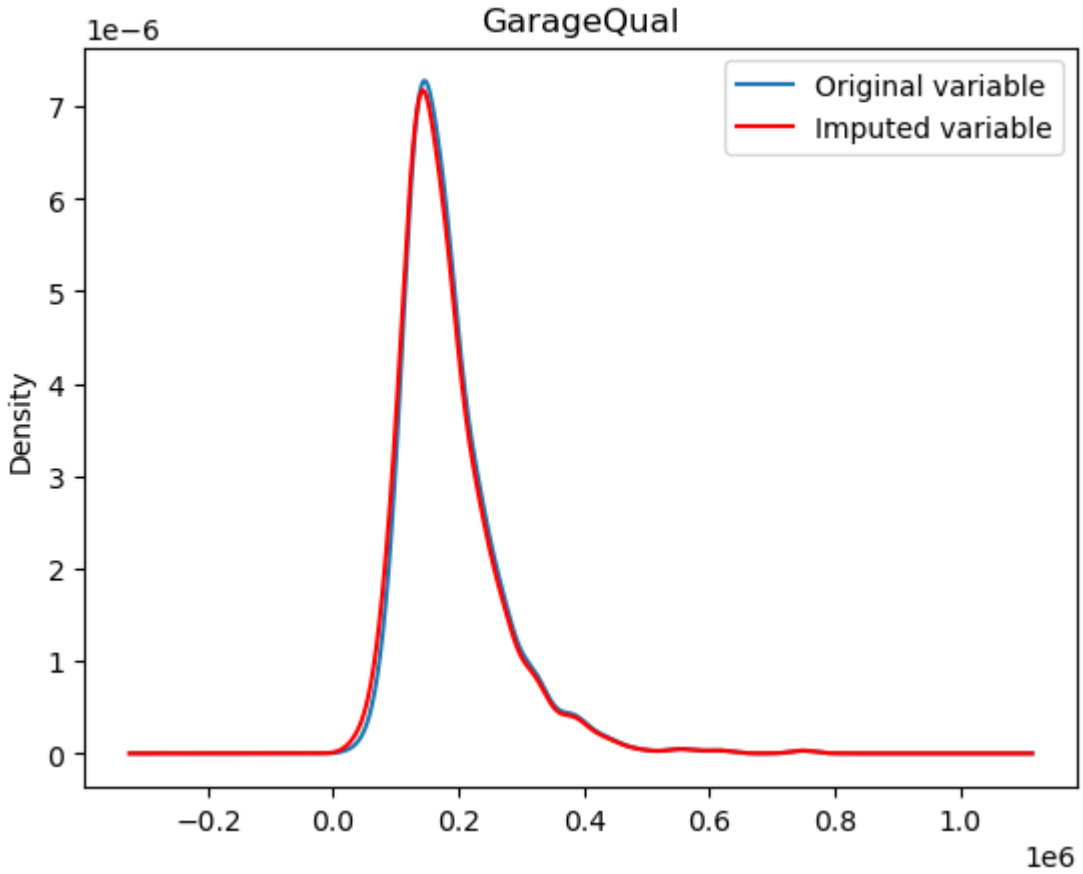
Out[13]: <Axes: xlabel='GarageQual'>



Draw plot again | After Imputation

In [14]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
temp.plot(kind='kde', ax=ax)
# distribution of the variable after imputation
df[df['GarageQual'] == 'TA']['SalePrice'].plot(kind='kde', ax=ax, color='red')
lines, labels = ax.get_legend_handles_labels()
labels = ['Original variable', 'Imputed variable']
ax.legend(lines, labels, loc='best')
# add title
plt.title('GarageQual')
```

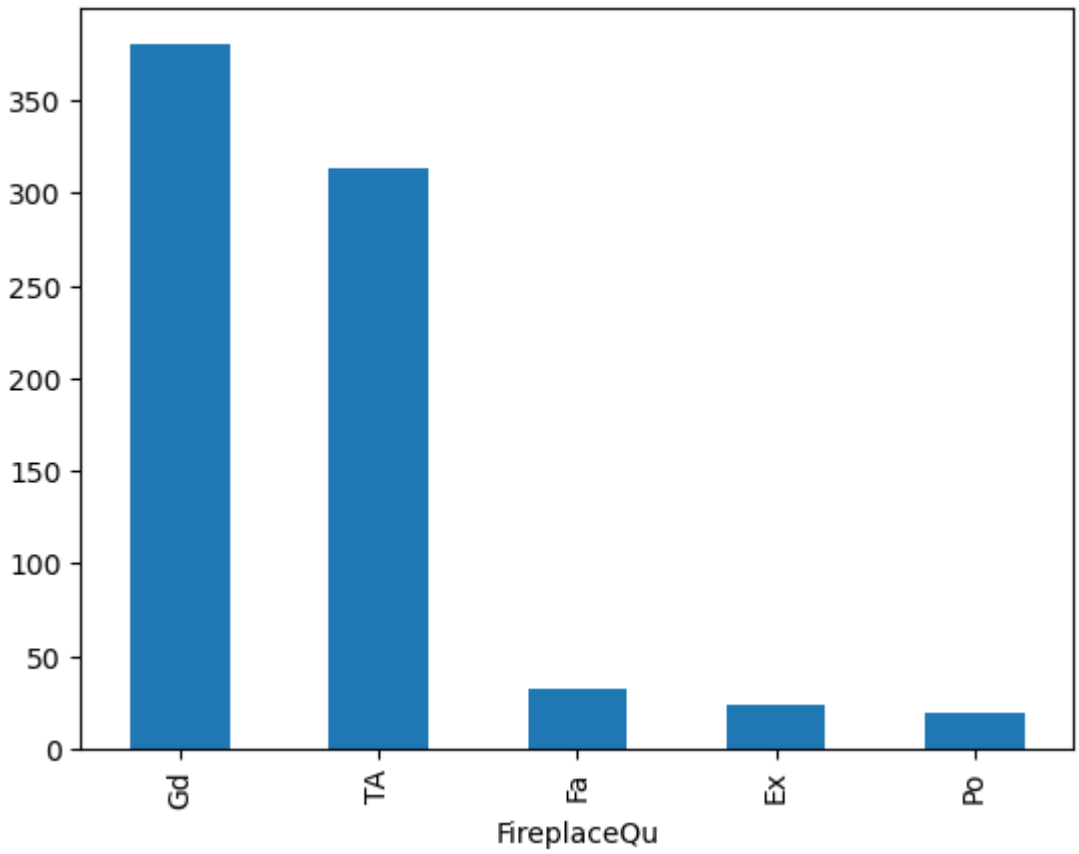
Out[14]:
```
Text(0.5, 1.0, 'GarageQual')
```



Plot Bar Fire Place

In [15]:
```
df['FireplaceQu'].value_counts().plot(kind='bar')
```

Out[15]:
```
<Axes: xlabel='FireplaceQu'>
```



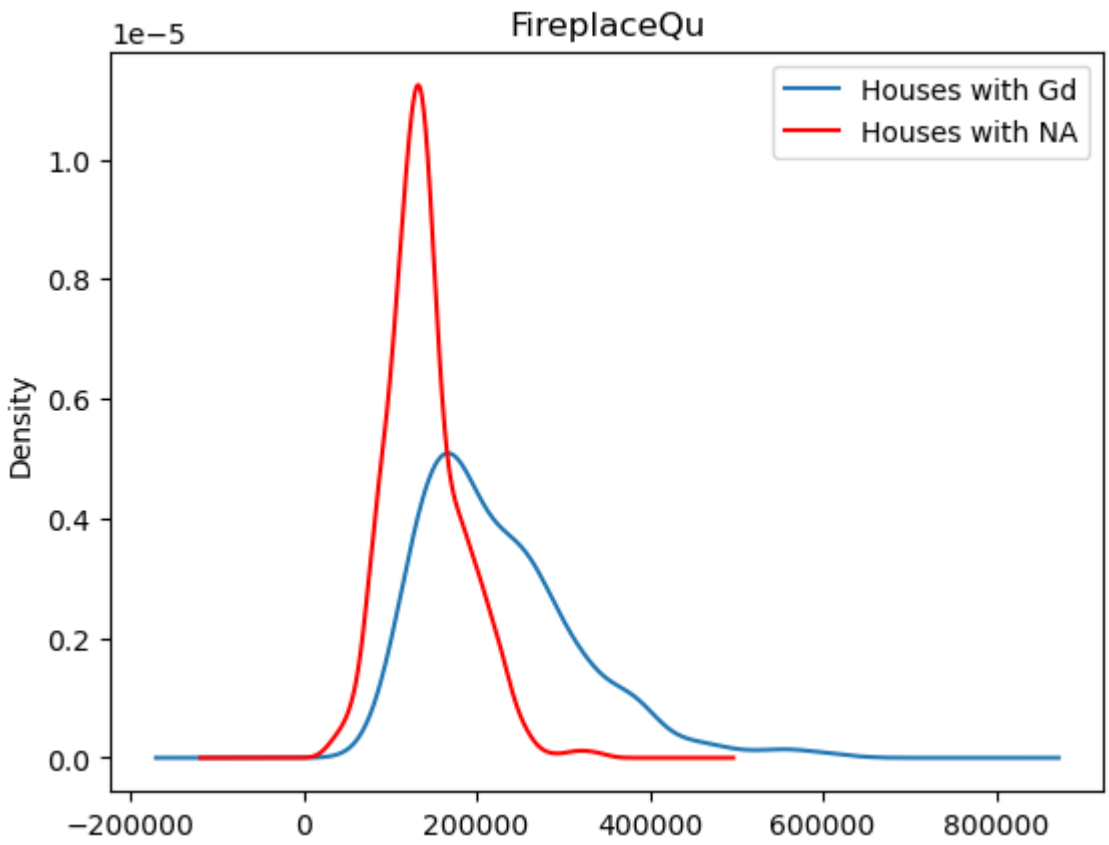
In [16]:
```
df['FireplaceQu'].mode()
```

Out[16]:
```
0    Gd
Name: FireplaceQu, dtype: object
```

kde Plot | Replace House with GD and NA

In [17]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
df[df['FireplaceQu']=='Gd']['SalePrice'].plot(kind='kde', ax=ax)
df[df['FireplaceQu'].isnull()]['SalePrice'].plot(kind='kde', ax=ax, color='red')
lines, labels = ax.get_legend_handles_labels()
labels = ['Houses with Gd', 'Houses with NA']
ax.legend(lines, labels, loc='best')
plt.title('FireplaceQu')
```

Out[17]:
```
Text(0.5, 1.0, 'FireplaceQu')
```



Store Temp Variable

In [18]:
```
temp = df[df['FireplaceQu']=='Gd']['SalePrice']
```

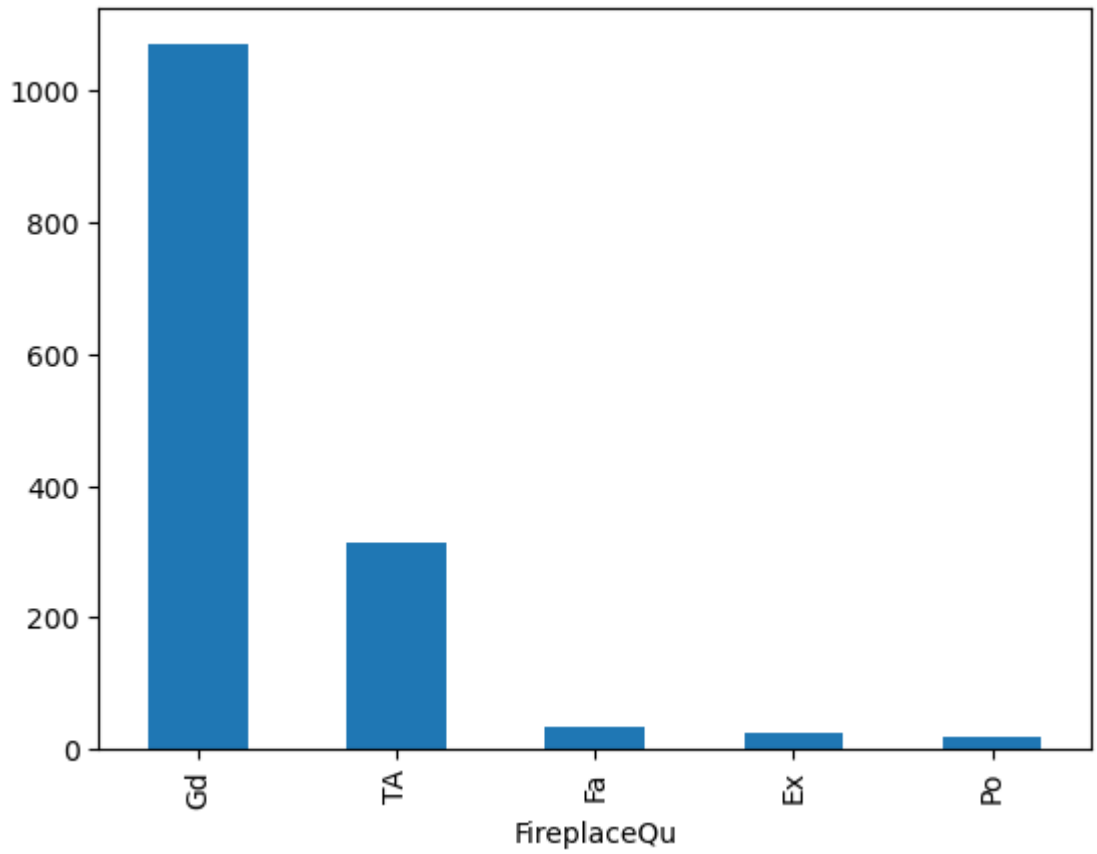
Replace missing value with GD

In [19]:
```
df['FireplaceQu'].fillna('Gd', inplace=True)
```

Draw Bar Plot

In [20]:
```
df['FireplaceQu'].value_counts().plot(kind='bar')
```

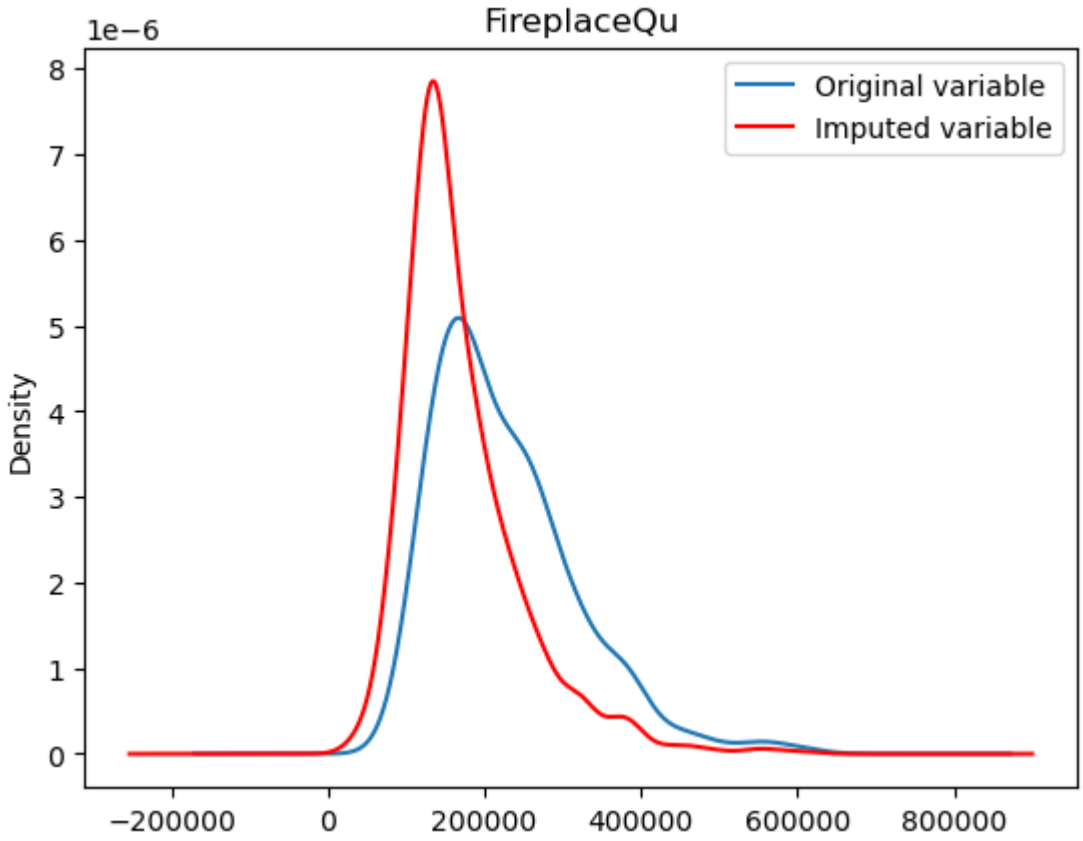
Out[20]:
```
<Axes: xlabel='FireplaceQu'>
```



Draw plot again | After Imputation

In [21]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
temp.plot(kind='kde', ax=ax)
# distribution of the variable after imputation
df[df['FireplaceQu'] == 'Gd']['SalePrice'].plot(kind='kde', ax=ax, color='red')
lines, labels = ax.get_legend_handles_labels()
labels = ['Original variable', 'Imputed variable']
ax.legend(lines, labels, loc='best')
# add title
plt.title('FireplaceQu')
```

Out[21]:
```
Text(0.5, 1.0, 'FireplaceQu')
```



In [ ]:

Day 12 | Missing Value - imputation

Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Import Data

In [3]:
```
df = pd.read_csv('train.csv',usecols=['GarageQual','FireplaceQu','SalePrice'])
```

In [4]:
```
df.head()
```

Out[4]:

	FireplaceQu	GarageQual	SalePrice
0	NaN	TA	208500
1	TA	TA	181500
2	TA	TA	223500
3	Gd	TA	140000
4	TA	TA	250000

Check missing (null) value

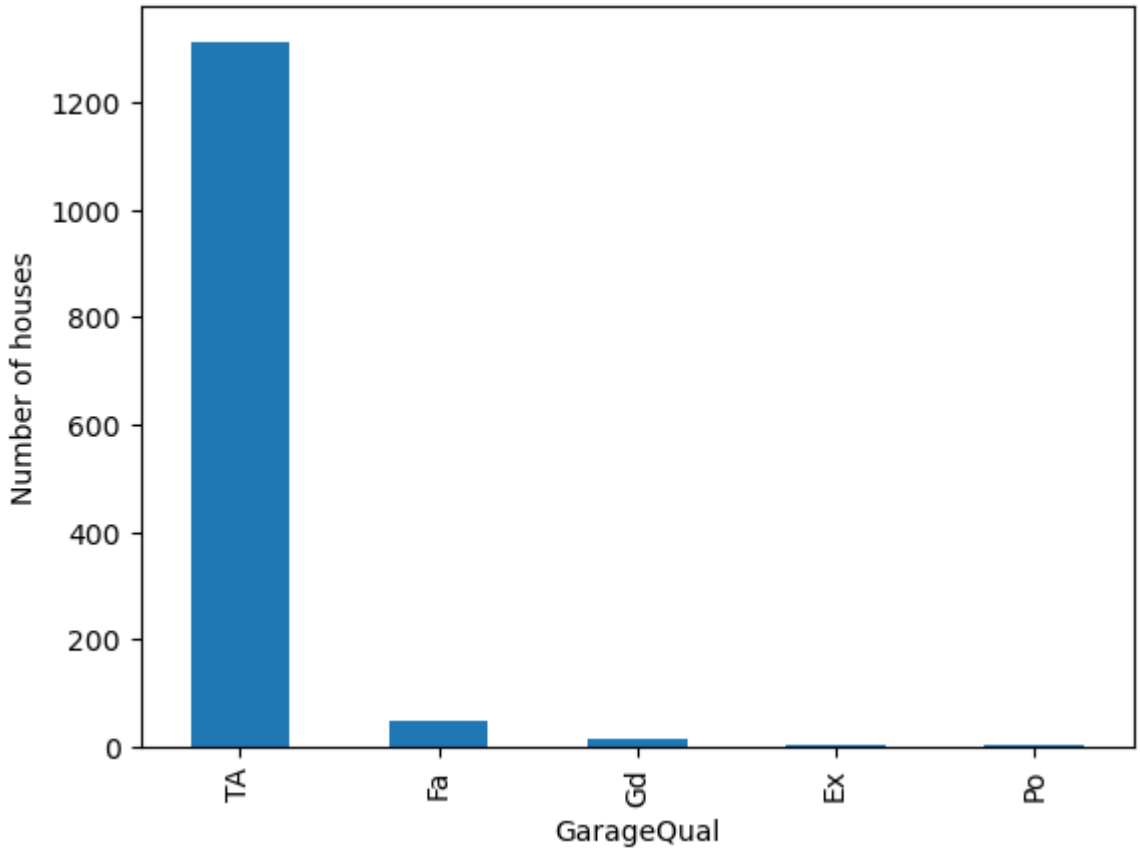
In [5]:
```
df.isnull().mean()*100
```

Out[5]:
```
FireplaceQu    47.260274
GarageQual      5.547945
SalePrice       0.000000
dtype: float64
```

Plot bar | Garage

In [6]:
```
df['GarageQual'].value_counts().sort_values(ascending=False).plot.bar()
plt.xlabel('GarageQual')
plt.ylabel('Number of houses')
```

Out[6]:
```
Text(0, 0.5, 'Number of houses')
```



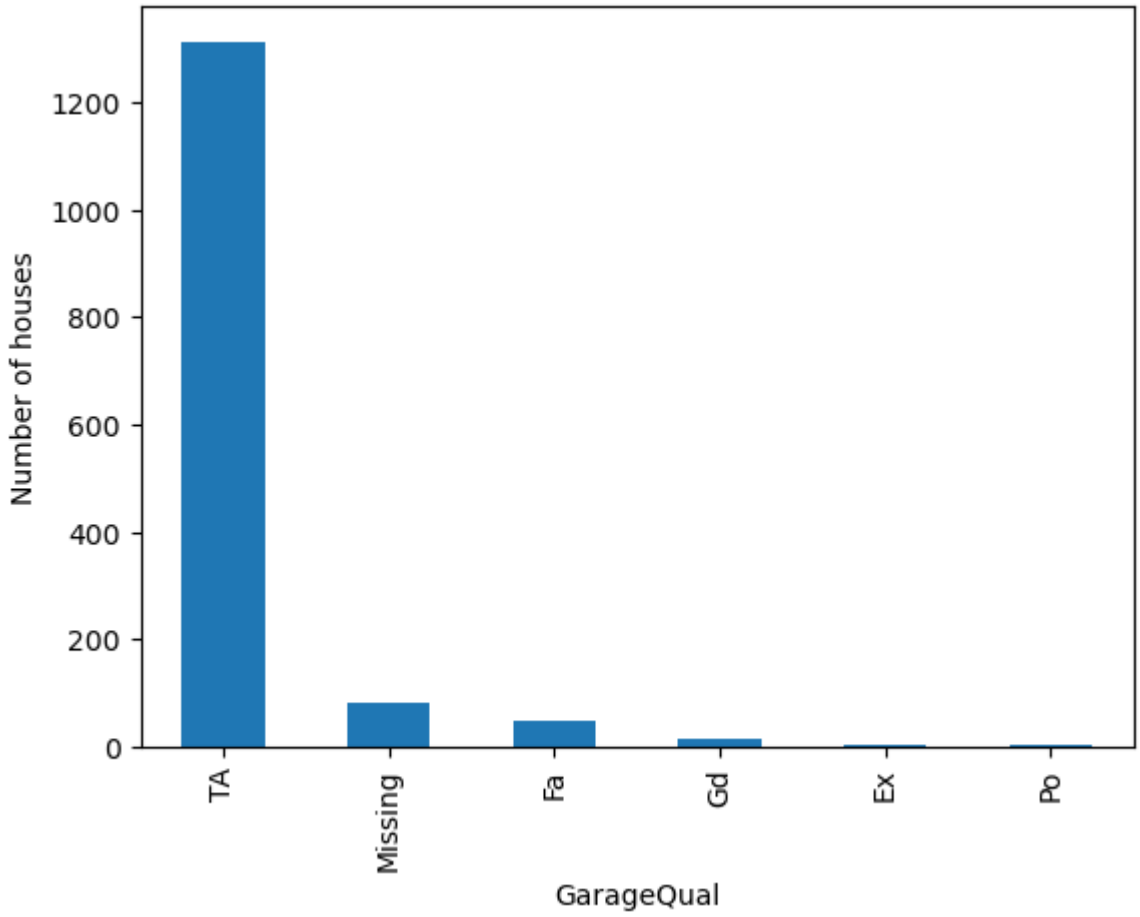
Replace Missing Value in “Missing” Word

In [11]:
```
df['GarageQual'].fillna('Missing', inplace=True)
```

Category Missing Values

In [12]:
```
df['GarageQual'].value_counts().sort_values(ascending=False).plot.bar()
plt.xlabel('GarageQual')
plt.ylabel('Number of houses')
```

Out[12]:
```
Text(0, 0.5, 'Number of houses')
```



In [ ]:

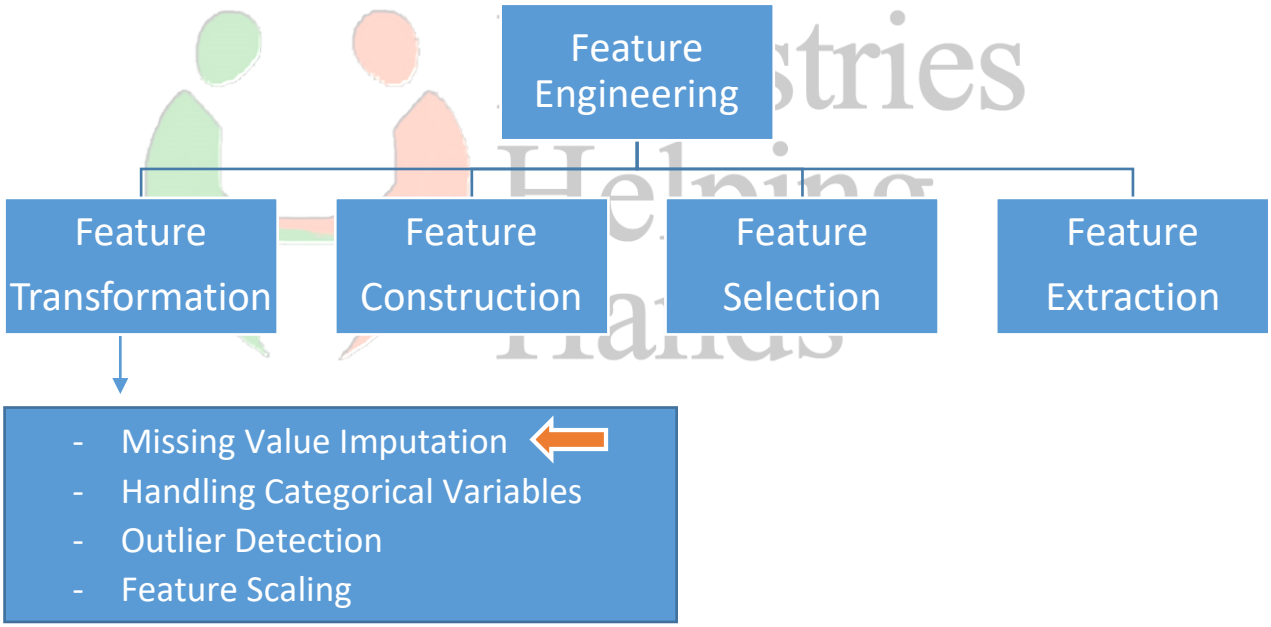
Data Science | 30 Days of Machine Learning | Day - 12

Educator Name: Nishant Dhote
Support Team: **+91-7880-113-112**

----Today Topics | Day 12----

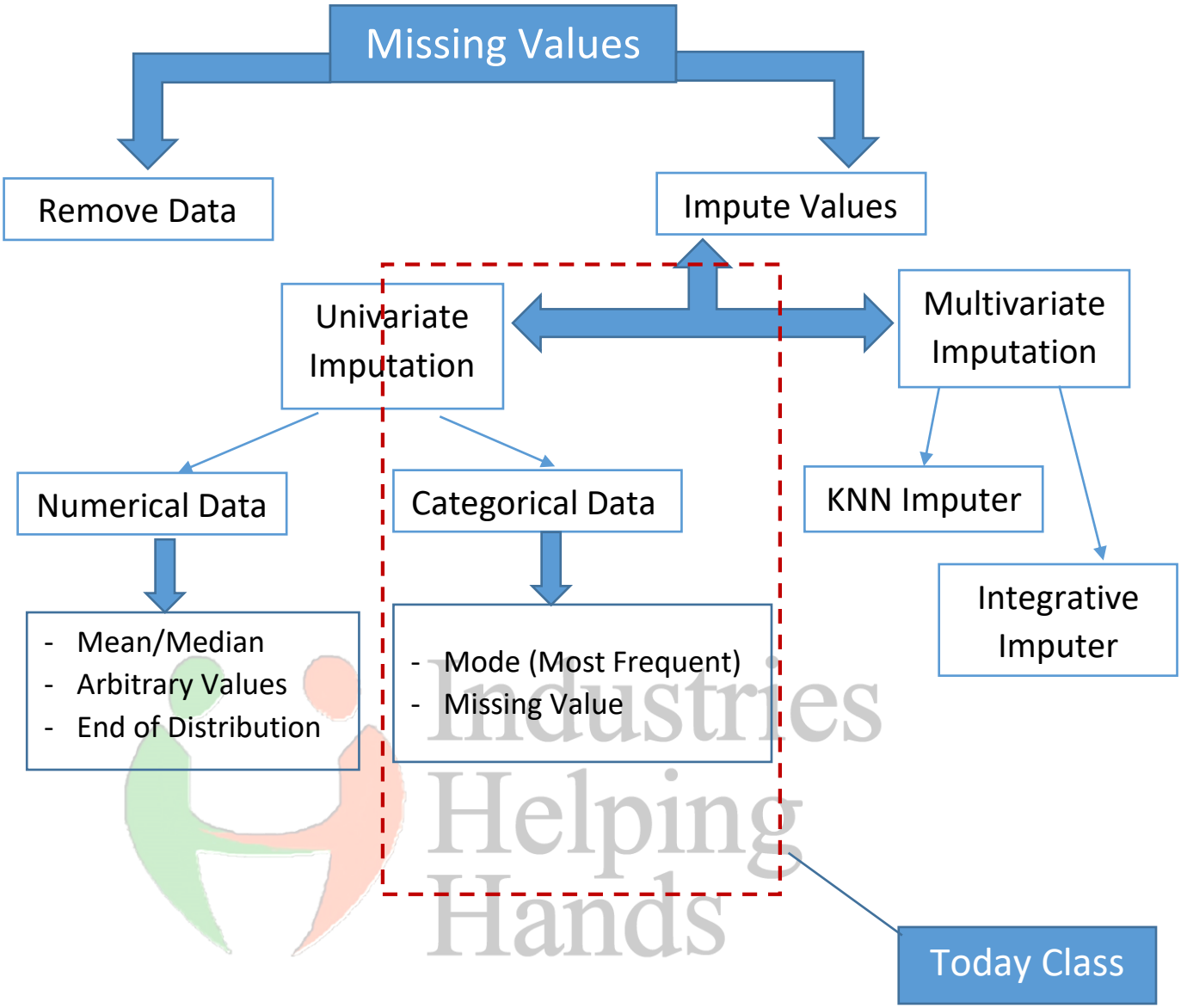
- What is the purpose of using "Mode"?
- What is most frequent imputation?
- Which variables can be imputed with most frequent / mode Imputation?
- When to use mode / most frequent category imputation?
- Missing value imputation?

Dataset Link GitHub: https://github.com/TheiScale/30_Days_Machine_Learning/



- Two Important ways to handle missing values in the dataset:

  1. Deleting rows with missing values (Remove Missing Values)
  2. Impute missing values (Fill)
     - Univariate (Numerical & Categorical Removal)
     - Multivariate (KNN & Iterative Imputer)



What is the purpose of using "Mode"?

The mode is a statistical measure in machine learning that represents the most frequently occurring value in a dataset. It can be used to identify the central tendency of categorical data.

What is most frequent imputation?

Mode imputation consists of replacing all occurrences of missing values (NA) within a variable by the mode, which in other words refers to the most frequent value or most frequent category.

Make
Ford
Ford
Fiat
BMW
Ford
Kia
Fiat
Ford
Kia

Mode = Ford

Price
Ford
Ford
Fiat
BMW
Ford
Kia
Ford
Fiat
Ford
Ford
Kia

Which variables can be imputed with most frequent / mode Imputation?

Although the mode, or most frequent value can be calculated for both numerical and categorical variables, in practice, we only use this technique on categorical variables. The reason is that for numerical variables, the mean or the median tend to better represent the average value of the population.

When to use mode / most frequent category imputation?

Data is missing completely at random.
No more than 5% of the variable contains missing data.

House Price Kaggle Datasets: https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data

<Start Coding | Mode - imputation>

#Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

#Import Dataset

```
df =
pd.read_csv('train.csv',usecols=['GarageQual','Fir
eplaceQu','SalePrice'])
```

```
df.head()
```

#Check missing (null) value

```
df.isnull().mean()*100
```

#Plot Bar Garage Value

```
df['GarageQual'].value_counts().plot(kind='bar')
```

```
df['GarageQual'].mode()
```

#kde Plot | Compare Houses with TA | Null

```
fig = plt.figure()
ax = fig.add_subplot(111)

df[df['GarageQual']=='TA']['SalePrice'].plot(kind
='kde', ax=ax)

df[df['GarageQual'].isnull()]['SalePrice'].plot(k
ind='kde', ax=ax, color='red')

lines, labels = ax.get_legend_handles_labels()
labels = ['Houses with TA', 'Houses with NA']
ax.legend(lines, labels, loc='best')

plt.title('GarageQual')
```

#Store variable TA in temp

```
temp = df[df['GarageQual']=='TA']['SalePrice']
```

#Replace missing value with TA

```
df['GarageQual'].fillna('TA', inplace=True)
```

#Review Bar Plot Changes

```
df['GarageQual'].value_counts().plot(kind='bar')
```

#Draw plot again | After Imputation

```
fig = plt.figure()
ax = fig.add_subplot(111)


temp.plot(kind='kde', ax=ax)

# distribution of the variable after imputation
df[df['GarageQual'] ==
'TA']['SalePrice'].plot(kind='kde', ax=ax,
color='red')

lines, labels = ax.get_legend_handles_labels()
labels = ['Original variable', 'Imputed
variable']
ax.legend(lines, labels, loc='best')

# add title
plt.title('GarageQual')
```

#Plot Bar Fire Place

```
df['FireplaceQu'].value_counts().plot(kind='bar')
----
df['FireplaceQu'].mode()
```

#kde Plot | Replace House with GD and NA

```
fig = plt.figure()
ax = fig.add_subplot(111)

df[df['FireplaceQu']=='Gd']['SalePrice'].plot(kin
d='kde', ax=ax)

df[df['FireplaceQu'].isnull()]['SalePrice'].plot(
kind='kde', ax=ax, color='red')

lines, labels = ax.get_legend_handles_labels()
labels = ['Houses with Gd', 'Houses with NA']
ax.legend(lines, labels, loc='best')

plt.title('FireplaceQu')
```

#Store Temp Variable

```
temp = df[df['FireplaceQu']=='Gd']['SalePrice']
```

#Replace missing value with TA

```
df['FireplaceQu'].fillna('Gd', inplace=True)
```

#Draw Bar Plot

```
df['FireplaceQu'].value_counts().plot(kind='bar')
```

#Draw plot again | After Imputation

```
fig = plt.figure()
ax = fig.add_subplot(111)


temp.plot(kind='kde', ax=ax)

# distribution of the variable after imputation

df[df['FireplaceQu'] ==
'Gd']['SalePrice'].plot(kind='kde', ax=ax,
color='red')

lines, labels = ax.get_legend_handles_labels()
labels = ['Original variable', 'Imputed
variable']
ax.legend(lines, labels, loc='best')

# add title
plt.title('FireplaceQu')
```

Missing Value Imputation: Data imputation is a method for retaining the majority of the dataset's data and information by substituting missing data with a different value.

<Start Coding | Missing Value - imputation>

#Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

#Import Data

```
df =
pd.read_csv('train.csv',usecols=['GarageQual','Fi
replaceQu','SalePrice'])

----
df.head()
```

#Check missing (null) value

```
df.isnull().mean()*100
```

#Plot bar | Garage

```
df['GarageQual'].value_counts().sort_values(ascen
ding=False).plot.bar()
plt.xlabel('GarageQual')
plt.ylabel('Number of houses')
```

#Replace Missing Value in "Missing" Word

```
df['GarageQual'].fillna('Missing', inplace=True
```

#Category Missing Values

```
df['GarageQual'].value_counts().sort_values(ascend
ing=False).plot.bar()
plt.xlabel('GarageQual')
plt.ylabel('Number of houses')
```

Industries Helping Hands

Day 12: Curious Data Minds

- **Data Science and AI in the Travel Industry:**

Read Blog: https://www.altexsoft.com/blog/data-science-and-ai-in-the-travel-industry-9-real-life-use-cases/

https://economictimes.indiatimes.com/jobs/government-jobs/ayodhya-tourism-boom-may-create-150000-200000-direct-and-indirect-jobs/articleshow/107124078.cms?from=mdr

How OYO uses Data Analytics

As of January 2020, it has more than 43,000 properties and 10 lakh (1 million) rooms across 800 cities in 80 countries, including

India, Malaysia, the UAE, Nepal, China, Brazil, Mexico, the UK, Philippines, Japan, Saudi Arabia, Sri Lanka, Indonesia, Vietnam, and the United States.

Area served **Asia, Europe and Americas**
Revenue **₹4,157 crore**

OYO Business Model Canvas

Customer Segments

- People who love to travel and explore different cities
- Bars, hotels, clubs owners, event organizers.

Key Partners

- Hotels, Clubs and Bars
- Event Organizations
- Sponsors

Value Proposition

- Simplicity- Customer Friendly & no information overflow.
- Convenience- Users can explore list of events in one click.
- Speed- Quick Navigation & Personalized Results.
- Design- Modern and adaptive.
- Reliability- Customer's confidentiality is of utmost priority.

Key Activities

- App Maintenance
- Promotion and Advertising
- Creating Community

Customer Relationships

- Advertisements and Social Media.
- Sharing of events with App community.
- By providing accessibility and reliability

Key Resources

- Skilled OYO employees
- A Powerful Mobile Application
- Large Database of Users and Clients

Cost Structure

1) Fixed Costs

- Salary of Employees
- IT-development &

2) Variable Costs-

- Production
- Transportation

Revenue Streams

- Advertising
- Sponsors and Partnerships
- Business Consulting

Channels

- Mobile Applications
- Website

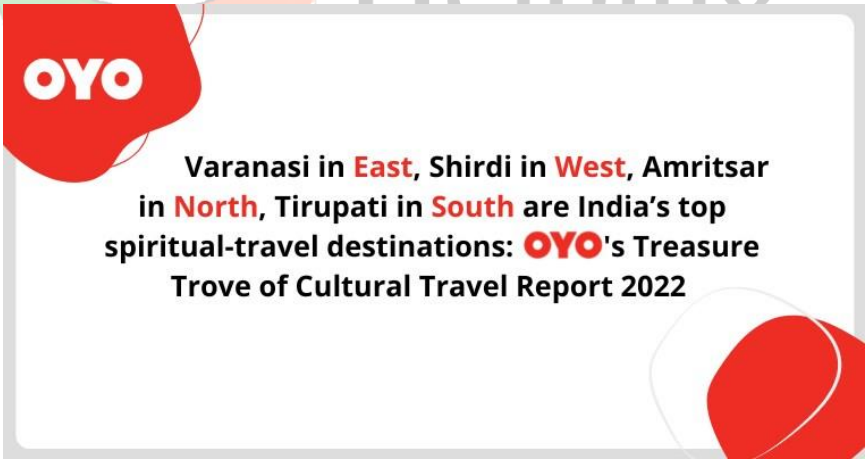
"Our data Analysts use natural curiosity and innovative tools to derive deep insights into customer behavior. These insights not only help us improve our service but also take effective business decisions,"

said Ritesh Agarwal, founder & CEO of OYO.

OYO users spent 3,232 years' worth of time on the OYO app in India – the highest in India in FY2021

Subah-Sham. Quite literally. The most popular time to make bookings on the OYO app were 11:00 AM – 1:00 PM and evening 6:00 PM – 9:00 PM

Fan Alert: A travel agent from India made 1193 bookings for an OYO in 2021



Day 13 | ML | Random - Sample - imputation | Numeric Data

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
```

Import Dataset

In [2]:
```
df = pd.read_csv('train.csv',usecols=['Age','Fare','Survived'])
```

In [3]:
```
df.head()
```

Out[3]:

	Survived	Age	Fare
0	0	22.0	7.2500
1	1	38.0	71.2833
2	1	26.0	7.9250
3	1	35.0	53.1000
4	0	35.0	8.0500

Check missing (null) value

In [4]:
```
df.isnull().mean() * 100
```

Out[4]:
```
Survived     0.00000
Age         19.86532
Fare         0.00000
dtype: float64
```

Create X & Y

In [5]:
```
X = df.drop(columns=['Survived'])
y = df['Survived']
```

Apply Train Test Split

In [6]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [9]:
```
X_train
```

Out[9]:

	Age	Fare
30	40.0	27.7208
10	4.0	16.7000
873	47.0	9.0000
182	9.0	31.3875
876	20.0	9.8458
...	...	...
534	30.0	8.6625
584	NaN	8.7125
493	71.0	49.5042
527	NaN	221.7792
168	NaN	25.9250

712 rows × 2 columns

New column create in Both Train & Test

In [10]:
```
X_train['Age_imputed'] = X_train['Age']
X_test['Age_imputed'] = X_test['Age']
```

In [11]:
```
X_test.tail()
```

Out[11]:

	Age	Fare	Age_imputed
89	24.0	8.0500	24.0
80	22.0	9.0000	22.0
846	NaN	69.5500	NaN
870	26.0	7.8958	26.0
251	29.0	10.4625	29.0

In [12]:
```
X_test.head()
```

Out[12]:

	Age	Fare	Age_imputed
707	42.0	26.2875	42.0
37	21.0	8.0500	21.0
615	24.0	65.0000	24.0
169	28.0	56.4958	28.0
68	17.0	7.9250	17.0

In [13]:
```
X_train.tail()
```

Out[13]:

	Age	Fare	Age_imputed
534	30.0	8.6625	30.0
584	NaN	8.7125	NaN
493	71.0	49.5042	71.0
527	NaN	221.7792	NaN
168	NaN	25.9250	NaN

In [14]:
```
X_train.head()
```

Out[14]:

	Age	Fare	Age_imputed
30	40.0	27.7208	40.0
10	4.0	16.7000	4.0
873	47.0	9.0000	47.0
182	9.0	31.3875	9.0
876	20.0	9.8458	20.0

Replace Value Age_imputed

In [15]:
```
X_train['Age_imputed'][X_train['Age_imputed'].isnull()] = X_train['Age'].dropna().sam
X_test['Age_imputed'][X_test['Age_imputed'].isnull()] = X_train['Age'].dropna().sampl
```

Review Sample Random Generate Value

In [16]:
```
X_train['Age'].dropna().sample(1).values
```

Out[16]:
```
array([65.])
```

In [17]:
```
X_train['Age'].isnull().sum()
```

Out[17]:
```
148
```

In [21]:
```
X_train['Age'].dropna().sample(X_train['Age'].isnull().sum()).values
```

Out[21]:
```
array([35.  , 62.  , 28.  , 24.  , 22.  , 23.  ,  9.  , 25.  , 24.  ,
        3.  , 36.  , 24.  , 40.  , 17.  , 28.  , 40.  , 26.  , 30.  ,
       43.  , 16.  , 39.  , 35.  , 24.  ,  4.  , 40.  , 42.  , 50.  ,
       31.  , 31.  ,  4.  ,  2.  , 42.  , 26.  , 21.  , 56.  , 45.  ,
        0.75, 14.  , 15.  , 38.  , 22.  , 36.  ,  4.  , 21.  , 20.  ,
       18.  ,  1.  , 24.  , 54.  , 23.  , 60.  , 17.  , 35.  , 50.  ,
        2.  , 36.  , 25.  , 14.  , 43.  , 17.  , 25.  , 28.  , 30.  ,
       29.  , 51.  , 16.  ,  9.  , 22.  , 50.  , 63.  , 56.  , 40.  ,
       30.  , 33.  , 31.  , 31.  , 36.  , 35.  ,  4.  , 54.  , 39.  ,
       39.  , 47.  , 16.  , 28.  , 16.  , 44.  , 50.  , 15.  , 21.  ,
       20.  , 61.  , 25.  , 46.  , 36.  , 27.  , 42.  , 18.  , 23.  ,
        3.  , 25.  , 14.  , 19.  , 21.  , 27.  , 25.  , 25.  , 19.  ,
       38.  , 23.  , 39.  , 43.  , 39.  , 52.  , 32.  , 48.  , 54.  ,
       24.  , 15.  , 18.  , 32.  , 44.  , 27.  , 56.  , 28.  ,  9.  ,
       58.  , 45.  , 48.  , 16.  , 24.  , 41.  , 25.  , 28.  , 16.  ,
       35.  , 20.  , 47.  , 20.  , 24.  , 19.  , 32.  , 64.  , 27.  ,
       33.  , 57.  , 41.  , 16.  ])
```

In [24]:
```
X_train
```

Out[24]:

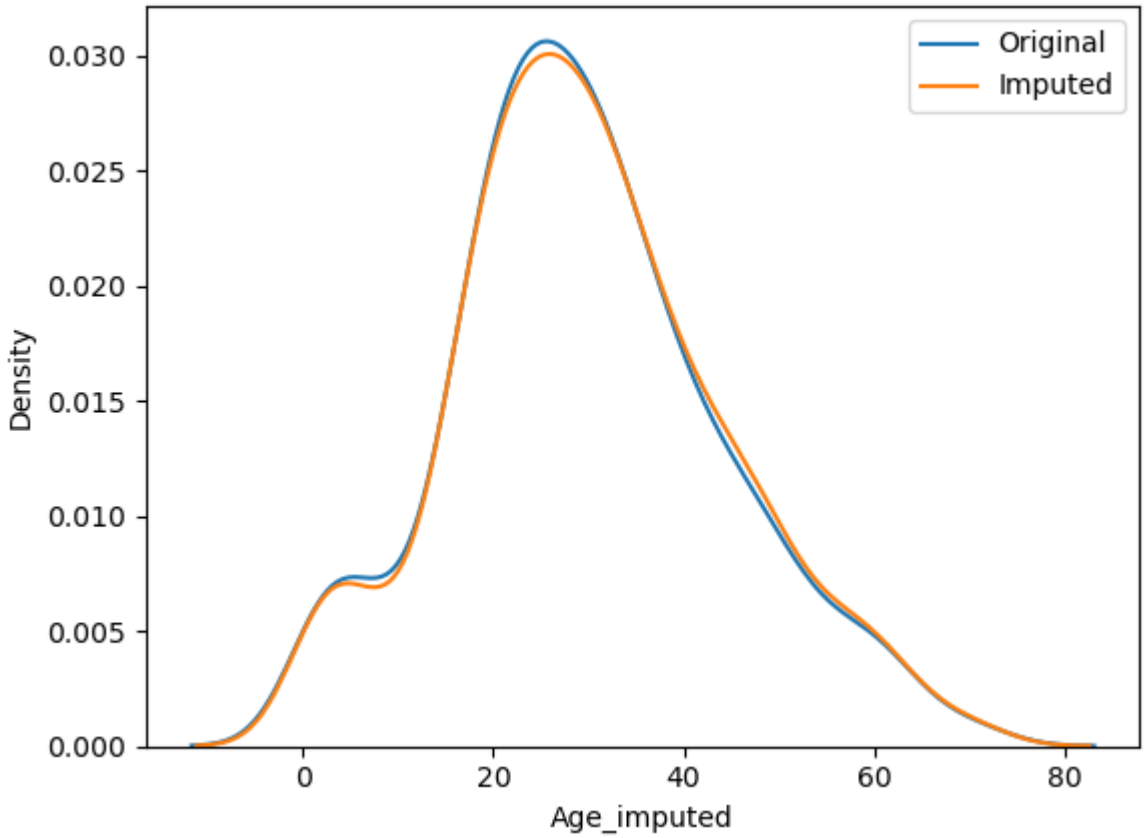
	Age	Fare	Age_imputed
30	40.0	27.7208	40.0
10	4.0	16.7000	4.0
873	47.0	9.0000	47.0
182	9.0	31.3875	9.0
876	20.0	9.8458	20.0
...	...	...	...
534	30.0	8.6625	30.0
584	NaN	8.7125	18.0
493	71.0	49.5042	71.0
527	NaN	221.7792	32.0
168	NaN	25.9250	24.0

712 rows × 3 columns

Compare Original Age and Imputed Age

In [25]:
```
sns.distplot(X_train['Age'],label='Original',hist=False)
sns.distplot(X_train['Age_imputed'],label = 'Imputed',hist=False)
plt.legend()
plt.show()
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_12888\2131456624.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.githu
b.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(X_train['Age'],label='Original',hist=False)
C:\Users\ASUS\AppData\Local\Temp\ipykernel_12888\2131456624.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.githu
b.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(X_train['Age_imputed'],label = 'Imputed',hist=False)
```



Compare Variable Variance

In [26]:
```
print('Original variable variance: ', X_train['Age'].var())
print('Variance after random imputation: ', X_train['Age_imputed'].var())
```

```
Original variable variance:  204.3495133904614
Variance after random imputation:  206.42663730700545
```

In [ ]:

Random Sample imputation for Categorical Data

Import Libraries

In [17]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
```

Import Dataset

In [18]:
```
df = pd.read_csv('house-train.csv',usecols=['GarageQual','FireplaceQu', 'SalePrice'])
```

In [19]:
```
df.sample(10)
```

Out[19]:

	FireplaceQu	GarageQual	SalePrice
1262	TA	TA	161500
501	NaN	TA	226700
881	TA	TA	187500
1151	Gd	TA	149900
1440	Gd	TA	191000
712	Gd	TA	189000
46	Ex	TA	239686
766	TA	TA	196500
161	Gd	TA	412500
66	Gd	TA	180000

Check missing (null) value

In [20]:
```
df.isnull().mean() * 100
```

Out[20]:
```
FireplaceQu    47.260274
GarageQual      5.547945
SalePrice       0.000000
dtype: float64
```

Create X & Y

In [21]:
```
X = df
y = df['SalePrice']
```

Apply Train Test Split

In [22]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [23]:
```
X_train
```

Out[23]:

	FireplaceQu	GarageQual	SalePrice
455	TA	TA	175500
863	NaN	TA	132500
504	Fa	TA	147000
1413	Gd	TA	257000
1389	Gd	TA	131000
...	...	...	...
466	Po	TA	167000
299	Gd	TA	158500
493	Fa	TA	155000
527	Gd	TA	446261
1192	NaN	TA	125000

1168 rows × 3 columns

New column create in Both Garage & Fire place

In [24]:
```
X_train['GarageQual_imputed'] = X_train['GarageQual']
X_test['GarageQual_imputed'] = X_test['GarageQual']
X_train['FireplaceQu_imputed'] = X_train['FireplaceQu']
X_test['FireplaceQu_imputed'] = X_test['FireplaceQu']
```

In [25]:
```
X_train.sample(5)
```

Out[25]:

	FireplaceQu	GarageQual	SalePrice	GarageQual_imputed	FireplaceQu_imputed
258	TA	TA	231500	TA	TA
609	NaN	TA	118500	TA	NaN
195	TA	TA	148500	TA	TA
1052	TA	TA	165000	TA	TA
1068	Gd	TA	151400	TA	Gd

Replace Garage Value and Fireplace Imputed

In [30]:
```
X_train['GarageQual_imputed'][X_train['GarageQual_imputed'].isnull()] = X_train['Gara
X_test['GarageQual_imputed'][X_test['GarageQual_imputed'].isnull()] = X_train['Garage
X_train['FireplaceQu_imputed'][X_train['FireplaceQu_imputed'].isnull()] = X_train['Fi
X_test['FireplaceQu_imputed'][X_test['FireplaceQu_imputed'].isnull()] = X_train['Fire
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_10452\3418880262.py:1: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  X_train['GarageQual_imputed'][X_train['GarageQual_imputed'].isnull()] = X_train['G
arageQual'].dropna().sample(X_train['GarageQual'].isnull().sum()).values
C:\Users\ASUS\AppData\Local\Temp\ipykernel_10452\3418880262.py:2: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  X_test['GarageQual_imputed'][X_test['GarageQual_imputed'].isnull()] = X_train['Gar
ageQual'].dropna().sample(X_test['GarageQual'].isnull().sum()).values
C:\Users\ASUS\AppData\Local\Temp\ipykernel_10452\3418880262.py:3: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  X_train['FireplaceQu_imputed'][X_train['FireplaceQu_imputed'].isnull()] = X_train
['FireplaceQu'].dropna().sample(X_train['FireplaceQu'].isnull().sum()).values
C:\Users\ASUS\AppData\Local\Temp\ipykernel_10452\3418880262.py:4: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  X_test['FireplaceQu_imputed'][X_test['FireplaceQu_imputed'].isnull()] = X_train['F
ireplaceQu'].dropna().sample(X_test['FireplaceQu'].isnull().sum()).values
```

Review Frequency in Garage Original & Imputed

In [32]:
```
temp = pd.concat(
[
X_train['GarageQual'].value_counts() / len(X_train['GarageQual'].dropna()),
X_train['GarageQual_imputed'].value_counts() / len(X_train)
],
axis=1)
temp.columns = ['original', 'imputed']
temp
```

Out[32]:

	original	imputed
TA	0.951043	0.952055
Fa	0.037171	0.035959
Gd	0.009973	0.010274
Po	0.000907	0.000856
Ex	0.000907	0.000856

Review Frequency in Fireplace Original & Imputed

In [33]:
```
temp = pd.concat(
[
X_train['FireplaceQu'].value_counts() / len(X_train['FireplaceQu'].dropna()),
X_train['FireplaceQu_imputed'].value_counts() / len(df)
],
axis=1)
temp.columns = ['original', 'imputed']
temp
```

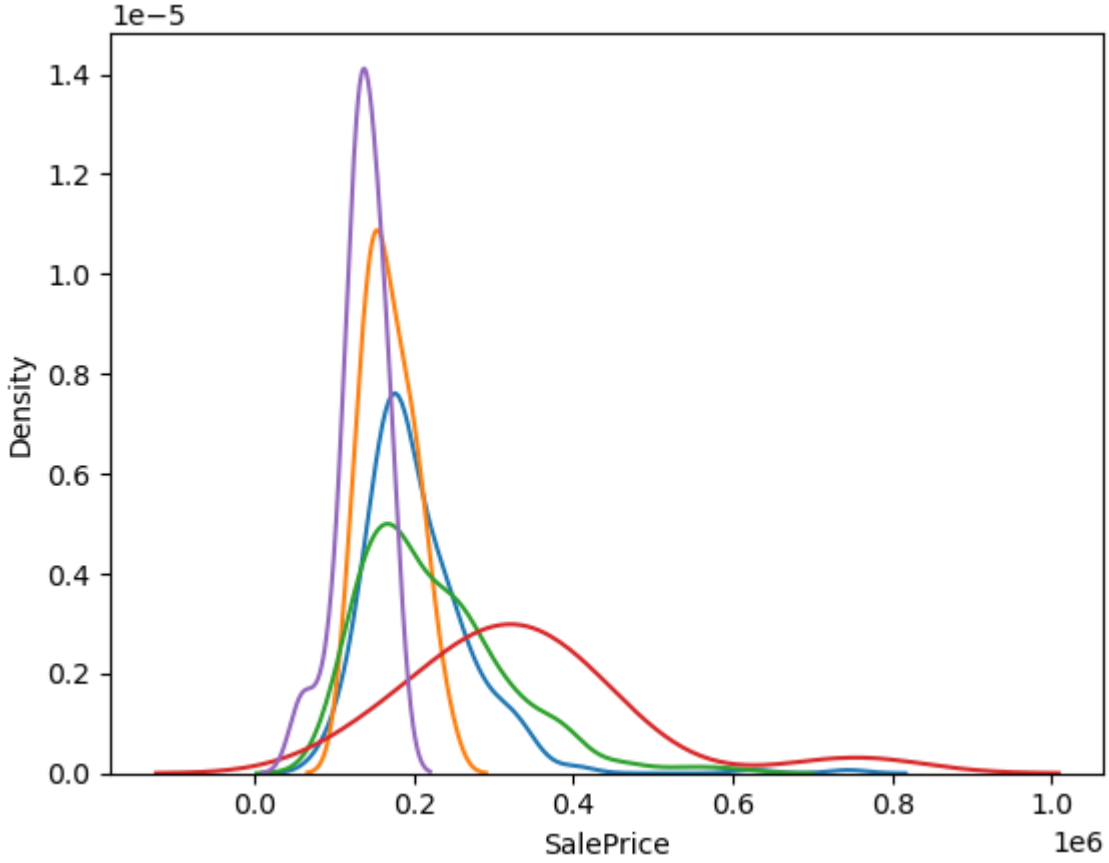
Out[33]:

	original	imputed
Gd	0.494272	0.394521
TA	0.412439	0.330822
Fa	0.040917	0.032192
Po	0.027823	0.022603
Ex	0.024550	0.019863

Compare category Fireplace before Imputation

In [35]:
```
for category in X_train['FireplaceQu'].dropna().unique(): sns.distplot(X_train[X_trai
plt.show()
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_10452\4070077928.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.githu
b.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  for category in X_train['FireplaceQu'].dropna().unique(): sns.distplot(X_train[X_t
rain['FireplaceQu'] == category]['SalePrice'],hist=False,label=category)
C:\Users\ASUS\AppData\Local\Temp\ipykernel_10452\4070077928.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.githu
b.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  for category in X_train['FireplaceQu'].dropna().unique(): sns.distplot(X_train[X_t
rain['FireplaceQu'] == category]['SalePrice'],hist=False,label=category)
C:\Users\ASUS\AppData\Local\Temp\ipykernel_10452\4070077928.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.githu
b.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  for category in X_train['FireplaceQu'].dropna().unique(): sns.distplot(X_train[X_t
rain['FireplaceQu'] == category]['SalePrice'],hist=False,label=category)
C:\Users\ASUS\AppData\Local\Temp\ipykernel_10452\4070077928.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.githu
b.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  for category in X_train['FireplaceQu'].dropna().unique(): sns.distplot(X_train[X_t
rain['FireplaceQu'] == category]['SalePrice'],hist=False,label=category)
C:\Users\ASUS\AppData\Local\Temp\ipykernel_10452\4070077928.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.githu
b.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  for category in X_train['FireplaceQu'].dropna().unique(): sns.distplot(X_train[X_t
rain['FireplaceQu'] == category]['SalePrice'],hist=False,label=category)
```



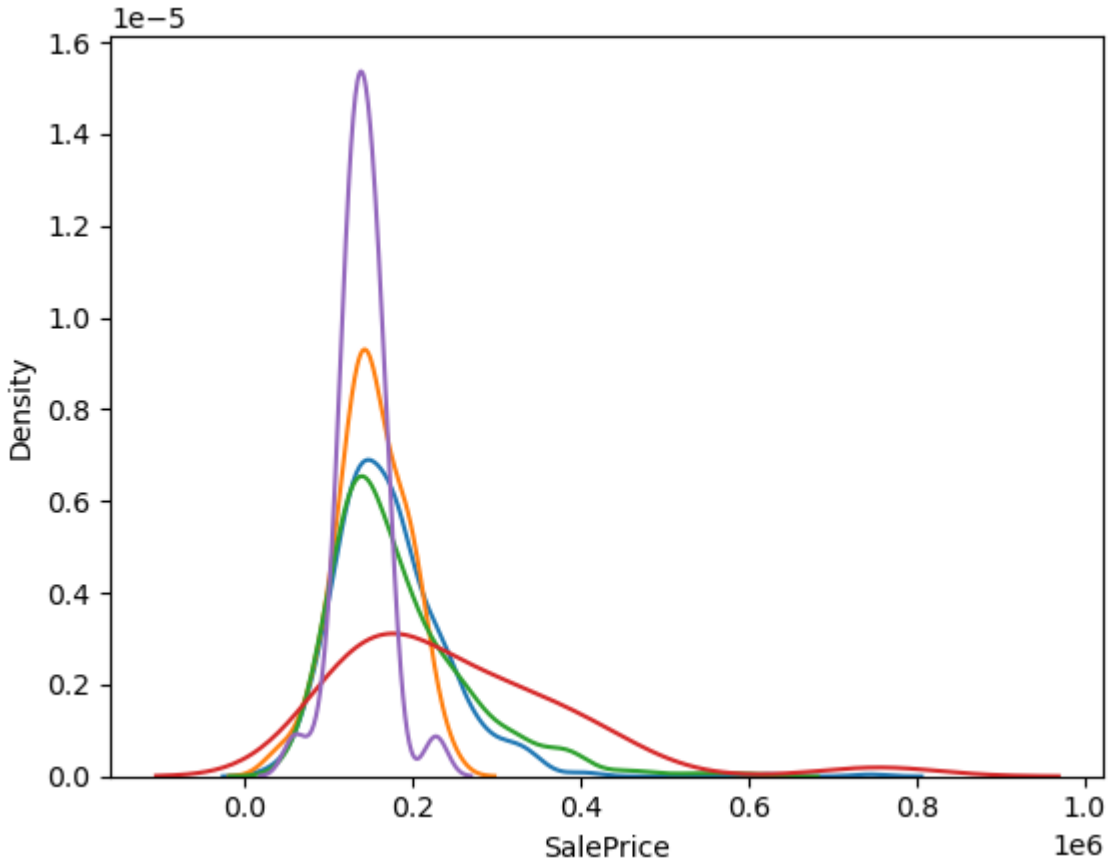
Compare category fireplace after Imputation

In [37]:
```
for category in X_train['FireplaceQu_imputed'].dropna().unique(): sns.distplot(X_trai
plt.show()
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_10452\1231138130.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.githu
b.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  for category in X_train['FireplaceQu_imputed'].dropna().unique(): sns.distplot(X_t
rain[X_train['FireplaceQu_imputed'] == category]['SalePrice'],hist=False,label=categ
ory)
C:\Users\ASUS\AppData\Local\Temp\ipykernel_10452\1231138130.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.githu
b.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  for category in X_train['FireplaceQu_imputed'].dropna().unique(): sns.distplot(X_t
rain[X_train['FireplaceQu_imputed'] == category]['SalePrice'],hist=False,label=categ
ory)
C:\Users\ASUS\AppData\Local\Temp\ipykernel_10452\1231138130.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.githu
b.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  for category in X_train['FireplaceQu_imputed'].dropna().unique(): sns.distplot(X_t
rain[X_train['FireplaceQu_imputed'] == category]['SalePrice'],hist=False,label=categ
ory)
C:\Users\ASUS\AppData\Local\Temp\ipykernel_10452\1231138130.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.githu
b.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  for category in X_train['FireplaceQu_imputed'].dropna().unique(): sns.distplot(X_t
rain[X_train['FireplaceQu_imputed'] == category]['SalePrice'],hist=False,label=categ
ory)
C:\Users\ASUS\AppData\Local\Temp\ipykernel_10452\1231138130.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.githu
```

b.com/mwaskom/de44147ed2974457ad6372750bbe5751)

```
for category in X_train['FireplaceQu_imputed'].dropna().unique(): sns.distplot(X_t
rain[X_train['FireplaceQu_imputed'] == category]['SalePrice'],hist=False,label=categ
ory)
```



In [ ]:

Data Science | 30 Days of Machine Learning | Day - 13

Educator Name: Nishant Dhote
Support Team: **+91-7880-113-112**

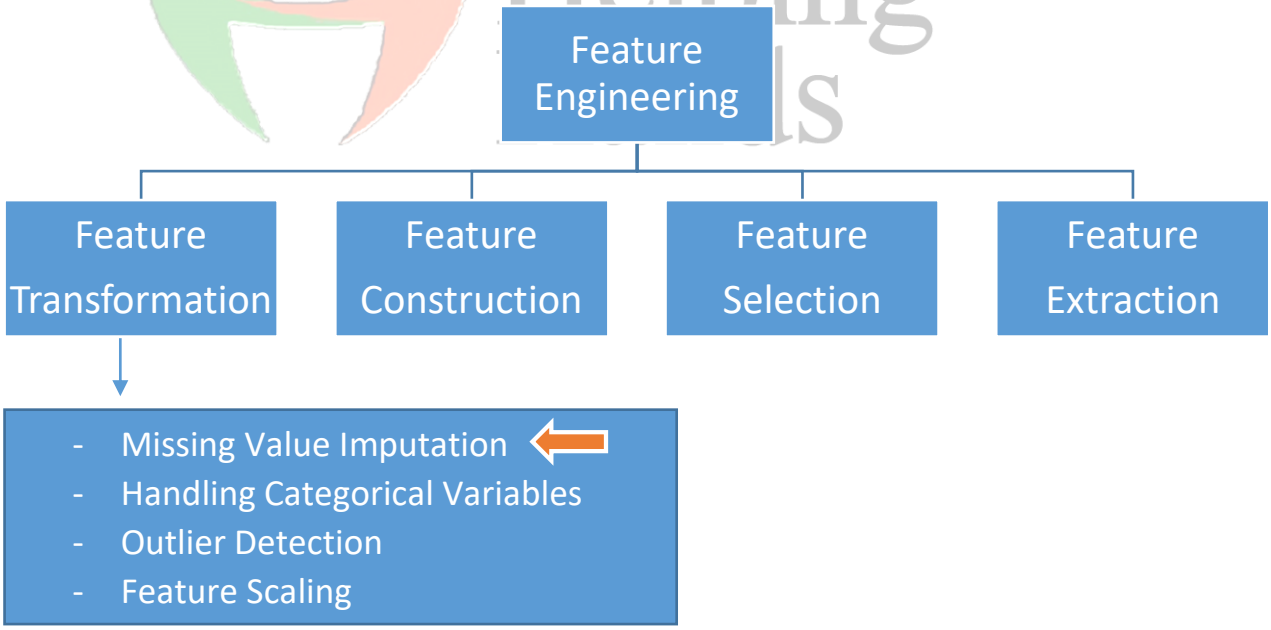
----Today Topics | Day 13----

Feature Engineering (Missing Value Imputation)

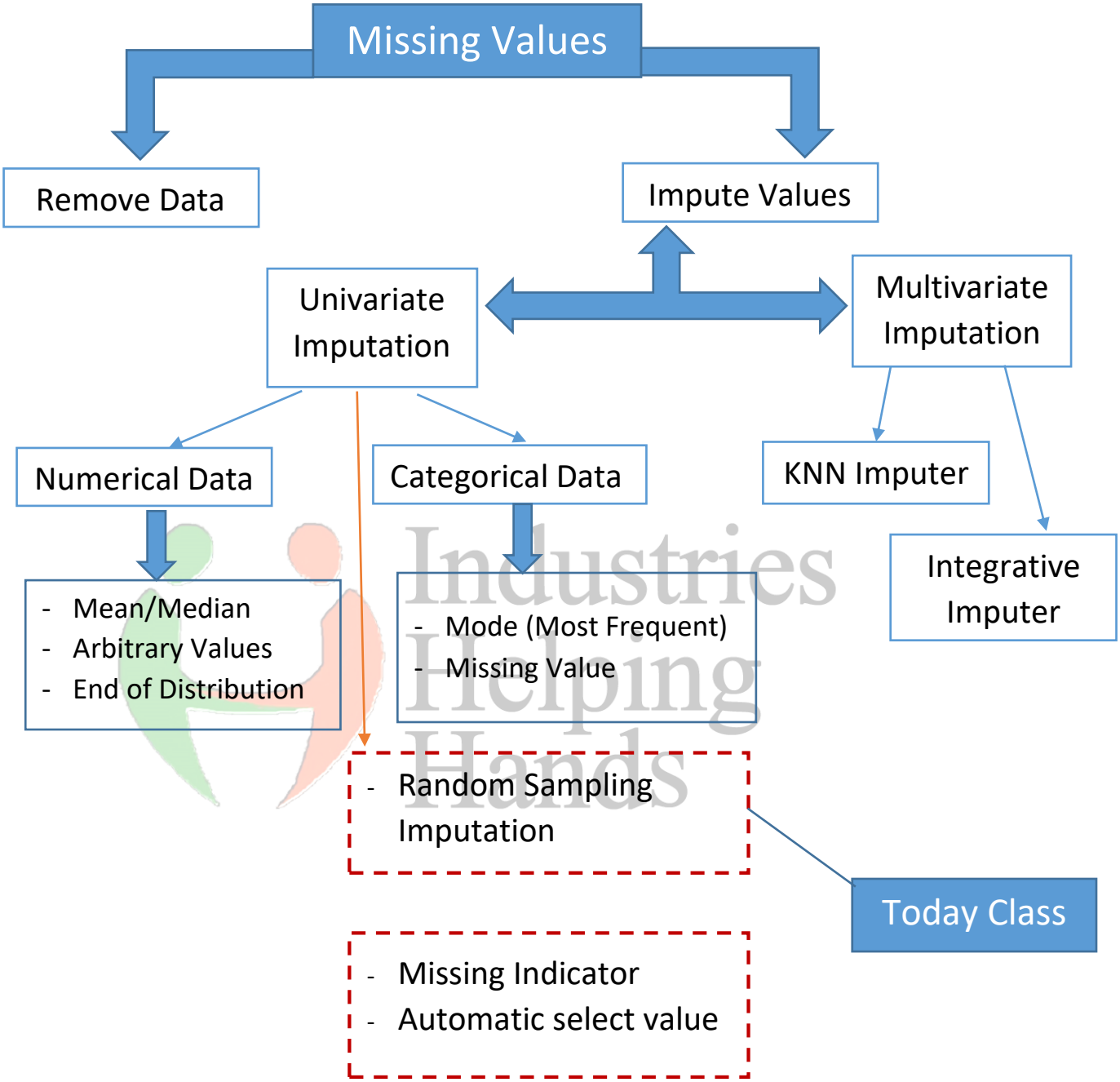
- Random Imputation in Univariate Imputation
- For Numerical Data
- For Categorical Data

- Missing Indicator in Univariate Imputation
- Automatic select value for Imputer parameter

Dataset Link GitHub: https://github.com/TheiScale/30_Days_Machine_Learning/



Today's Topics:



What is random sampling imputation?

Random sampling imputation consists of extracting random observations from the pool of available values in the variable. Random sampling imputation preserves the original distribution, which differs from the other imputation techniques we've discussed in this chapter and is suitable for numerical and categorical variables alike. In this recipe, we will implement random sample imputation with pandas and Feature-engine.

Example Numerical Data:

Age
15
18
21
N/A
22
15
18

Random Value Select (15/18/21/22)

Example Categorical Data:

Gender
M
F
M
N/A
F
M
F

Random Value Select (M/F)

Advantages & Disadvantage:

- Use pandas and implementation easy, not use SK learn uses and good for linear models and not use much in decision tree based algorithm technique.
- Complicated Deployment: Memory heavy for deployment, as we need to store the original training dataset to extract value replace from N/A.
- Well suited for linear algorithms as this does not destroy the distribution, regardless of the % of N/A.

Numeric Data: Titanic & Categorical Data: House Price

GitHub: https://github.com/TheiScale/30_Days_Machine_Learning/

<Start Coding | Random - Sample - imputation>

#Import Libraries

```
import numpy as np
import pandas as pd

from sklearn.model_selection import
train_test_split

import matplotlib.pyplot as plt
import seaborn as sns
```

#Import Dataset

```
df =
pd.read_csv('train.csv',usecols=['Age','Fare','Su
rvived'])
----
df.head()
```

#Check missing (null) value

```
df.isnull().mean() * 100
```

#Create X & Y

```
X = df.drop(columns=['Survived'])
y = df['Survived']
```

#Apply Train Test Split

```
X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.2,random_state=2
)
```

```
---
```

```
X_train
```

#New column create in Both Train & Test

```
X_train['Age_imputed'] = X_train['Age']
```

```
X_test['Age_imputed'] = X_test['Age']
```

```
----
```

```
X_test.tail()
```

```
X_train.head()
```

#Replace Value Age_imputed

```
X_train['Age_imputed'][X_train['Age_imputed'].isn
ull()] =
X_train['Age'].dropna().sample(X_train['Age'].isn
ull().sum()).values
```

```
X_test['Age_imputed'][X_test['Age_imputed'].isnul
l()] =
X_train['Age'].dropna().sample(X_test['Age'].isnu
ll().sum()).values
```

#Review Sample Random Generate Value

```
X_train['Age'].dropna().sample(1).values
```

```
X_train['Age'].isnull().sum()
```

```
X_train['Age'].dropna().sample(X_train['Age'].isn
ull().sum()).values
```

```
X_train
```

#Compare Original Age and Imputed Age

```
sns.distplot(X_train['Age'],label='Original',hist
 =False)
sns.distplot(X_train['Age_imputed'],label =
 'Imputed',hist=False)

plt.legend()
plt.show()
```

#Compare Variable Variance

```
print('Original variable variance: ',
 X_train['Age'].var())
print('Variance after random imputation: ',
 X_train['Age_imputed'].var())
```

#Random Sample imputation for Categorical Data

#Import Libraries

```
import numpy as np
import pandas as pd

from sklearn.model_selection import
train_test_split

import matplotlib.pyplot as plt
import seaborn as sns
```

#Import Dataset

```
df = pd.read_csv('house-
train.csv',usecols=['GarageQual','FireplaceQu',
'SalePrice'])
```

---

df.head()

#Check missing (null) value

```
df.isnull().mean() * 100
```

#Create X & Y

```
X = data
y = data['SalePrice']
```

 # Apply Train Test Split

```
X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.2,random_state=2
)
```

#New column create in Both Garage & Fire place

```
X_train['GarageQual_imputed'] = X_train['GarageQual']
X_test['GarageQual_imputed'] = X_test['GarageQual']

X_train['FireplaceQu_imputed'] =
X_train['FireplaceQu']
X_test['FireplaceQu_imputed'] = X_test['FireplaceQu']

----
X_train.sample(5)
```

#Replace Garage Value and Fireplace Imputed

```
X_train['GarageQual_imputed'][X_train['GarageQual_impu
ted'].isnull()] =
X_train['GarageQual'].dropna().sample(X_train['GarageQ
ual'].isnull().sum()).values
X_test['GarageQual_imputed'][X_test['GarageQual_impute
d'].isnull()] =
X_train['GarageQual'].dropna().sample(X_test['GarageQu
al'].isnull().sum()).values

X_train['FireplaceQu_imputed'][X_train['FireplaceQu_im
puted'].isnull()] =
X_train['FireplaceQu'].dropna().sample(X_train['Firepl
aceQu'].isnull().sum()).values
X_test['FireplaceQu_imputed'][X_test['FireplaceQu_impu
ted'].isnull()] =
```

```
X_train['FireplaceQu'].dropna().sample(X_test['Firepla
ceQu'].isnull().sum()).values
```

#Review Frequency in Garage Original & Imputed

```
temp = pd.concat(
        [
            X_train['GarageQual'].value_counts()
/ len(X_train['GarageQual'].dropna()),

X_train['GarageQual_imputed'].value_counts() /
len(X_train)
        ],
        axis=1)
temp.columns = ['original', 'imputed']
temp
```

#Review Frequency in Fireplace Original & Imputed

```
temp = pd.concat(
        [
            X_train['FireplaceQu'].value_counts()
/ len(X_train['FireplaceQu'].dropna()),

X_train['FireplaceQu_imputed'].value_counts() /
len(df)
        ],
        axis=1)

temp.columns = ['original', 'imputed']

temp
```

#Compare category Fireplace before Imputation

```
for category in
X_train['FireplaceQu'].dropna().unique():
    sns.distplot(X_train[X_train['FireplaceQu'] ==
category]['SalePrice'],hist=False,label=category)
plt.show()
```

#Compare category fireplace after Imputation

```
for category in
X_train['FireplaceQu_imputed'].dropna().unique():

sns.distplot(X_train[X_train['FireplaceQu_imputed
'] ==
category]['SalePrice'],hist=False,label=category)
plt.show()
```

Industries Helping Hands

Day 13: Curious Data Minds

- **Data Science and AI in the Travel Industry:**

Read Blog: https://www.altexsoft.com/blog/data-science-and-ai-in-the-travel-industry-9-real-life-use-cases/

https://economictimes.indiatimes.com/jobs/government-jobs/ayodhya-tourism-boom-may-create-150000-200000-direct-and-indirect-jobs/articleshow/107124078.cms?from=mdr

How OYO uses Data Analytics

As of January 2020, it has more than 43,000 properties and 10 lakh (1 million) rooms across 800 cities in 80 countries, including

India, Malaysia, the UAE, Nepal, China, Brazil, Mexico, the UK, Philippines, Japan, Saudi Arabia, Sri Lanka, Indonesia, Vietnam, and the United States.

Area served Asia, Europe and Americas
Revenue ₹4,157 crore

"Our data Analysts use natural curiosity and innovative tools to derive deep insights into customer behavior. These insights not only help us improve our service but also take effective business decisions,"

said Ritesh Agarwal, founder & CEO of OYO.

OYO users spent 3,232 years' worth of time on the OYO app in India – the highest in India in FY2021

Subah-Sham. Quite literally. The most popular time to make bookings on the OYO app were 11:00 AM – 1:00 PM and evening 6:00 PM – 9:00 PM

Fan Alert: A travel agent from India made 1193 bookings for an OYO in 2021

Data Science | 30 Days of Machine Learning | Day - 14

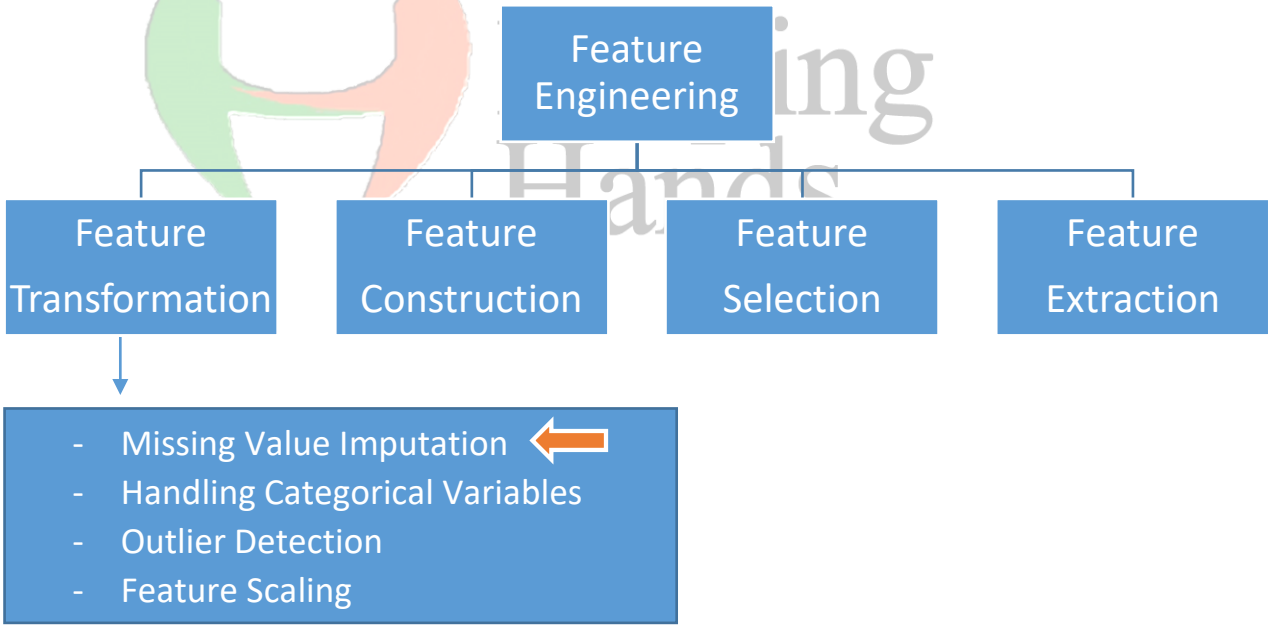
Educator Name: Nishant Dhote
Support Team: **+91-7880-113-112**

----Today Topics | Day 14----

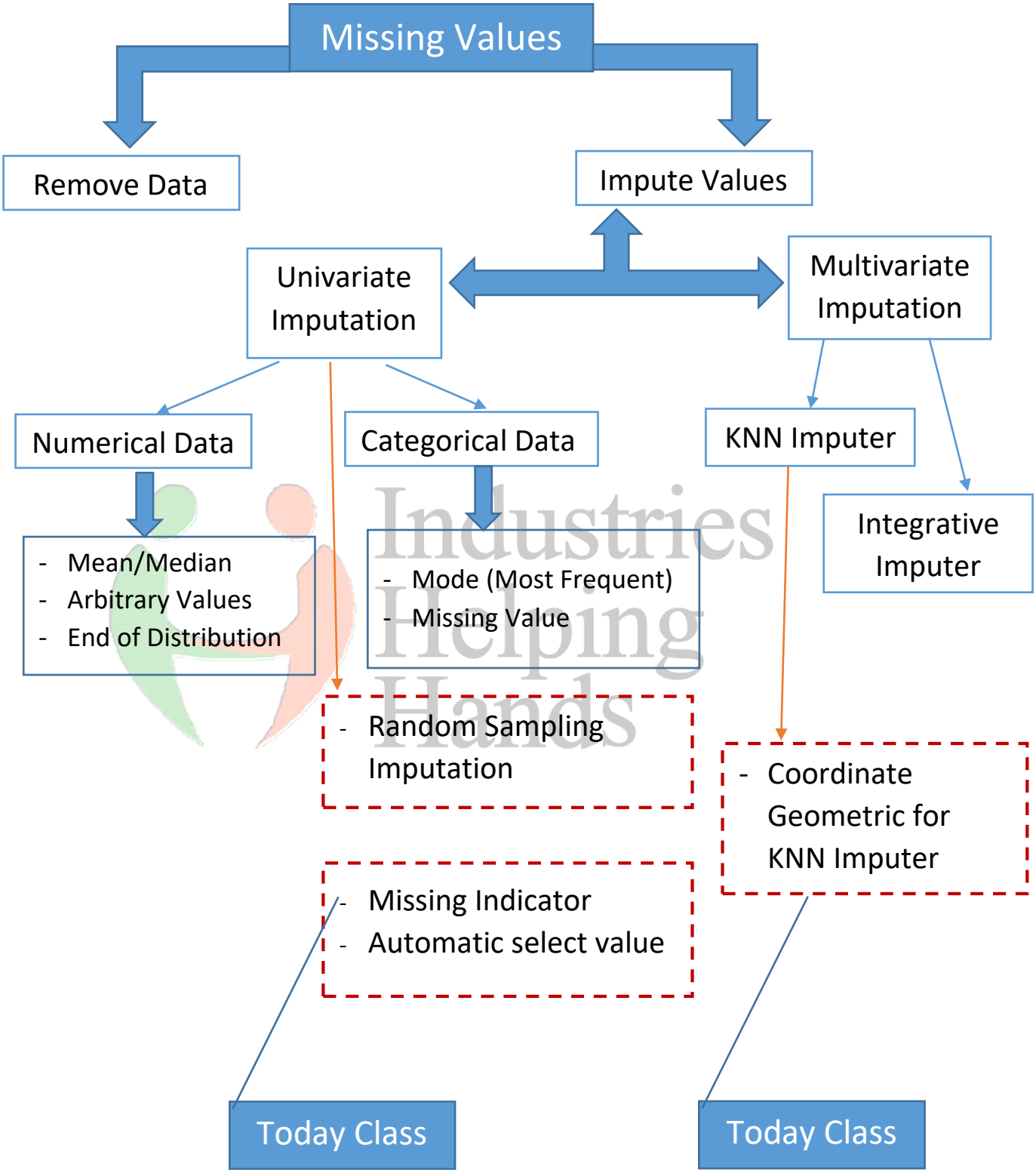
Feature Engineering (Missing Value Imputation)

- Missing Indicator in Univariate Imputation
- Automatic select value for Imputer parameter
- Coordinate Geometric for KNN Imputer
- Calculation in 2D and 3D Distance

Dataset Link GitHub: https://github.com/TheiScale/30_Days_Machine_Learning/



Today's Topics:



- **Missing Indicator in Univariate Imputation**

The Missing Indicator Method (MIM), which adds indicator variables to indicate the missing pattern, can be used in conjunction with imputation to improve model performance. While commonly used in data science, MIM is surprisingly understudied from an empirical and especially theoretical perspective. In this paper, we show empirically and theoretically that MIM improves performance for informative missing values, and we prove that MIM does not hurt linear models asymptotically for uninformative missing values.

Example:

Age	Fare	Age-NA
15	58	False
18	70	False
21	58	False
N/A	55	True
22	58	False
15	55	False
18	70	False

Industries Helping Hands

Today Class we use Titanic Dataset

GitHub: https://github.com/TheiScale/30_Days_Machine_Learning/

- <Start Coding | Missing Indicator Imputation >

#Import Libraries

```
import numpy as np
import pandas as pd

from sklearn.model_selection import
train_test_split

from sklearn.impute import
MissingIndicator,SimpleImputer
```

#Import Dataset

```
df =
pd.read_csv('train.csv',usecols=['Age','Fare','Su
rvived'])
----
df.head()
```

#Create X & Y

```
X = df.drop(columns=['Survived'])

y = df['Survived']
```

#Train & Test Split

```
X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.2,random_state=2
)
```

```
X_train.head()
```

#Review Without "Missing Indicator Method" Technique

```
si = SimpleImputer()
X_train_trf = si.fit_transform(X_train)
X_test_trf = si.transform(X_test)
```

```
X_train_trf
```

#Call Logistic Regression

```
from sklearn.linear_model import
LogisticRegression
clf = LogisticRegression()
clf.fit(X_train_trf,y_train)
y_pred = clf.predict(X_test_trf)

from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

#Define Missing Indicator

```
mi = MissingIndicator()

mi.fit(X_train)

----
MissingIndicator()

----
mi.features_
```

#Transform: Train Missing

```
X_train_missing = mi.transform(X_train)
----
X_train_missing
```

#Transform: Test Missing

```
X_test_missing = mi.transform(X_test)
----
X_test_missing
```

#Create New Column

```
X_train['Age_NA'] = X_train_missing
----
X_train
```

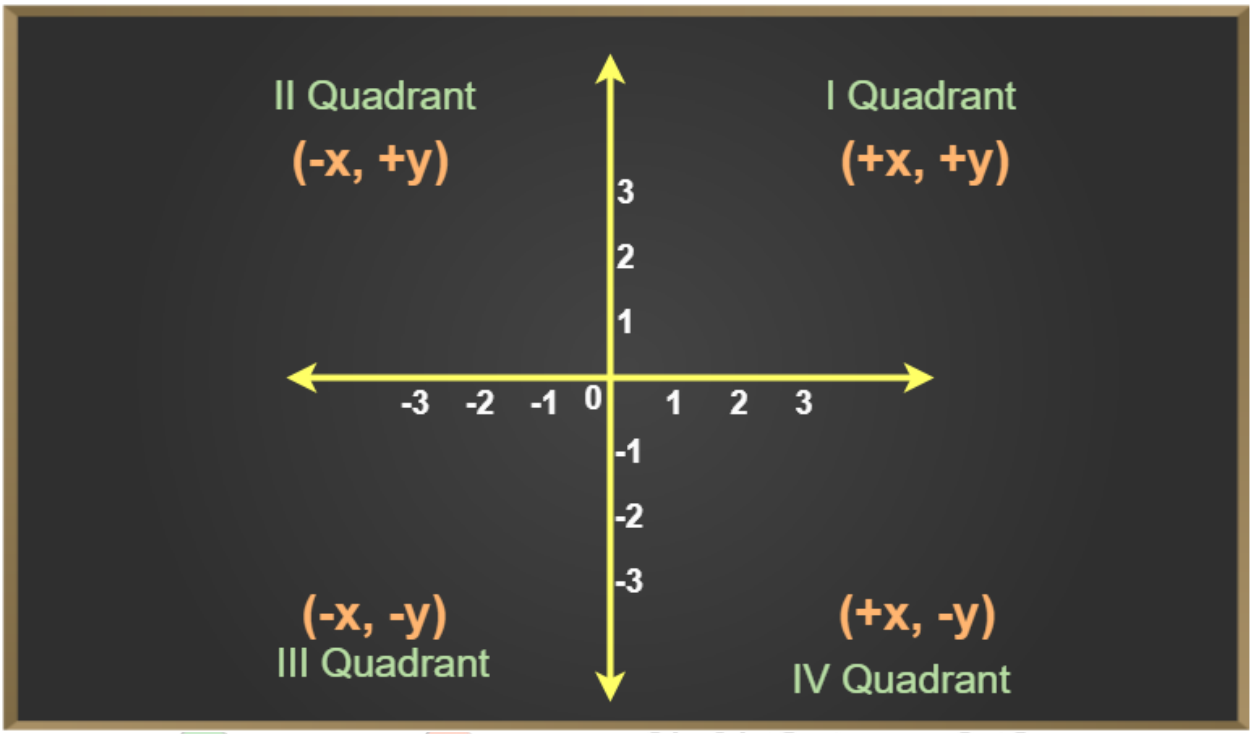
Industries Helping Hands

#Writing Code again and Check Accuracy

```
si = SimpleImputer()

X_train_trf2 = si.fit_transform(X_train)
X_test_trf2 = si.transform(X_test)


----
from sklearn.linear_model import
LogisticRegression

clf = LogisticRegression()

clf.fit(X_train_trf2,y_train)

y_pred = clf.predict(X_test_trf2)

from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Industries Helping Hands

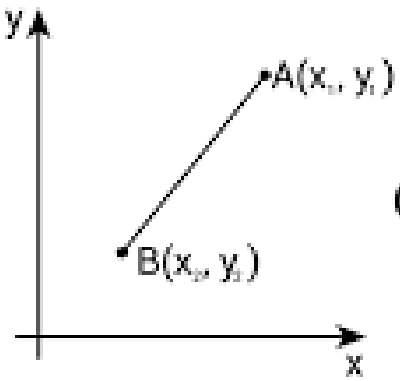
- Coordinate Geometric for KNN Imputer



Basic Formulas : https://www.geeksforgeeks.org/coordinate-geometry/

2D – Distance Formula:

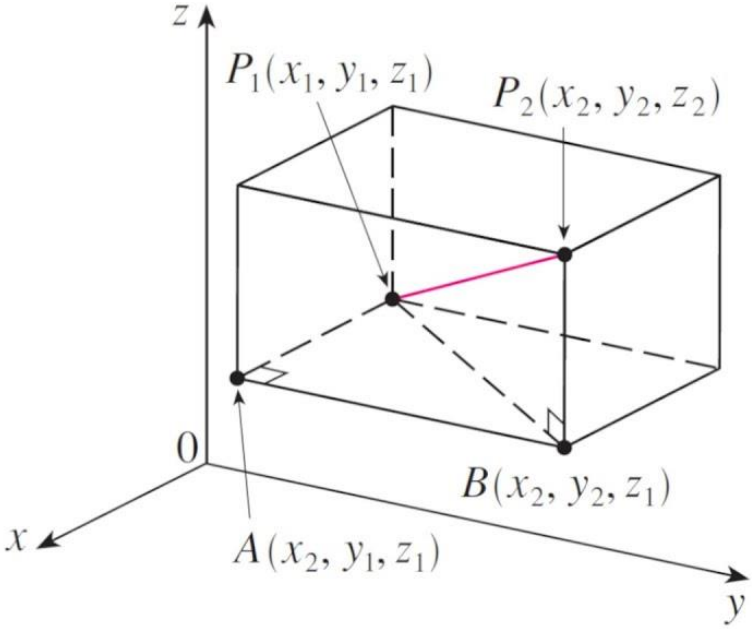
Distance Formula



$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

3D – Distance Formula:

$$|P_1P_2| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$



Distance Formula in 3D

Industries Helping Hands

distance between points

2D: $\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$

3D: $\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2}$

4D: $\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2 + (a_1 - a_0)^2}$

nD: $\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2 + (a_1 - a_0)^2 + \cdots}$

Example:

S.No	Variable 1	Variable 2	Variable 3	Variable 4
1	28	--	48	22
2	--	40	37	24
3	34	22	55	26
4	26	--	30	--
5	50	20	49	--

Distance between 2 points formula

Consider two points A (x_1, y_1) and B(x_2, y_2) on the given coordinate axis. The distance between these points is given as:

Helping Hands

Distance Between 2 Points Formula in 3D

The distance between two points $P(x_1, y_1, z_1)$ and $Q(x_2, y_2, z_2)$

$PQ = \sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2]}$

Distance Between 2 Points Formula Derivation in 3D

Let the points $P(x_1, y_1, z_1)$ and $Q(x_2, y_2, z_2)$ be referred to a system of rectangular axes OX,OY and OZ as shown in the figure.

sklearn.metrics.pairwise.nan_euclidean_distances

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.nan_euclidean_distances.html#sklearn.metrics.pairwise.nan_euclidean_distances

Industries Helping Hands

Day 14: Curious Data Minds

Suggest Topic in Comment Box?

Industries Helping Hands

Day 14 | Start Coding | Missing Indicator Imputation

Import Libraries

In [ ]:
```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.impute import MissingIndicator,SimpleImputer
```

Import Dataset

In [2]:
```
df = pd.read_csv('train.csv',usecols=['Age','Fare','Survived'])
```

In [3]:
```
df.head()
```

Out[3]:

	Survived	Age	Fare
0	0	22.0	7.2500
1	1	38.0	71.2833
2	1	26.0	7.9250
3	1	35.0	53.1000
4	0	35.0	8.0500

Create X & Y

In [4]:
```
X = df.drop(columns=['Survived'])
y = df['Survived']
```

Train & Test Split

In [5]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [6]:
```
X_train.head()
```

Out[6]:

	Age	Fare
30	40.0	27.7208
10	4.0	16.7000
873	47.0	9.0000
182	9.0	31.3875
876	20.0	9.8458

Review Without “Missing Indicator Method” Technique

In [7]:
```
si = SimpleImputer()
X_train_trf = si.fit_transform(X_train)
X_test_trf = si.transform(X_test)
```

In [8]:
```
X_train_trf
```

Out[8]:
```
array([[ 40.        ,  27.7208    ],
       [  4.        ,  16.7       ],
       [ 47.        ,   9.        ],
       ...,
       [ 71.        ,  49.5042    ],
       [ 29.78590426, 221.7792    ],
       [ 29.78590426,  25.925     ]])
```

Call Logistic Regression

In [9]:
```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression()

clf.fit(X_train_trf,y_train)

y_pred = clf.predict(X_test_trf)

from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[9]:
```
0.6145251396648045
```

Define Missing Indicator

In [10]:
```
mi = MissingIndicator()

mi.fit(X_train)
```

Out[10]:
```
▾ MissingIndicator
MissingIndicator()
```

In [11]:
```
MissingIndicator()
```

Out[11]:
```
▾ MissingIndicator
MissingIndicator()
```

In [12]:
```
mi.features_
```

Out[12]:
```
array([0], dtype=int64)
```

In [13]:
```
X_train_missing = mi.transform(X_train)
```

In [14]:
```
X_train_missing
```

```
       [False],
       [ True],
       [False],
       [False],
       [False
```

In [18]:
```
X_test
```

Out[18]:

	Age	Fare
707	42.0	26.2875
37	21.0	8.0500
615	24.0	65.0000
169	28.0	56.4958
68	17.0	7.9250
...	...	...
89	24.0	8.0500
80	22.0	9.0000
846	NaN	69.5500
870	26.0	7.8958
251	29.0	10.4625

179 rows × 2 columns

Create New Column

In [19]:
```
X_test['Age_NA'] = X_test_missing
```

In [20]:
```
X_train
```

Out[20]:

	Age	Fare	Age_NA
30	40.0	27.7208	False
10	4.0	16.7000	False
873	47.0	9.0000	False
182	9.0	31.3875	False
876	20.0	9.8458	False
...	...	...	...
534	30.0	8.6625	False
584	NaN	8.7125	True
493	71.0	49.5042	False
527	NaN	221.7792	True
168	NaN	25.9250	True

712 rows × 3 columns

Writing Code again and Check Accuracy

In [23]:
```
si = SimpleImputer()

X_train_trf2 = si.fit_transform(X_train)
X_test_trf2 = si.transform(X_test)
```

In [24]:
```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression()
clf.fit(X_train_trf2,y_train)
y_pred = clf.predict(X_test_trf2)
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[24]:
```
0.6312849162011173
```

In [ ]:

FORMERLY Industries Helping Hands

THE iSCALE

Data Science | 30 Days of Machine Learning | Day - 15

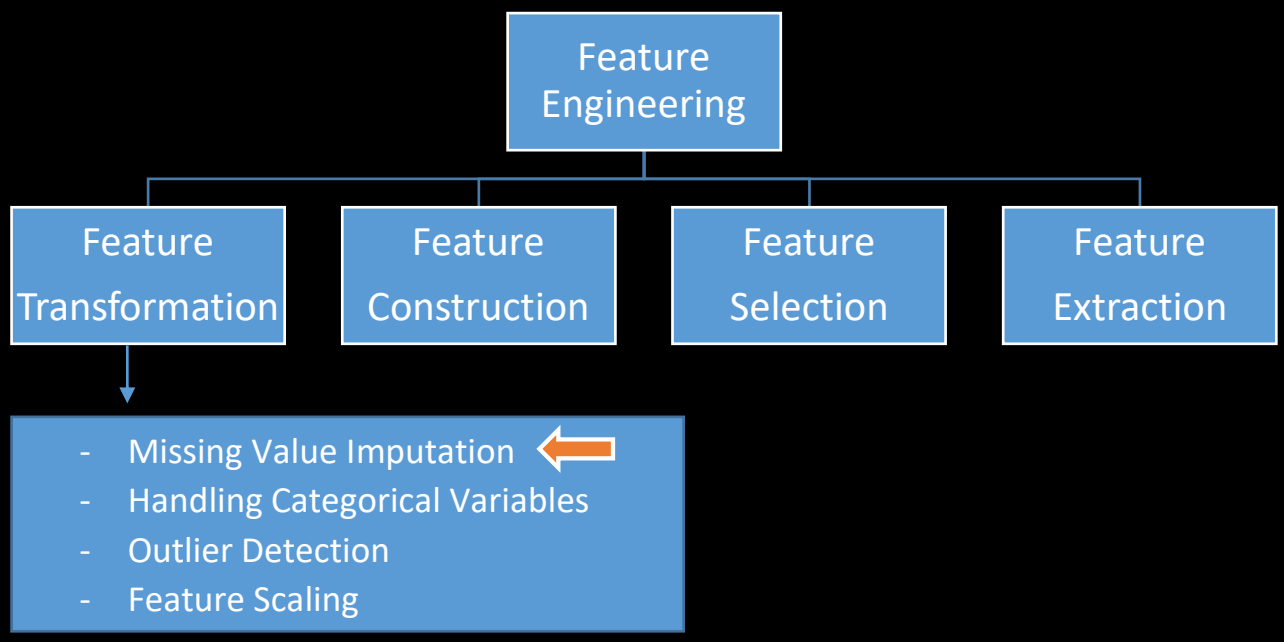
Educator Name: Nishant Dhote
Support Team: **+91-7880-113-112**

----Today Topics | Day 15----

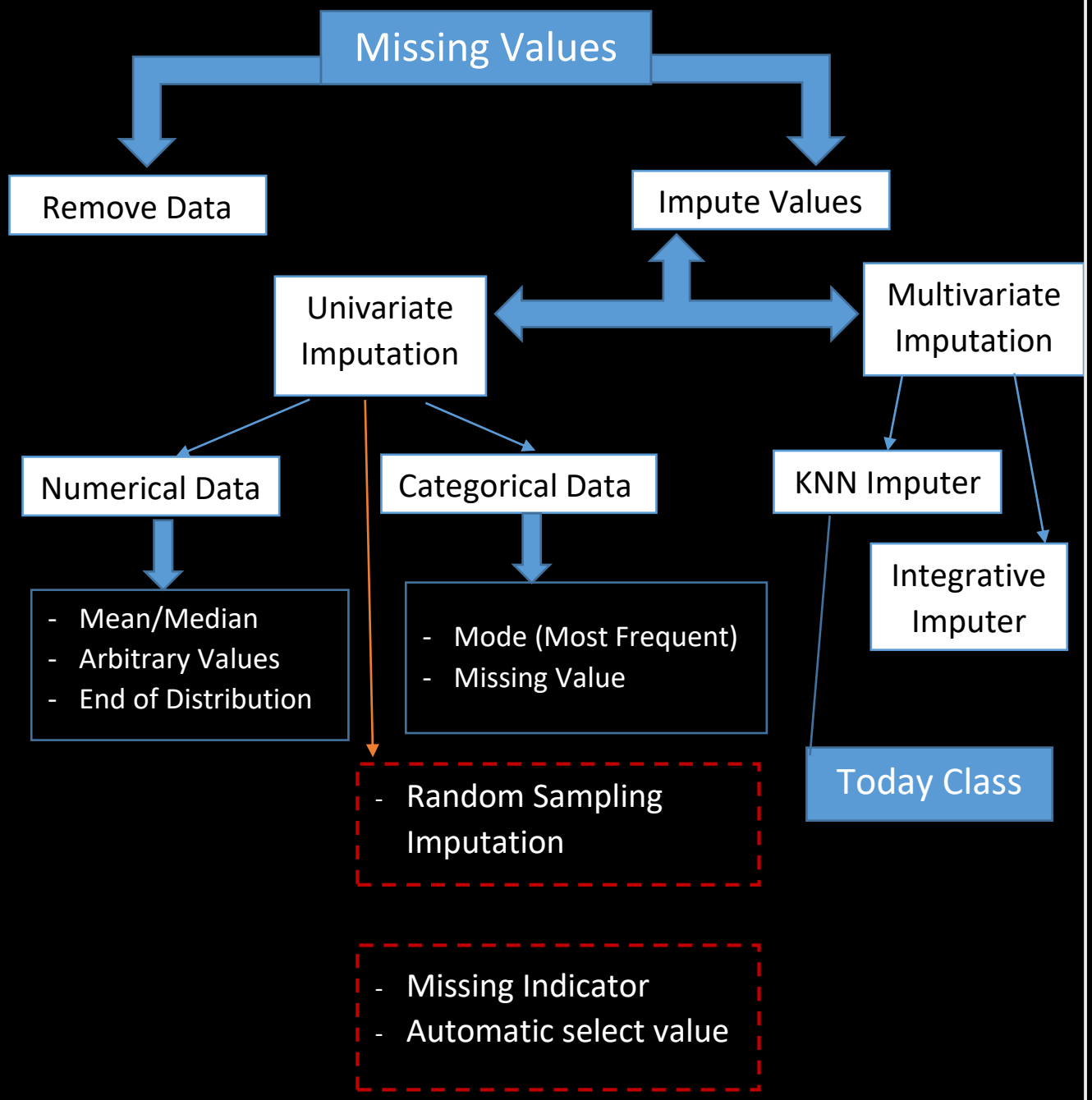
Feature Engineering (Missing Value Imputation)

- KNN Imputer
- K-Nearest Neighbour Calculation Method
- What is Euclidean Distance in Machine Learning?
- How to find K nearest neighbour?
- Find missing imputation value?

Dataset Link GitHub: https://github.com/TheiScale/30_Days_Machine_Learning/



Today's Topics:



- **KNN Imputer:** KNN imputer is a scikit-learn class used to fill out or predict the missing values in a dataset. It is a more useful method which works on the basic approach of the KNN algorithm rather than the naive approach (straightforward attempt to solve solution) of filling all the values with mean or the median. In this approach, we specify a distance from the missing values which is also known as the K parameter. The missing value will be predicted in reference to the mean of the neighbours.

S.No	Variable 1	Variable 2	Variable 3	Variable 4
1	28	--	48	22
2	--	40	37	24
3	34	22	55	26
4	26	--	30	--
5	50	20	49	--

K-Nearest Neighbour: K= The number of nearest neighbours to a new unknown variable that has to be predicted or classified is denoted by the symbol 'K'.

sklearn.metrics.pairwise.nan_euclidean_distances

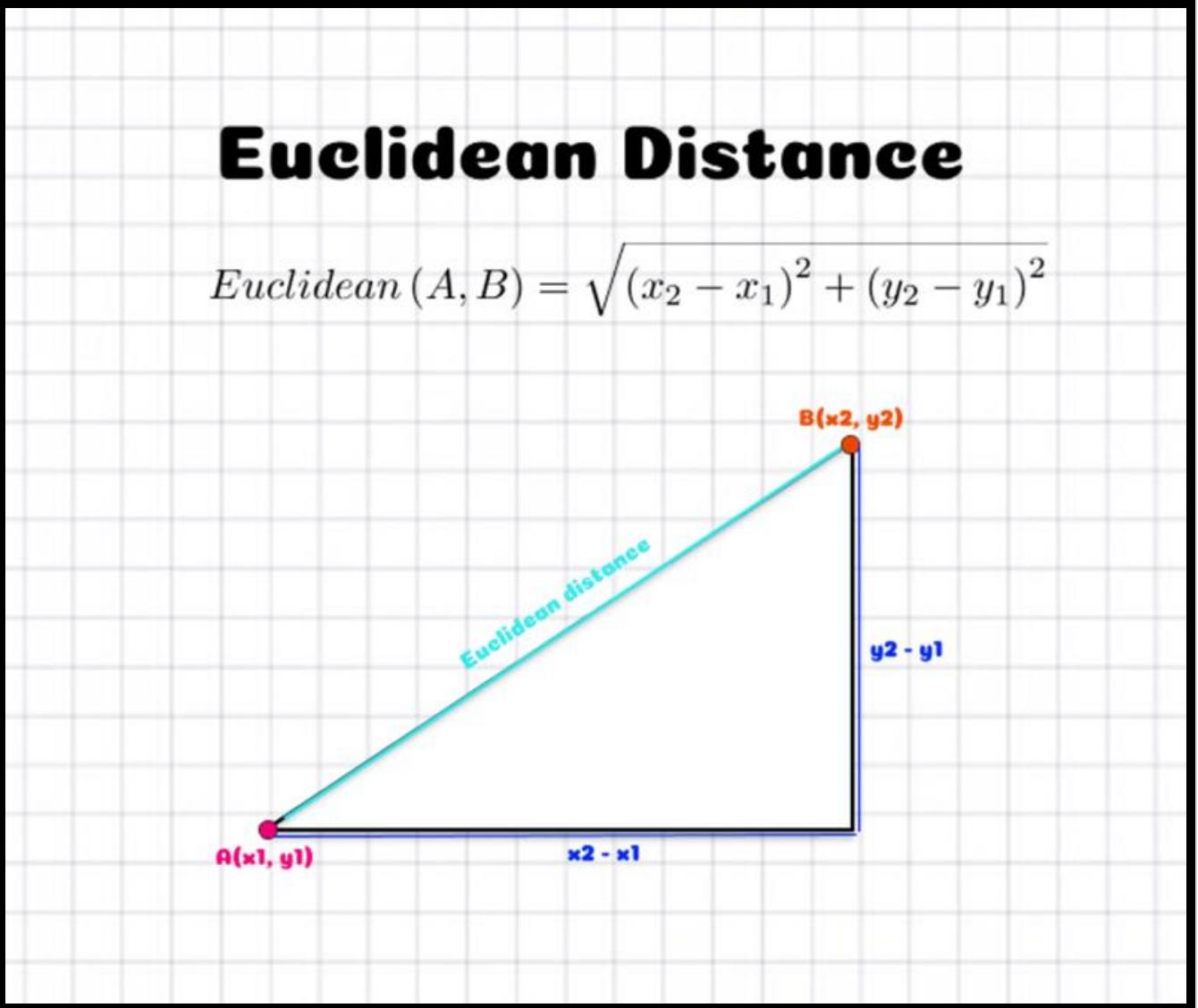
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.nan_euclidean_distances.html#sklearn.metrics.pairwise.nan_euclidean_distances

dist(x,y) = sqrt(weight * sq. distance from present coordinates) where

weight = Total # of coordinates / # of present coordinates

What is Euclidean Distance in Machine Learning?

Euclidean distance is used in many machine learning algorithms as a default distance metric to measure the similarity between two recorded observations. However, the observations to be compared must include features that are continuous and have numeric variables like weight, height, salary, etc.



2 Step We follow:

1. Find "K" nearest neighbour?
2. Find the value?

Example 1: Calculation between Row 1 and Row 2

S.No	Feature 1	Feature 2	Feature 3	Feature 4
1	28	--	48	22
2	--	40	37	24
3	34	22	55	26
4	26	--	30	--
5	50	20	49	--

For this value?

Non-Eclid Dis = ?

$$= \sqrt{w_j \times (40-2)^2 + (37-49)^2 + (24-22)^2}$$

$$w_j = \frac{\text{total No of Coordinate}}{\text{No of Present Coordinate}} = \frac{3}{2}$$

$$= \sqrt{\frac{3}{2} \times (-11)^2 + (2)^2}$$

$$= \sqrt{1.5 \times [121 + 4]} = \sqrt{1.5 \times 125}$$

$$= \sqrt{187.5}$$

$$\boxed{= 13.69}$$

Example 2: Calculation between Row 2 and Row 3

S.No	Feature 1	Feature 2	Feature 3	Feature 4
1	28	--	48	22
2	--	40	37	24
3	34	22	55	26
4	26	--	30	--
5	50	20	49	--

Non E. →

$$= \sqrt{\frac{3}{3} \times \left[(22-40)^2 + (55-37)^2 + (26-24)^2\right]}$$

$$= \sqrt{1 \times \left[(-18)^2 + (18)^2 + (2)^2\right]}$$

$$= \sqrt{324 + 225 + 4} = \sqrt{553}$$

$$\boxed{= 23.5} \rightarrow \text{Row 2, Row 3}$$

Example 3: Calculation between Row 2 and Row 4

S.No	Feature 1	Feature 2	Feature 3	Feature 4
1	28	--	48	22
2	--	40	37	24
3	34	22	55	26
4	26	--	30	--
5	50	20	49	--

$\rightarrow x = ?$

$$= \sqrt{\frac{3}{1} \times \left((x-40)^2 + (30-37)^2 + (x-24)^2 \right)}$$

$$= \sqrt{\frac{3}{1} \times 49}$$

$$= \sqrt{147}$$

$$\boxed{= 12.12}$$

Example 4: Calculation between Row 2 and Row 5

S.No	Feature 1	Feature 2	Feature 3	Feature 4
1	28	--	48	22
2	--	40	37	24
3	34	22	55	26
4	26	--	30	--
5	50	20	49	--

$x = 2$

$$= \sqrt{w_2 \times [(20-40)^2 + (49-37)^2 + \alpha}$$

$$= \sqrt{\frac{3}{2} \times [400 + 144]}$$

$$= \sqrt{1.5 \times 544}$$

$$= \sqrt{816} = 28.5$$

All 4 Euclidean Distance Example:

S.No	Feature 1	Feature 2	Feature 3	Feature 4
1	28	--	48	22
2	--	40	37	24
3	34	22	55	26
4	26	--	30	--
5	50	20	49	--

Example → 1 (Row 1 & 2)

$$= \sqrt{\frac{3}{2}(37-48)^2 + (24-22)^2}$$

$$= \sqrt{1.5 \times [(-11)^2 + (2)^2]}$$

$$= \sqrt{1.5 \times [121+4]}$$

$$= \sqrt{185} \rightarrow 13.69$$

Example 2 (Row 2 & 3)

$$= \sqrt{\frac{3}{3} \times (22-40)^2 + (55-37)^2 + (26-24)^2}$$

$$= \sqrt{(-18)^2 + (18)^2 + (2)^2}$$

$$= \sqrt{324 + 225 + 4} = 23.5$$

Example - 3 (Row 2 & 4)

$$= \sqrt{\frac{3}{1} \times (30-37)^2}$$

$$= \sqrt{3 \times 49}$$

$$= \sqrt{147}$$

$$= 12.12$$

Example-4 (Row 2 & 5)

$$= \sqrt{\frac{3}{2}[(20-40)^2 + (49-37)^2]}$$

$$= \sqrt{1.5[400 + 144]}$$

$$= \sqrt{1.5 \times 544} = \sqrt{816}$$

$$= 28.5$$

S.No	Feature 1	Feature 2	Feature 3	Feature 4
1	28	--	48	22
2	--	40	37	24
3	34	22	55	26
4	26	--	30	--
5	50	20	49	--

2 Step We follow:

Find "K" nearest neighbour?
Find the value?

v

IF = K = 2

Row 1 13.69

Row-4 12.12

$$\frac{28+26}{2} = \frac{54}{2} = 27$$

missing value

K = 3

Row 1 13.69

Row 4 12.12

Row 23.5

$$\frac{28+26+34}{3} = 29.3$$

<Start-Coding>

```
#Import Library

import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.impute import KNNImputer,SimpleImputer
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score


#Import Dataset

df =
pd.read_csv('train.csv')[['Age','Pclass','Fare','Survi
ved']]

----
df.sample(10)


#Check Missing Value

df.isnull().mean() * 100


#Define X & Y

X = df.drop(columns=['Survived'])
y = df['Survived']
```

#Train Test Split

```
X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.2,random_state=2)

----

X_train
```

#Apply KNN Imputer

```
knn = KNNImputer(n_neighbors=1,weights='distance')

X_train_trf = knn.fit_transform(X_train)
X_test_trf = knn.transform(X_test)
```

#Convert in Data Frame

```
pd.DataFrame(X_train_trf,columns=X_train.columns)
```

#Apply Logistic Regression

```
lr = LogisticRegression()

lr.fit(X_train_trf,y_train)

y_pred = lr.predict(X_test_trf)

accuracy_score(y_test,y_pred)
```

FORMERLY Industries Helping Hands

THE iSCALE

Day 15: Curious Data Minds

What is a machine learning role in healthcare sector?

Read Blog: https://builtin.com/artificial-intelligence/machine-learning-healthcare

https://www.bluebash.co/blog/ai-healthcare-future-trends-2024/

Day 15 | KNN Imputer

Import Library

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.impute import KNNImputer,SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Import Dataset

In [2]:
```
df = pd.read_csv('train.csv')[['Age','Pclass','Fare','Survived']]
```

In [3]:
```
df.head()
```

Out[3]:

	Age	Pclass	Fare	Survived
0	22.0	3	7.2500	0
1	38.0	1	71.2833	1
2	26.0	3	7.9250	1
3	35.0	1	53.1000	1
4	35.0	3	8.0500	0

Check Missing Value

In [4]:
```
df.isnull().mean() * 100
```

Out[4]:
```
Age         19.86532
Pclass       0.00000
Fare         0.00000
Survived     0.00000
dtype: float64
```

Define X & Y

In [5]:
```
X = df.drop(columns=['Survived'])
y = df['Survived']
```

Train Test Split

In [6]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [7]:
```
X_train
```

Out[7]:

	Age	Pclass	Fare
30	40.0	1	27.7208
10	4.0	3	16.7000
873	47.0	3	9.0000
182	9.0	3	31.3875
876	20.0	3	9.8458
...	...	...	...
534	30.0	3	8.6625
584	NaN	3	8.7125
493	71.0	1	49.5042
527	NaN	1	221.7792
168	NaN	1	25.9250

712 rows × 3 columns

Apply KNN Imputer

In [19]:
```
knn = KNNImputer(n_neighbors=10,weights='distance')
X_train_trf = knn.fit_transform(X_train)
X_test_trf = knn.transform(X_test)
```

Convert in Data Frame

In [16]:
```
pd.DataFrame(X_train_trf,columns=X_train.columns)
```

Out[16]:

	Age	Pclass	Fare
0	40.000000	1.0	27.7208
1	4.000000	3.0	16.7000
2	47.000000	3.0	9.0000
3	9.000000	3.0	31.3875
4	20.000000	3.0	9.8458
...	...	...	...
707	30.000000	3.0	8.6625
708	26.151292	3.0	8.7125
709	71.000000	1.0	49.5042
710	32.666667	1.0	221.7792
711	49.762895	1.0	25.9250

712 rows × 3 columns

In [17]:
```
pd.DataFrame(X_train_trf,columns=X_train.columns).sample(10)
```

Out[17]:

	Age	Pclass	Fare
607	63.0	3.0	9.5875
394	52.0	1.0	30.5000
537	16.0	2.0	10.5000
694	22.0	3.0	7.8958
373	19.0	2.0	36.7500
364	19.0	2.0	13.0000
359	21.0	3.0	7.7500
583	16.0	3.0	8.0500
460	33.0	3.0	15.8500
160	22.0	3.0	9.3500

Apply Logistic Regression

In [20]:
```
lr = LogisticRegression()
lr.fit(X_train_trf,y_train)
y_pred = lr.predict(X_test_trf)
accuracy_score(y_test,y_pred)
```

Out[20]: 0.7039106145251397

In [ ]:

FORMERLY Industries Helping Hands

THE iSCALE

Data Science | 30 Days of Machine Learning | Day - 16

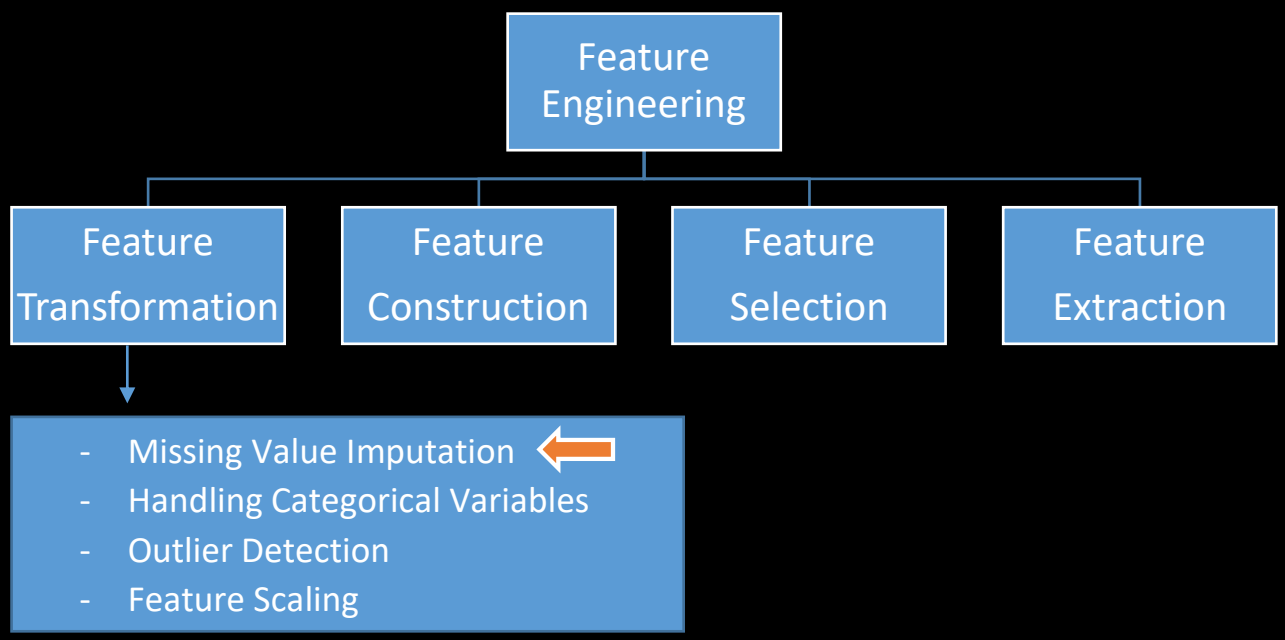
Educator Name: Nishant Dhote
Support Team: **+91-7880-113-112**

----Today Topics | Day 16----

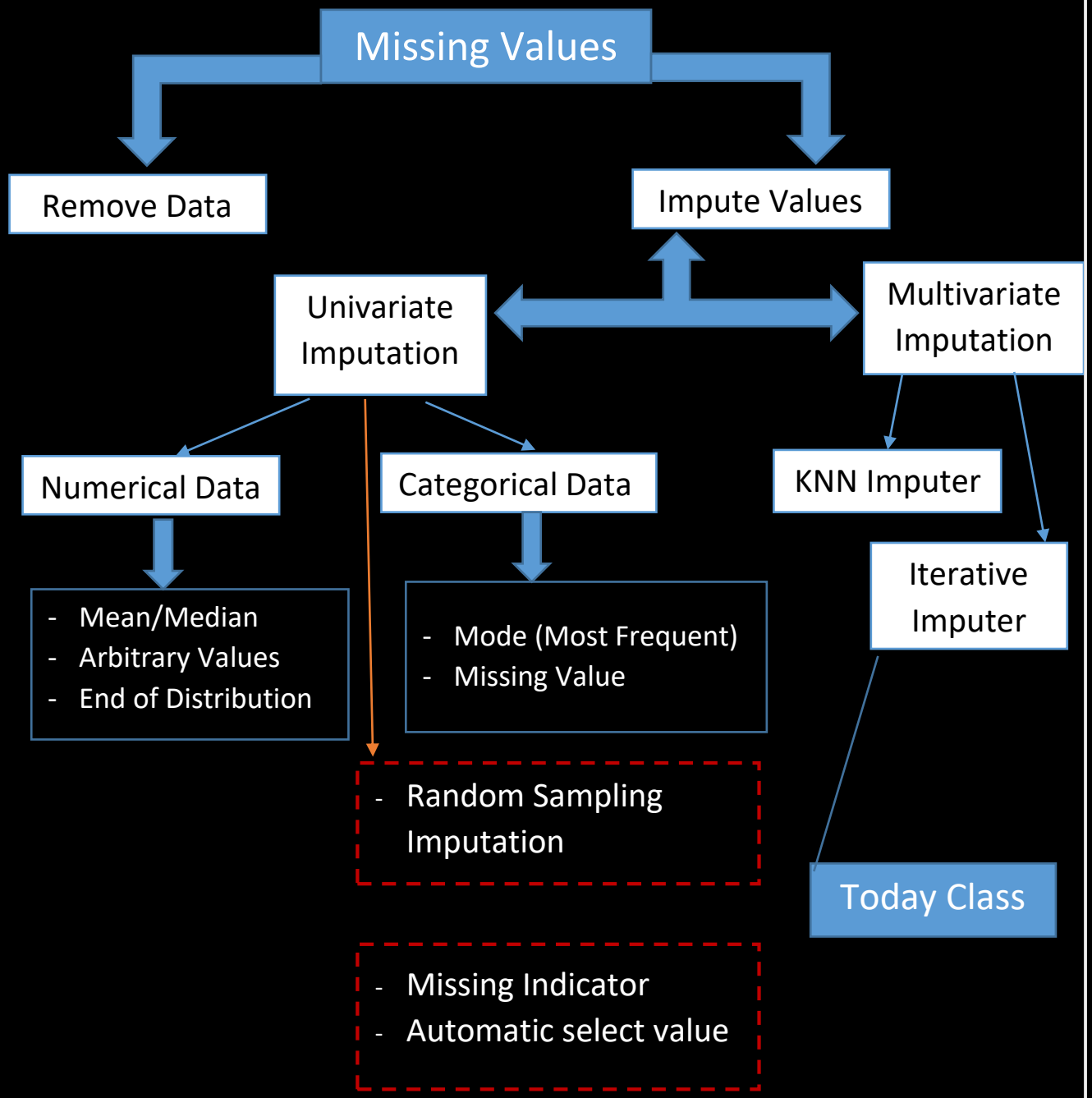
Feature Engineering (Missing Value Imputation)

- Iterative Imputer
- MICE- Multiple Imputation by Chained Equations
- Missing Completely at Random (MCAR)
- Missing at Random (MAR)
- Missing Not at Random (MNAR)
- Find Predictive Value for Iterative Imputer Technique

Dataset Link GitHub: https://github.com/TheiScale/30_Days_Machine_Learning/



Today's Topics:



- **Multivariate Imputation:** Multiple imputations can be used in cases where the data are MCAR, MAR, and even when the data are MNAR. Multiple imputation methods are known as multivariate imputation.

- **Iterative Imputer:** Iterative Imputer is a multivariate imputing strategy that models a column with the missing values (target variable) as a function.

========== ==========

MICE stands for "Multivariate Imputation by Chained Equations":

Having a better understanding of the reasoning for the missing ness of your data can help you determine what type of imputation method you can use. In general, there are three types of missing data.

1. Missing Completely at Random (MCAR):
2. Missing at Random (MAR):
3. Missing Not at Random (MNAR):

Missing Completely at Random (MCAR): The values in your dataset are missing completely at random. This is when there is no clear reasoning as to why a certain value in your dataset is missing. (There is no reason why your data was not collected.)

Missing at Random (MAR): The values in your dataset are missing at random. This is when we can determine some correlation to why the data value may be missing. An example of this is if a certain question in a survey is blank for multiple surveys of the same gender. A way in which we can handle this situation is *by using other features* to do a grouped mean/median replacement— the data missing is still recoverable.

Missing Not at Random (MNAR): The values in your dataset are not missing at random. This is when we can see a clear pattern to the missing values. An example of this is if a certain question category in a survey is left blank by surveys because of the question itself, as it may be a sensitive question to the surveyed (the missing ness depends on the missing data) — the data missing will be hard to recover unless further research is done. Unlike the other two types of missing data, MNAR is non ignorable.

Note:
MICE "Multivariate Imputation by Chained Equations" generally used in Missing at Random (MAR):

- MICE used in Input column

Today Class we use Dataset 50 Start-up:
https://www.kaggle.com/datasets/abhishek14398/50startups/data

1. Import – Original Data / Screenshot Image

50

	R&D Spend	Administration	Marketing Spend	Profit
21	8.0	15.0	30.0	11.0
37	4.0	5.0	20.0	9.0
2	15.0	10.0	41.0	19.0
14	12.0	16.0	26.0	13.0
44	2.0	15.0	3.0	7.0

5 Sample

2. Remove the Target Column

Profit

	R&D Spend	Administration	Marketing Spend
21	8.0	15.0	30.0
37	4.0	5.0	20.0
2	15.0	10.0	41.0
14	12.0	16.0	26.0
44	2.0	15.0	3.0

3. NaN value import (Manipulate the Data)

	R&D Spend	Administration	Marketing Spend
21	8.0	15.0	30.0
37	NaN	5.0	20.0
2	15.0	10.0	41.0
14	12.0	NaN	26.0
44	2.0	15.0	NaN

Step 1 - Impute all missing values with mean

	R&D Spend	Administration	Marketing Spend
21	8.00	15.00	30.00
37	9.25	5.00	20.00
2	15.00	10.00	41.00
14	12.00	11.25	26.00
44	2.00	15.00	29.25

Step 2 - Remove the column 1 imputed value (Left to Right)

L → R

	R&D Spend	Administration	Marketing Spend
21	8.0	15.00	30.00
37	NaN	5.00	20.00
2	15.0	10.00	41.00
14	12.0	11.25	26.00
44	2.0	15.00	29.25

I/P Data

Cor O/P Data

Training Data in X (Training Input)

In [19]:
```
X = df1.iloc[[0,2,3,4],1:3]
X
```

Out[19]:

	Administration	Marketing Spend
21	15.00	30.00
2	10.00	41.00
14	11.25	26.00
44	15.00	29.25

Training Data in Y (Corresponding Output)

In [18]:
```
y = df1.iloc[[0,2,3,4],0]
y
```

Out[18]:
```
21     8.0
2     15.0
14    12.0
44     2.0
Name: R&D Spend, dtype: float64
```

Step 3 - Predict missing value of column 1

df1

	R&D Spend	Administration	Marketing Spend
21	8.00	15.00	30.00
37	23.14	5.00	20.00
2	15.00	10.00	41.00
14	12.00	11.25	26.00
44	2.00	15.00	29.25

Step 4 - Remove the column 2 imputed value (Left to Right)

	R&D Spend	Administration	Marketing Spend
21	8.00	15.0	30.00
37	23.14	5.0	20.00
2	15.00	10.0	41.00
14	12.00	NaN	26.00
44	2.00	15.0	29.25

Training Data in X (Training Input) | Column 2

```
X = df1.iloc[[0,1,2,4],[0,2]]
X
```

	R&D Spend	Marketing Spend
21	8.00	30.00
37	23.14	20.00
2	15.00	41.00
44	2.00	29.25

Training Data in Y (Corresponding Output) | Column 2

```
y = df1.iloc[[0,1,2,4],1]
y
```

```
21    15.0
37     5.0
2     10.0
44    15.0
Name: Administration, dtype: float64
```

Step 5 - Predict missing value of column 2

	R&D Spend	Administration	Marketing Spend
21	8.00	15.00	30.00
37	23.14	5.00	20.00
2	15.00	10.00	41.00
14	12.00	11.06	26.00
44	2.00	15.00	29.25

Step 6 - Remove the column 3 imputed value (Left to Right)

	R&D Spend	Administration	Marketing Spend
21	8.00	15.00	30.0
37	23.14	5.00	20.0
2	15.00	10.00	41.0
14	12.00	11.06	26.0
44	2.00	15.00	NaN

Training Data in X (Training Input) | Column 3

```
X = df1.iloc[0:4,0:2]
X
```

	R&D Spend	Administration
21	8.00	15.00
37	23.14	5.00
2	15.00	10.00
14	12.00	11.06

Training Data in Y (Corresponding Output) | Column 3

```
y = df1.iloc[0:4,-1]
y
```

```
21    30.0
37    20.0
2     41.0
14    26.0
Name: Marketing Spend, dtype: float64
```

Step 7 - Predict missing value of column 3

	R&D Spend	Administration	Marketing Spend
21	8.00	15.00	30.00
37	23.14	5.00	20.00
2	15.00	10.00	41.00
14	12.00	11.06	26.00
44	2.00	15.00	31.56

Step 8 - Subtract 0th (df0) iteration from 1st (df1) iteration

```
df1 - df0
```

	R&D Spend	Administration	Marketing Spend
21	0.00	0.00	0.00
37	13.89	0.00	0.00
2	0.00	0.00	0.00
14	0.00	-0.19	0.00
44	0.00	0.00	2.31

Again Iteration Process

What is the iterative process?

The iterative process is the practice of building, refining, and improving a project, product, or initiative. Teams that use the iterative development process create, test, and revise until they're satisfied with the end result.

Day 16: Curious Data Minds

<Suggest Next Class Topic>

Welcome | Day 16 ML | TheiScale |By Nishant Sir

Import Library

In [8]:
```
import pandas as pd
import numpy as np

from sklearn.linear_model import LinearRegression
```

Import Dataset

In [9]:
```
df = np.round(pd.read_csv('50_Startups.csv')[['R&D Spend','Administration','Marketing
np.random.seed(9)
df = df.sample(5)
df
```

Out[9]:

	R&D Spend	Administration	Marketing Spend	Profit
21	8.0	15.0	30.0	11.0
37	4.0	5.0	20.0	9.0
2	15.0	10.0	41.0	19.0
14	12.0	16.0	26.0	13.0
44	2.0	15.0	3.0	7.0

Remove Target Column

In [10]:
```
df = df.iloc[:,0:-1]
df
```

Out[10]:

	R&D Spend	Administration	Marketing Spend
21	8.0	15.0	30.0
37	4.0	5.0	20.0
2	15.0	10.0	41.0
14	12.0	16.0	26.0
44	2.0	15.0	3.0

NaN value import (Manipulate the Data)

In [13]:
```
df.iloc[1,0] = np.NaN
df.iloc[3,1] = np.NaN
df.iloc[-1,-1] = np.NaN
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_12584\2040708130.py:1: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.iloc[1,0] = np.NaN
C:\Users\ASUS\AppData\Local\Temp\ipykernel_12584\2040708130.py:2: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.iloc[3,1] = np.NaN
C:\Users\ASUS\AppData\Local\Temp\ipykernel_12584\2040708130.py:3: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.iloc[-1,-1] = np.NaN
```

In [12]:
```
df.head()
```

Out[12]:

	R&D Spend	Administration	Marketing Spend
21	8.0	15.0	30.0
37	NaN	5.0	20.0
2	15.0	10.0	41.0
14	12.0	NaN	26.0
44	2.0	15.0	NaN

Step 1 - Impute all missing values with mean

In [14]:
```
df0 = pd.DataFrame()

df0['R&D Spend'] = df['R&D Spend'].fillna(df['R&D Spend'].mean())
df0['Administration'] = df['Administration'].fillna(df['Administration'].mean())
df0['Marketing Spend'] = df['Marketing Spend'].fillna(df['Marketing Spend'].mean())
```

In [15]:
```
df0
```

Out[15]:

	R&D Spend	Administration	Marketing Spend
21	8.00	15.00	30.00
37	9.25	5.00	20.00
2	15.00	10.00	41.00
14	12.00	11.25	26.00
44	2.00	15.00	29.25

Step 2 - Remove the column 1 imputed value (Left to Right)

In [16]:
```
df1 = df0.copy()

df1.iloc[1,0] = np.NaN

df1
```

Out[16]:

	R&D Spend	Administration	Marketing Spend
21	8.0	15.00	30.00
37	NaN	5.00	20.00
2	15.0	10.00	41.00
14	12.0	11.25	26.00
44	2.0	15.00	29.25

Training Data in X (Training Input)

In [19]:
```
X = df1.iloc[[0,2,3,4],1:3]
X
```

Out[19]:

	Administration	Marketing Spend
21	15.00	30.00
2	10.00	41.00
14	11.25	26.00
44	15.00	29.25

Training Data in Y (Corresponding Output)

In [18]:
```
y = df1.iloc[[0,2,3,4],0]
y
```

Out[18]:
```
21     8.0
2     15.0
14    12.0
44     2.0
Name: R&D Spend, dtype: float64
```

Step 3 - Predict missing value of column 1

In [23]:
```
lr = LinearRegression()
lr.fit(X,y)
lr.predict(df1.iloc[1,1:].values.reshape(1,2))
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[23]:
```
array([23.14158651])
```

In [24]:
```
df1.iloc[1,0] = 23.14
```

In [25]:
```
df1
```

Out[25]:

	R&D Spend	Administration	Marketing Spend
21	8.00	15.00	30.00
37	23.14	5.00	20.00
2	15.00	10.00	41.00
14	12.00	11.25	26.00
44	2.00	15.00	29.25

Step 4 - Remove the column 2 imputed value (Left to Right)

In [26]:
```
df1.iloc[3,1] = np.NaN

df1
```

Out[26]:

	R&D Spend	Administration	Marketing Spend
21	8.00	15.0	30.00
37	23.14	5.0	20.00
2	15.00	10.0	41.00
14	12.00	NaN	26.00
44	2.00	15.0	29.25

Training Data in X (Training Input) | Column 2

In [27]:
```
X = df1.iloc[[0,1,2,4],[0,2]]
X
```

Out[27]:

	R&D Spend	Marketing Spend
21	8.00	30.00
37	23.14	20.00
2	15.00	41.00
44	2.00	29.25

Training Data in Y (Corresponding Output) | Column 2

In [29]:
```
y = df1.iloc[[0,1,2,4],1]
y
```

Out[29]:
```
21    15.0
37     5.0
2     10.0
44    15.0
Name: Administration, dtype: float64
```

Step 5 - Predict missing value of column 2

In [30]:
```
lr = LinearRegression()
lr.fit(X,y)
lr.predict(df1.iloc[3,[0,2]].values.reshape(1,2))
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[30]:
```
array([11.06331285])
```

In [31]:
```
df1.iloc[3,1] = 11.06
```

In [32]:
```
df1
```

Out[32]:

	R&D Spend	Administration	Marketing Spend
21	8.00	15.00	30.00
37	23.14	5.00	20.00
2	15.00	10.00	41.00
14	12.00	11.06	26.00
44	2.00	15.00	29.25

Step 6 - Remove the column 3 imputed value (Left to Right)

In [33]:
```
df1.iloc[4,-1] = np.NaN

df1
```

Out[33]:

	R&D Spend	Administration	Marketing Spend
21	8.00	15.00	30.0
37	23.14	5.00	20.0
2	15.00	10.00	41.0
14	12.00	11.06	26.0
44	2.00	15.00	NaN

Training Data in X (Training Input) | Column 3

In [34]:
```
X = df1.iloc[0:4,0:2]
X
```

Out[34]:

	R&D Spend	Administration
21	8.00	15.00
37	23.14	5.00
2	15.00	10.00
14	12.00	11.06

Training Data in Y (Corresponding Output) | Column 3

In [35]:
```
y = df1.iloc[0:4,-1]
y
```

Out[35]:
```
21    30.0
37    20.0
2     41.0
14    26.0
Name: Marketing Spend, dtype: float64
```

Step 7 - Predict missing value of column 3

In [37]:
```
lr = LinearRegression()
lr.fit(X,y)
lr.predict(df1.iloc[4,0:2].values.reshape(1,2))
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[37]:
```
array([31.56351448])
```

In [38]:
```
df1.iloc[4,-1] = 31.56
```

In [39]:
```
df1
```

Out[39]:

	R&D Spend	Administration	Marketing Spend
21	8.00	15.00	30.00
37	23.14	5.00	20.00
2	15.00	10.00	41.00
14	12.00	11.06	26.00
44	2.00	15.00	31.56

Step 8 - Subtract 0th (df0) iteration from 1st (df1) iteration

In [40]:
```
df1 - df0
```

Out[40]:

	R&D Spend	Administration	Marketing Spend
21	0.00	0.00	0.00
37	13.89	0.00	0.00
2	0.00	0.00	0.00
14	0.00	-0.19	0.00
44	0.00	0.00	2.31

Again Iteration Process

In [42]:
```
df2 = df1.copy()

df2.iloc[1,0] = np.NaN

df2
```

Out[42]:

	R&D Spend	Administration	Marketing Spend
21	8.0	15.00	30.00
37	NaN	5.00	20.00
2	15.0	10.00	41.00
14	12.0	11.06	26.00
44	2.0	15.00	31.56

In [43]:
```
X = df2.iloc[[0,2,3,4],1:3]
y = df2.iloc[[0,2,3,4],0]

lr = LinearRegression()
lr.fit(X,y)
lr.predict(df2.iloc[1,1:].values.reshape(1,2))
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[43]:
```
array([23.78627207])
```

In [44]:
```
df2.iloc[1,0] = 23.78
```

In [45]:
```
df2.iloc[3,1] = np.NaN
X = df2.iloc[[0,1,2,4],[0,2]]
y = df2.iloc[[0,1,2,4],1]

lr = LinearRegression()
lr.fit(X,y)
lr.predict(df2.iloc[3,[0,2]].values.reshape(1,2))
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[45]:
```
array([11.22020174])
```

In [46]:
```
df2.iloc[3,1] = 11.22
```

In [47]:
```
df2.iloc[4,-1] = np.NaN

X = df2.iloc[0:4,0:2]
y = df2.iloc[0:4,-1]

lr = LinearRegression()
lr.fit(X,y)
lr.predict(df2.iloc[4,0:2].values.reshape(1,2))
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[47]:
```
array([38.87979054])
```

In [48]:
```
df2.iloc[4,-1] = np.NaN

X = df2.iloc[0:4,0:2]
y = df2.iloc[0:4,-1]

lr = LinearRegression()
lr.fit(X,y)
lr.predict(df2.iloc[4,0:2].values.reshape(1,2))
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[48]:
```
array([38.87979054])
```

In [49]:
```
df2.iloc[4,-1] = 31.56
```

In [50]:
```
df2
```

Out[50]:

	R&D Spend	Administration	Marketing Spend
21	8.00	15.00	30.00
37	23.78	5.00	20.00
2	15.00	10.00	41.00
14	12.00	11.22	26.00
44	2.00	15.00	31.56

In [51]:
```
df2 - df1
```

Out[51]:

	R&D Spend	Administration	Marketing Spend
21	0.00	0.00	0.0
37	0.64	0.00	0.0
2	0.00	0.00	0.0
14	0.00	0.16	0.0
44	0.00	0.00	0.0

In [52]:
```
df3 = df2.copy()

df3.iloc[1,0] = np.NaN

df3
```

Out[52]:

	R&D Spend	Administration	Marketing Spend
21	8.0	15.00	30.00
37	NaN	5.00	20.00
2	15.0	10.00	41.00
14	12.0	11.22	26.00
44	2.0	15.00	31.56

In [53]:
```
X = df3.iloc[[0,2,3,4],1:3]
y = df3.iloc[[0,2,3,4],0]

lr = LinearRegression()
lr.fit(X,y)
lr.predict(df3.iloc[1,1:].values.reshape(1,2))
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[53]:
```
array([24.57698058])
```

In [54]:
```
df3.iloc[1,0] = 24.57
```

In [55]:
```
df3.iloc[3,1] = np.NaN
X = df3.iloc[[0,1,2,4],[0,2]]
y = df3.iloc[[0,1,2,4],1]

lr = LinearRegression()
lr.fit(X,y)
lr.predict(df3.iloc[3,[0,2]].values.reshape(1,2))
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[55]:
```
array([11.37282844])
```

In [56]:
```
df3.iloc[3,1] = 11.37
```

In [57]:
```
df3.iloc[4,-1] = np.NaN

X = df3.iloc[0:4,0:2]
y = df3.iloc[0:4,-1]

lr = LinearRegression()
lr.fit(X,y)
lr.predict(df3.iloc[4,0:2].values.reshape(1,2))
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[57]:
```
array([45.53976417])
```

In [58]:
```
df3.iloc[4,-1] = 45.53
```

In [59]:
```
df2.iloc[3,1] = 11.22
```

In [60]:
```
df3
```

Out[60]:

	R&D Spend	Administration	Marketing Spend
21	8.00	15.00	30.00
37	24.57	5.00	20.00
2	15.00	10.00	41.00
14	12.00	11.37	26.00
44	2.00	15.00	45.53

In [61]:
```
df3 - df2
```

Out[61]:

	R&D Spend	Administration	Marketing Spend
21	0.00	0.00	0.00
37	0.79	0.00	0.00
2	0.00	0.00	0.00
14	0.00	0.15	0.00
44	0.00	0.00	13.97

In [ ]:
```
```

FORMERLY Industries Helping Hands

THE iSCALE

Data Science | 30 Days of Machine Learning | Day - 17

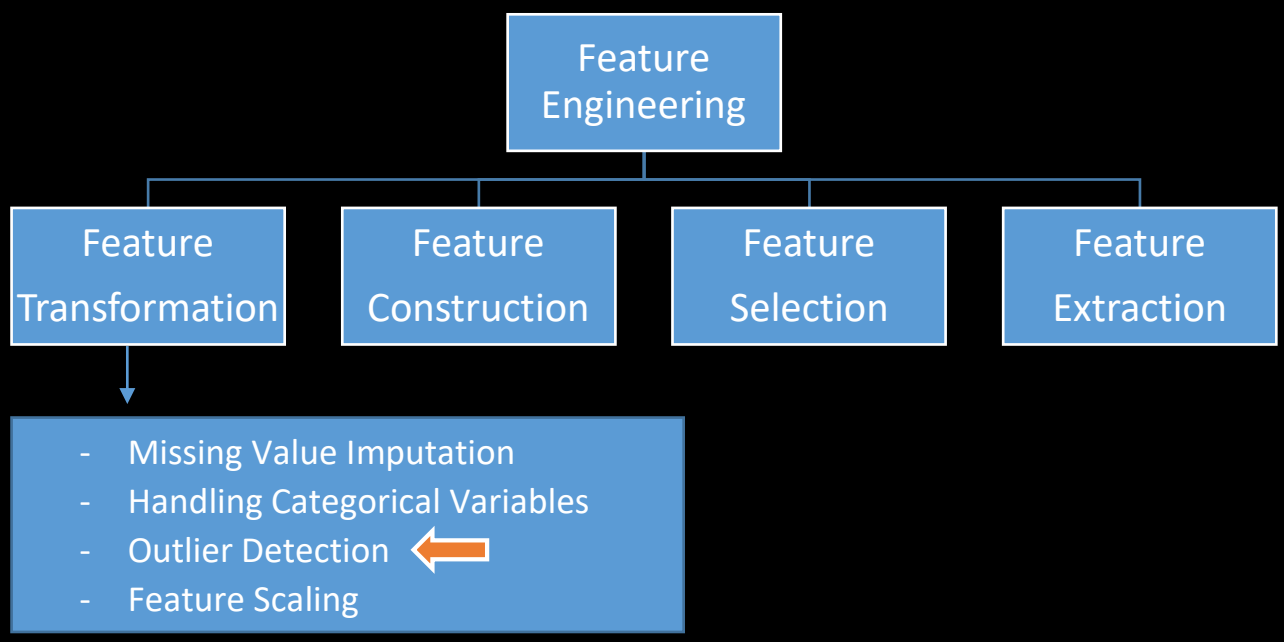
Educator Name: Nishant Dhote
Support Team: **+91-7880-113-112**

----Today Topics | Day 17----

Feature Engineering (Outliers)

- **What are the outliers in machine learning?**
- **When is Outlier Unsafe?**
- **What role play anomaly detection algorithms in outliers?**
- **Effect of Outliers on ML Algorithms**
- **How to treat Outliers?**
- **How to detect Outliers?**
- **Techniques to detect & remove Outliers**

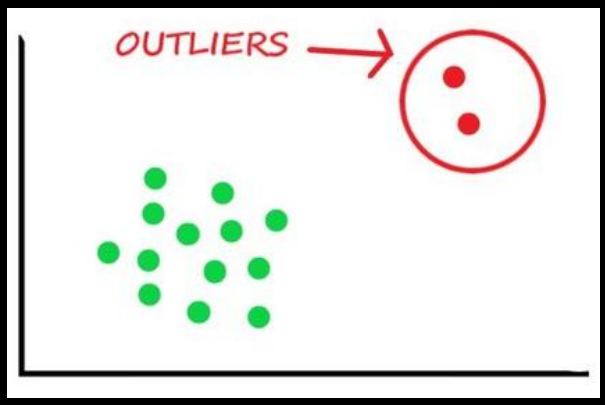
- Dataset Link GitHub: https://github.com/TheiScale/30_Days_Machine_Learning/



What are the outliers in machine learning?

In data analytics, outliers are values within a dataset that vary greatly from the others—they're either much larger, or significantly smaller.

In statistics, any observations or data points that deviate significantly and do not conform with the rest of the observation or data points in a dataset are called outliers. Outliers are extreme values in a feature or dataset.



For example, if you have a dataset with a feature height. The majority of the values in this feature range between 4.5−6.5 feet, but there is one value with 10 feet. This value would be considered an outlier, as it is not only an extreme value but an impossible height as well.

When is Outlier Unsafe?

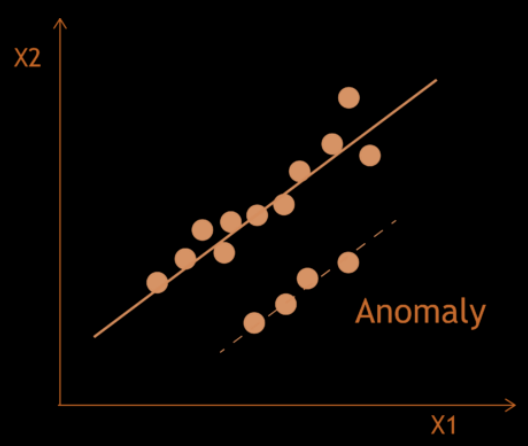
Outliers can be problematic/dangerous when they are caused by incorrect data entries. They can distort statistical measures and lead to misleading conclusions.

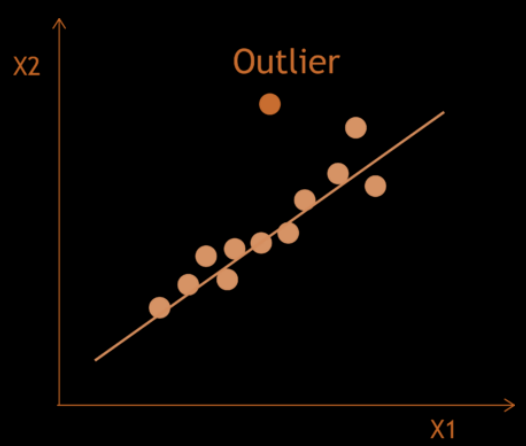
For example, let’s say you're analysing a dataset of students' test scores in a class. Most of the scores fall within a reasonable range between 60 and 100, but there's one data entry showing a score of 1000. This outlier is clearly a mistake, as test scores typically don't exceed 100.

What role play anomaly detection algorithms in outliers?

Anomaly detection is a technique used to identify patterns in data that deviate significantly from the norm or expected behaviour. It involves detecting data points, events, or observations that are rare, unusual, or suspicious compared to the majority of the dataset.

Example: In credit card fraud detection, anomaly detection algorithms are used to identify transactions that deviate from a cardholder's typical spending behaviour. For instance, if a credit card is suddenly used for a large transaction in a foreign country where the cardholder has never travelled before, the transaction may be flagged as an anomaly and subjected to further investigation to determine if it's a fraudulent activity.





Effect of Outliers on ML Algorithms: -

Outliers disrupt weight-based algorithms like linear regression, logistic regression, and AdaBoost. They pull the model towards them, leading to inaccurate predictions (linear regression), misclassification (logistic regression), and overfitting (AdaBoost).

These algorithms aim to minimize the error or maximize the performance by assigning weights to each data point during the learning process.

How to treat Outliers?

Trimming or removing outliers completely :-One way to treat outliers is by removing them completely from the dataset. However, this approach should be used carefully because it may reduce the size of the dataset significantly if there are many outliers.

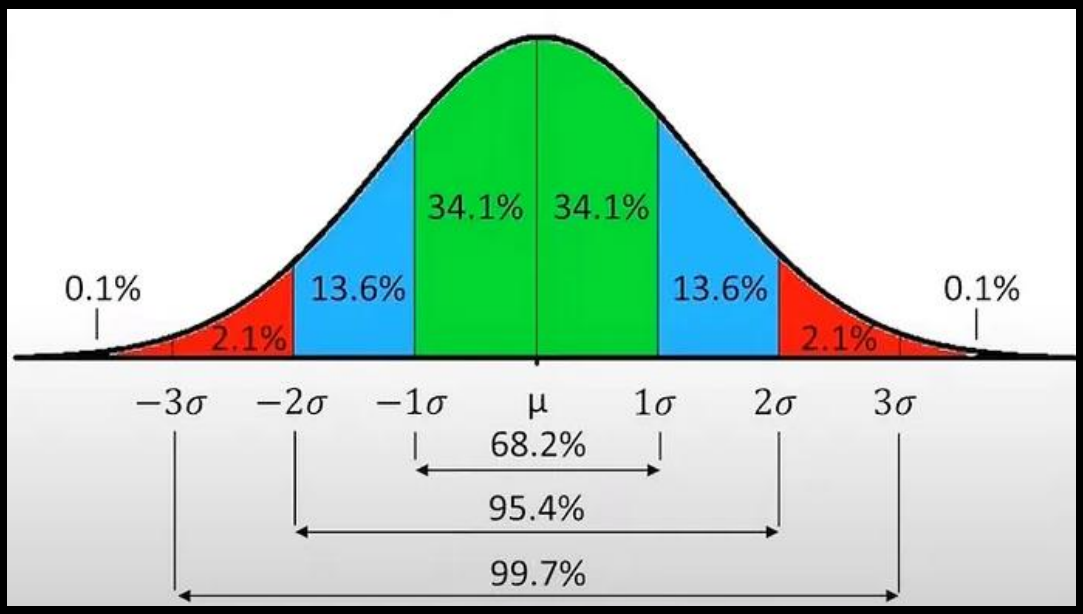
Capping: - This technique involves setting a limit or threshold on both ends of the data distribution and replacing outliers beyond those limits with the nearest non-outlier values. This approach helps in reducing the impact of extreme values without completely discarding them.

Treating outliers as missing values: - Another approach is to treat outliers as missing values and then handle them using appropriate missing data imputation techniques. This approach allows for a more flexible treatment of outliers based on the specific characteristics of the dataset.

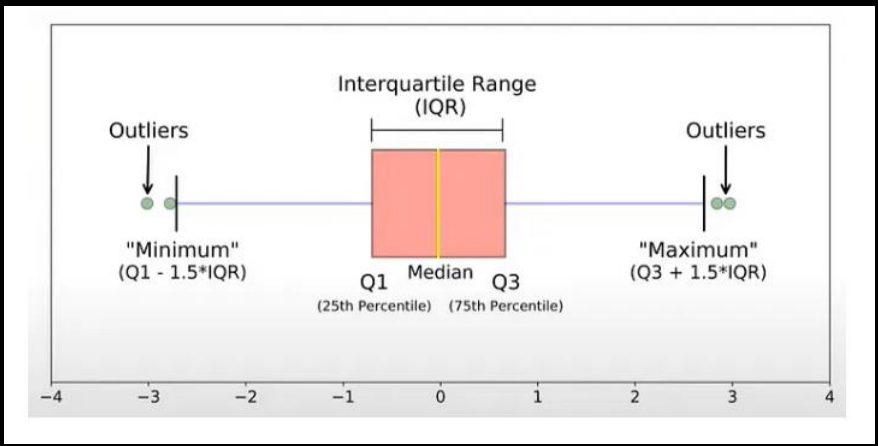
How to detect Outliers?

To detect outliers in a dataset, there are various approaches: -

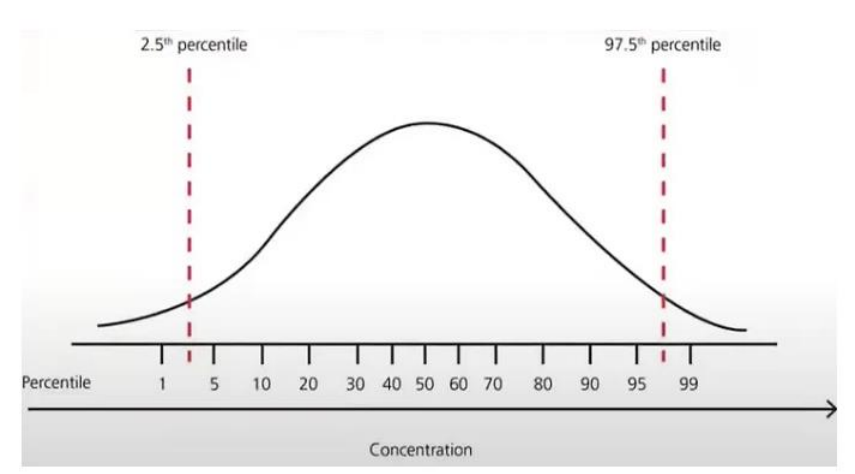
Normal distribution: - If a column follows a normal distribution, you can identify outliers by checking if a particular observation falls outside the range of mean plus or minus three standard deviations.

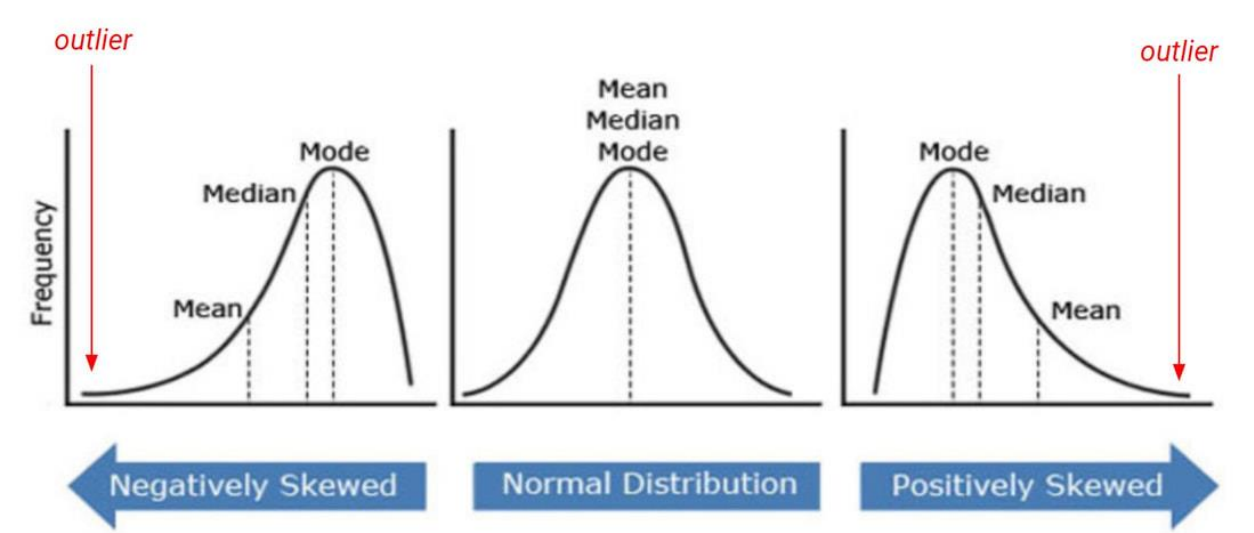


Skewed distribution: - For skewed distributions, you can use the interquartile range (IQR) proximity rule. Calculate the minimum value as (Q1–1.5 * IQR) and the maximum value as (Q3 + 1.5 * IQR). Any value that is lower than the minimum or higher than the maximum is considered an outlier.



Other distributions (percentile-based approach) :- In this approach, you can identify outliers by comparing the observations to specific percentiles. If an observation is greater than the 97.5th percentile or less than the 2.5th percentile, it is considered an outlier.





In negatively skewed data, the **mean is a smaller number** than the median and mode.	In a normal distribution, the **mean is the same number** as the median and mode.	In positively skewed data, the **mean is a larger number** than the median and mode.

Techniques to detect & remove Outliers: -

Z-score treatment: - This technique assumes that the column follows a normal distribution.

IQR (Interquartile Range) based filtering: - The IQR method involves calculating the range between the first quartile (Q1) and the third quartile (Q3).

Percentile method: - In this approach, a threshold is set based on percentiles. For example, if the threshold is set at 5%, any data point above the 95th percentile or below the 5th percentile is considered an outlier. These outliers can be removed or handled accordingly.

Winsorization:- Winsorization involves replacing outliers with values at a certain percentile, rather than removing them completely.

Day 17: Curious Data Minds

Art of Storytelling in Data Science:

- **Why Story telling is important for "Data Science" Interview?**

Storytelling is important for data science interviews because:

1. **Communication**: It helps explain complex data findings to non-technical people.
2. **Context**: It puts data analysis into a meaningful story, explaining why it's important.
3. **Engagement**: Stories capture attention and persuade people better than just data.
4. **Simplicity**: It simplifies complex data concepts, making them easier to understand.
5. **Memorability**: Stories are more memorable, ensuring that insights stick with people.

In interviews, storytelling shows you can explain data clearly, understand its context, and influence decisions—valuable skills in data science.

https://eightify.app/summary/miscellaneous/why-jeff-bezos-banned-powerpoint-at-amazon-lex-fridman-podcast-clips

- Focus point to remember in Interview

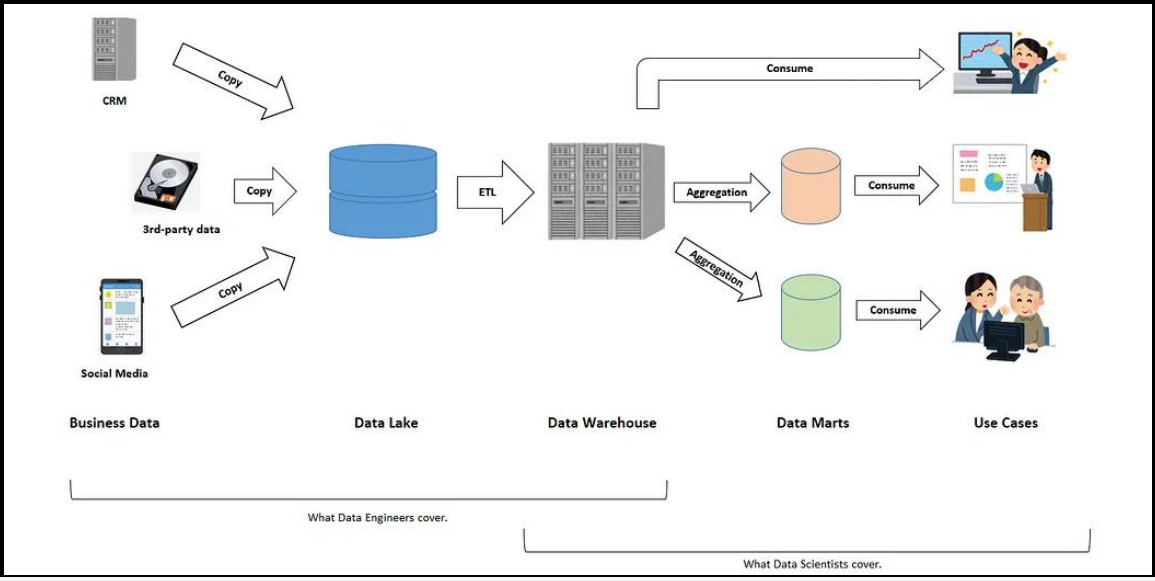
- **Tell interviewers what you know?**
- **Focus on Data Storytelling (About Projects)**

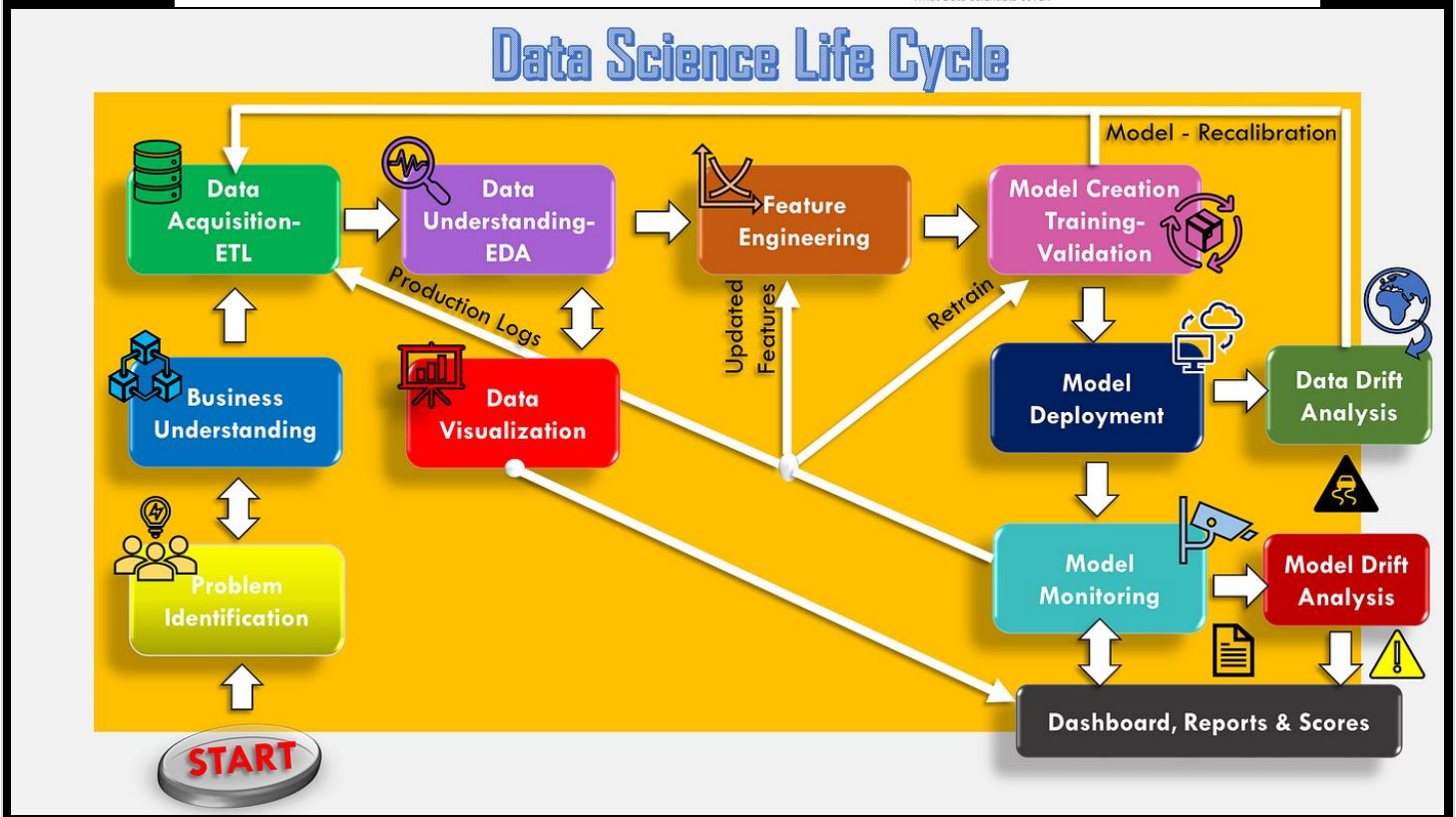
- How should be start the Data Story telling?

1. **Introduction**: Begin by introducing your project and why it's important.
2. **Hook**: Grab your interviewer's attention with an interesting fact or story related to your data.
3. **Problem Statement**: Clearly state the problem you're trying to solve or the question you're exploring.
4. **Objective**: Explain what you aim to achieve through your analysis.
5. **Preview of Insights**: Give a quick overview of the main findings or insights you'll be sharing.

- "Tell about your projects."
- "Discuss the architecture used."
- "Focus on the Life Cycle of Data Science Projects."
- "Focus on your roles in the team."
- "Tell about the challenges you faced and how you handled them."

https://towardsdatascience.com/fundamentals-of-data-architecture-to-help-data-scientists-understand-architectural-diagrams-better-7bd26de41c66





FORMERLY Industries Helping Hands

THE iSCALE

Data Science | 30 Days of Machine Learning | Day - 18

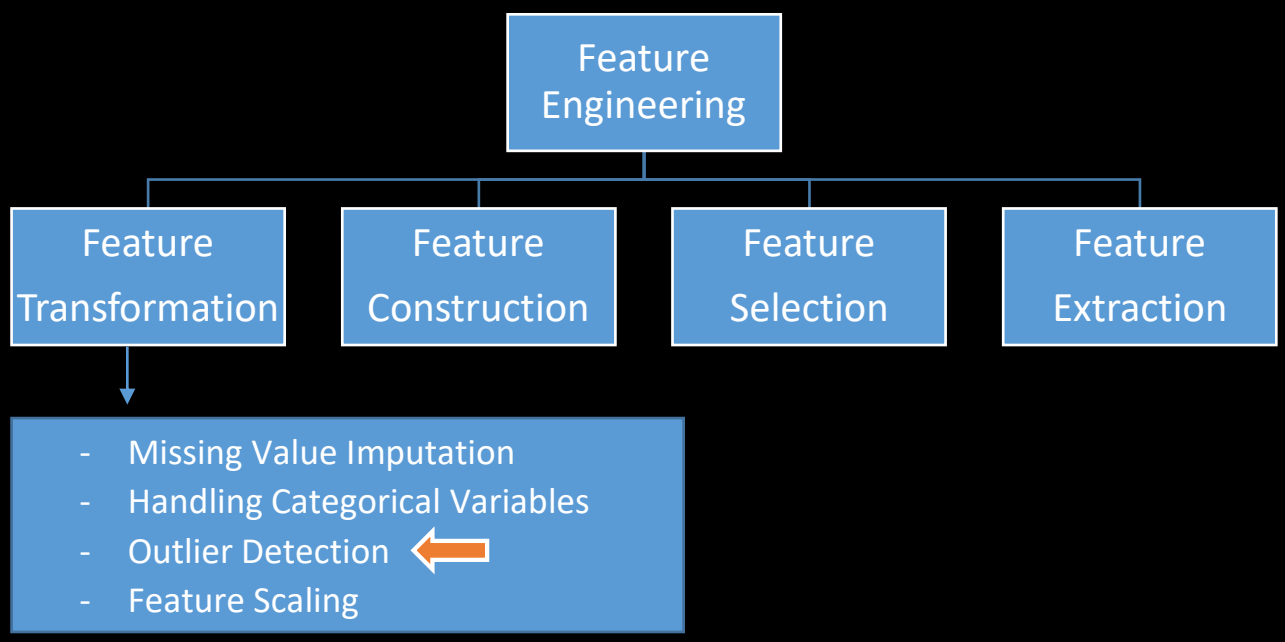
Educator Name: Nishant Dhote
Support Team: **+91-7880-113-112**

----Today Topics | Day 18----

Outliers: Z score technique

- **Outliers removal using Z score treatment**
- **Z Score is applicable for normal distribution**
- **What is Standard deviation?**
- **Standard Normal Distribution (SND)**
- **Why are Z-Scores Important?**
- **How to Calculate "Z-Score"?**
- **What is 68 - 95- 99 Rule?**
- **Practice Problems For Z-Scores Calculation**

- Dataset Link GitHub: https://github.com/TheiScale/30_Days_Machine_Learning/



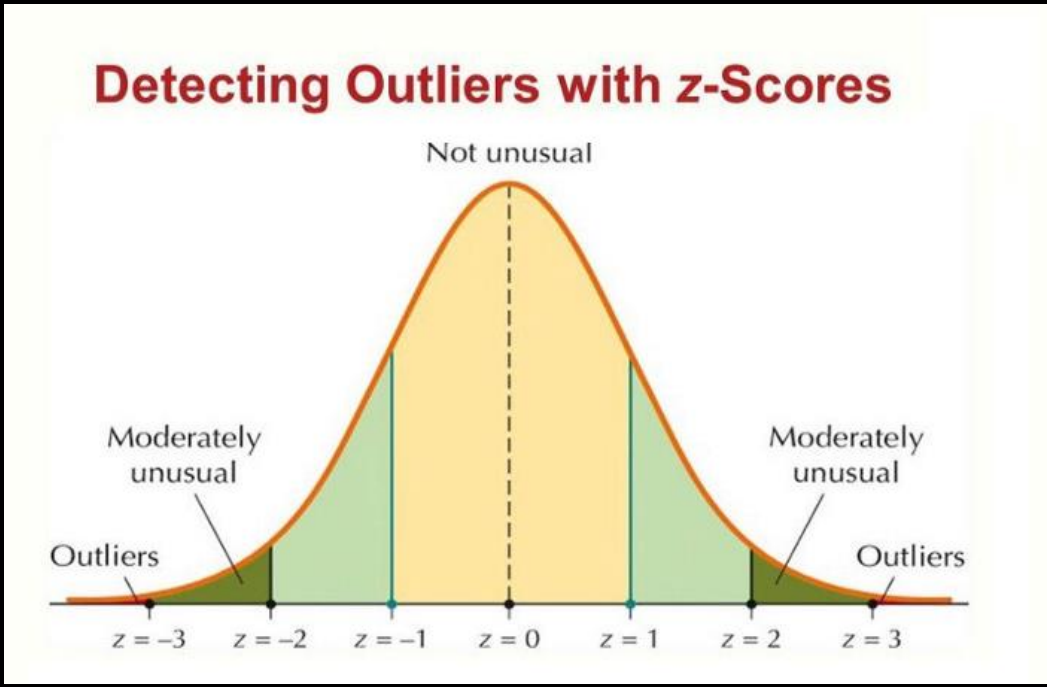
Techniques to detect & remove outliers: -

Z-score treatment: - This technique assumes that the column follows a normal distribution.

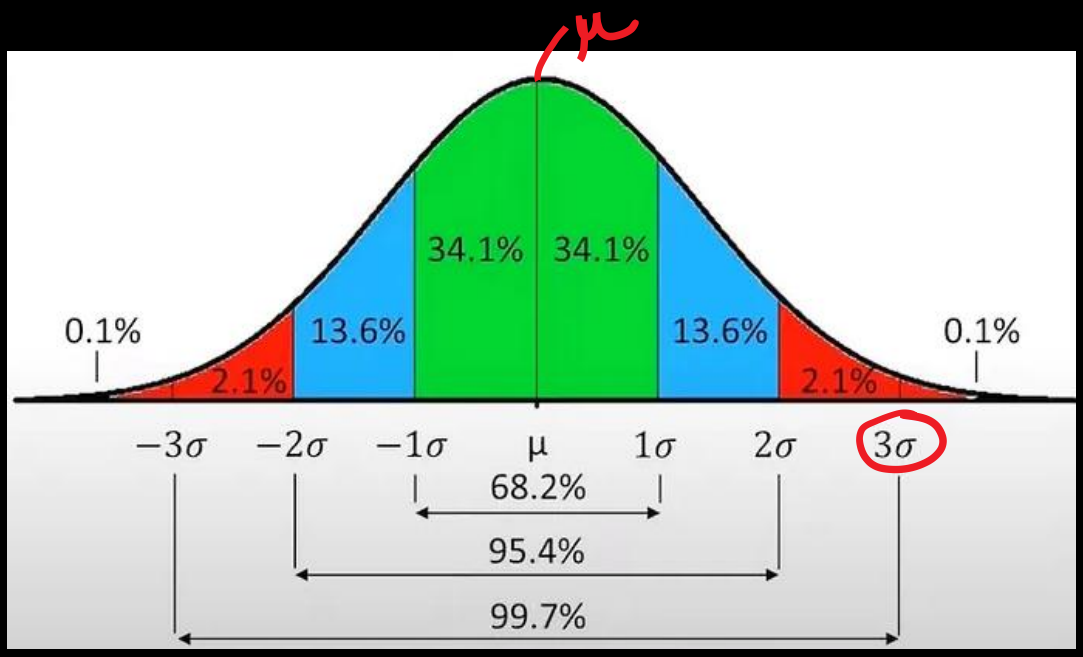
IQR (Interquartile Range) based filtering: - The IQR method involves calculating the range between the first quartile (Q1) and the third quartile (Q3).

Percentile method: - In this approach, a threshold is set based on percentiles. For example, if the threshold is set at 5%, any data point above the 95th percentile or below the 5th percentile is considered an outlier. These outliers can be removed or handled accordingly.

Winsorization:- Winsorization involves replacing outliers with values at a certain percentile, rather than removing them completely.



Outliers removal using Z score treatment:



$\mu + 1\sigma$ $\mu - 1\sigma$

68.2%

$\mu + 2\sigma$ $\mu - 2\sigma$

95.4%

$\mu + 3\sigma$ $\mu - 3\sigma$

99.7%

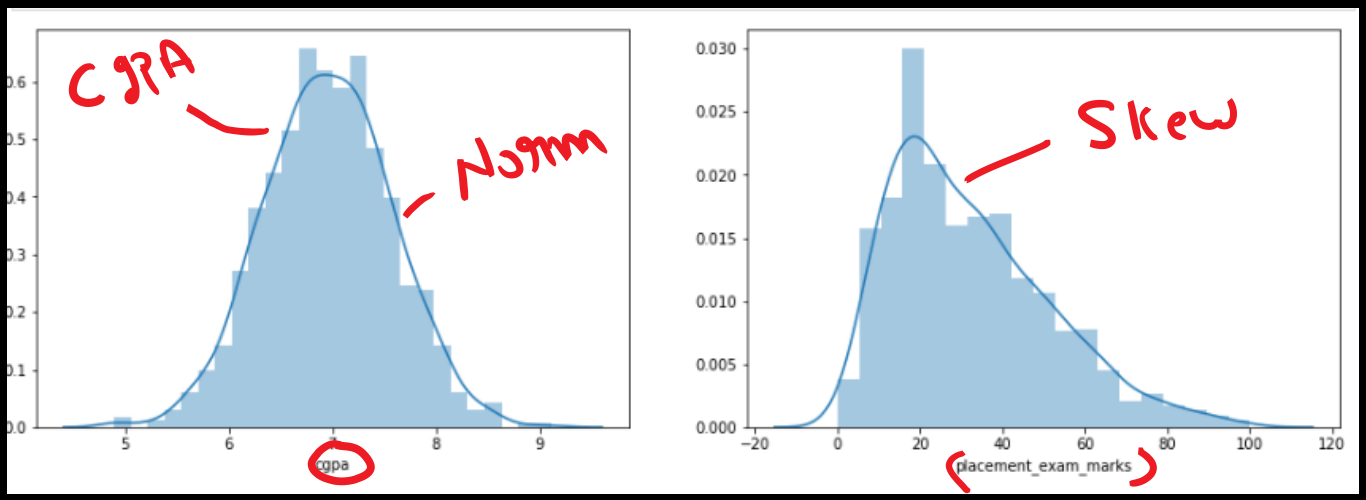
Z Score is applicable for normal distribution:

https://www.scribbr.com/statistics/standard-normal-distribution/#:~:text=While%20data%20points%20are%20referred,is%20greater%20than%20the%20mean.

Z-score only applies to distributions derived from the normal distribution, or distributions that can be approximated by them. Many skewed distributions cannot. Therefore, much of the time, Z-score does not apply skewed distributions.

1000
CgPA
Em

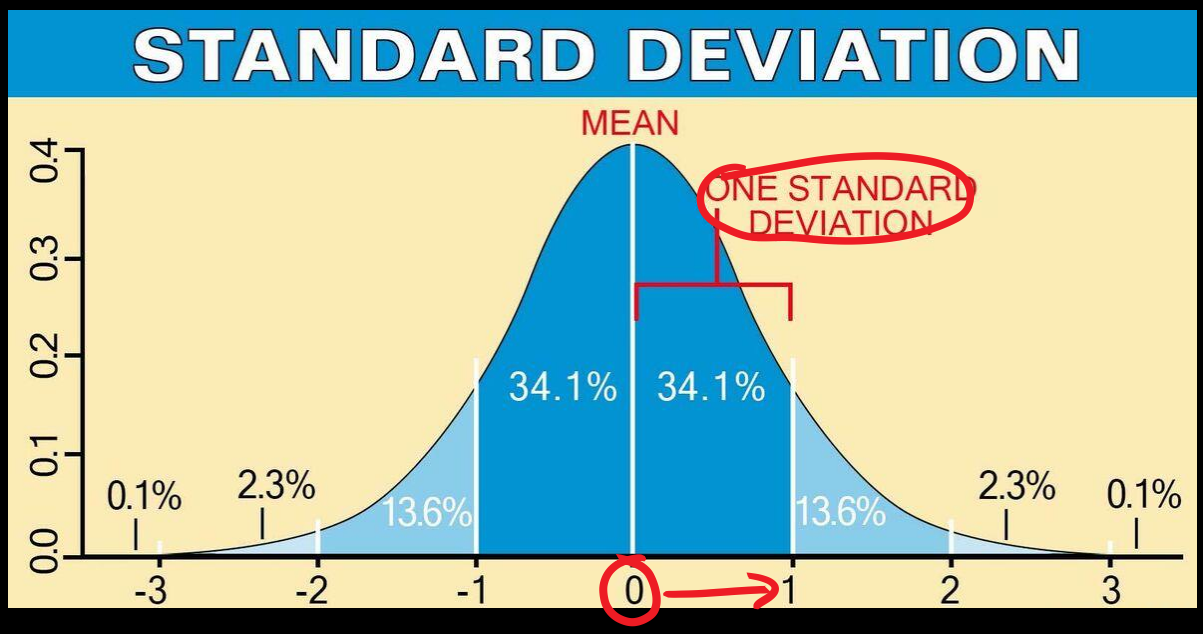
	cgpa	placement_exam_marks	placed
689	8.02	67.0	0
111	6.48	33.0	0
991	7.04	57.0	0
835	6.67	65.0	1
772	6.63	26.0	0



Stat'ist

What is Standard deviation?

Standard Deviation is a measure which shows how much variation (such as spread) from the mean exists. The standard deviation indicates a "typical" deviation from the mean. It is a popular measure of variability because it returns to the original units of measure of the data set.



Standard Normal Distribution (SND)

1. The SND (i.e., z-distribution) is always the same shape as the raw score distribution. For example, if the distribution of raw scores is normally distributed, so is the distribution of z-scores.

2. The mean of any SND always = 0.

3. The standard deviation of any SND always = 1. Therefore, one standard deviation of the raw score (whatever raw value this is) converts into 1 z-score unit.

Blog: https://www.simplypsychology.org/z-score.html

Why are Z-Scores Important?

It is useful to standardize the values (raw scores) of a normal distribution by converting them into z-scores because:

1. It allows researchers to calculate the probability of a score occurring within a standard normal distribution;

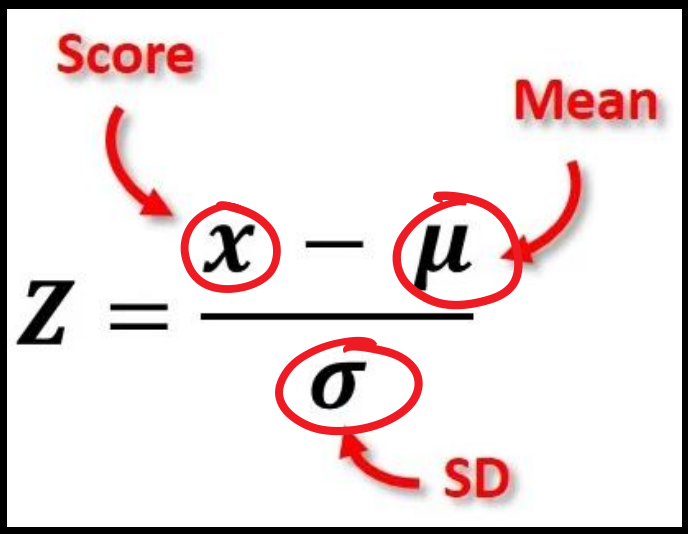
2. It enables us to compare two scores from different samples (which may have different means and standard deviations).

How to Calculate "Z-Score"?

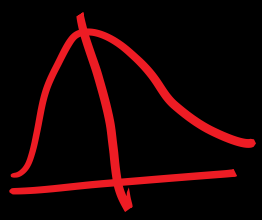
The formula for calculating a z-score is z = (x-μ)/σ, where x is the raw score, μ is the population mean, and σ is the population standard deviation.

$\mu + 3\sigma$

$\mu - 3\sigma$



mean

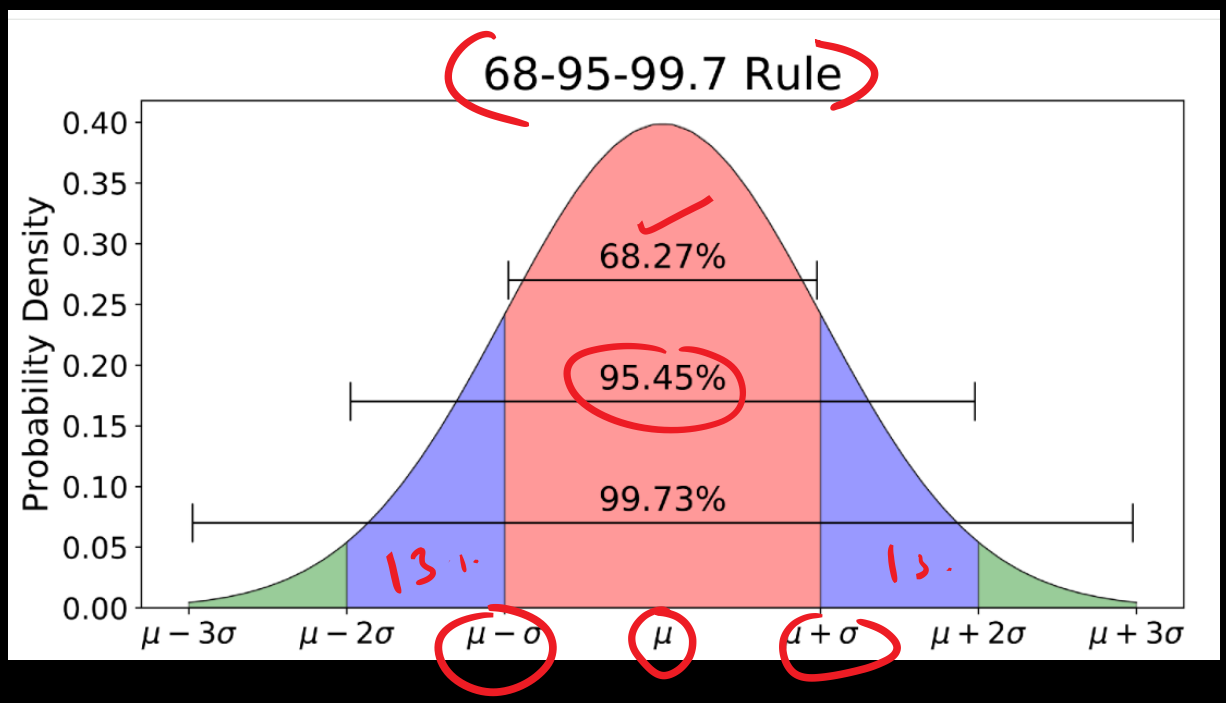


x = Raw Score

μ = mean

σ = Standard deviation

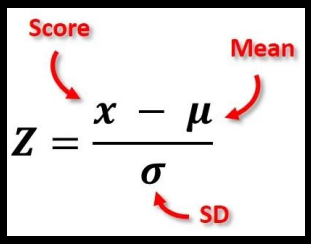
What is 68 - 95- 99 Rule:

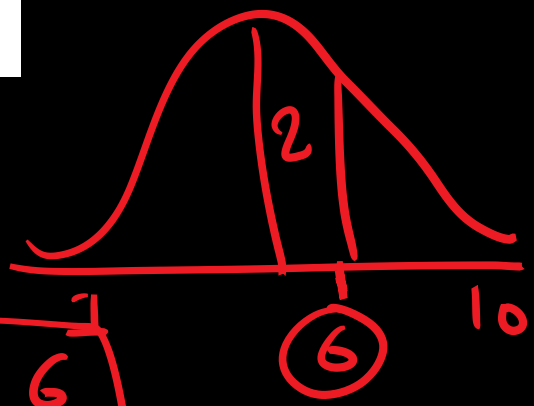


Practice Problems For Z-Scores Calculation:

Problem 1: Scores on a psychological well-being scale range from 1 to 10, with an average score of 6 and a standard deviation of 2. What is the z-score for a person who scored 4?

1 4 6 10



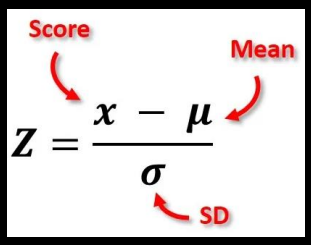


Solution 1:

Z-score = (4 – 6)/2 = -1

$$Z = \frac{x - \mu}{\sigma} = \frac{4 - 6}{2} = -2/2 = -1$$

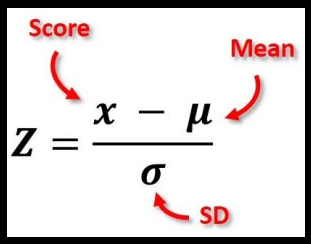
Problem 2: On a measure of anxiety, a group of participants show a mean score of 35 with a standard deviation of 5. What is the z-score corresponding to a score of 30?



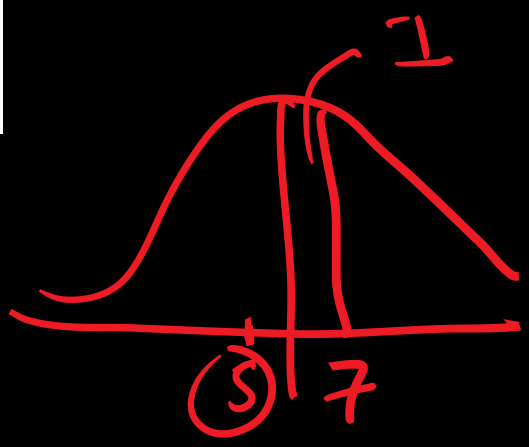
Solution 2:
Z-score = (30 – 35)/5 = -1

$$Z = \frac{x - \mu}{\sigma} = \frac{30 - 35}{5} = \frac{-5}{5} = -1$$

Problem 3: In a study on sleep, participants report an average of 7 hours of sleep per night, with a standard deviation of 1 hour. What is the z-score for a person reporting 5 hours of sleep?



Solution 3:
Z-score = (5 – 7)/1 = -2



$$Z = \frac{x - \mu}{\sigma} = \frac{5 - 7}{1}$$

$$\boxed{Z = -2}$$

Step 1: Finding the boundary values (Highest & Lowest)
Step 2: Finding the outliers
Step 3: Treat outliers with suitable technique.

How to treat outliers

- Trimming
- Capping

<Start Coding>

```
#Import Library

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#Import Dataset

df = pd.read_csv('placement.csv')

----
df

----
df.shape

----
df.sample(5)
```

```
#Plot Show in CGPA and Placement Marks

plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.distplot(df['cgpa'])

plt.subplot(1,2,2)
sns.distplot(df['placement_exam_marks'])

plt.show()

#Print Mean | Std | Min & Max Value

print("Mean value of cgpa",df['cgpa'].mean())
print("Std value of cgpa",df['cgpa'].std())
print("Min value of cgpa",df['cgpa'].min())
print("Max value of cgpa",df['cgpa'].max())


#Approach 1

#Step 1:Finding the boundary values (Highest &
Lowest)

print("Highest allowed",df['cgpa'].mean() +
3*df['cgpa'].std())

print("Lowest allowed",df['cgpa'].mean() -
3*df['cgpa'].std())

#Step 2: Finding the outliers

df[(df['cgpa'] > 8.80) | (df['cgpa'] < 5.11)]
```

```
#Step 3: Treat Outliers with Trimming

new_df = df[(df['cgpa'] < 8.80) & (df['cgpa'] >
5.11)]
new_df

# Approach 2 : Calculating the Zscore

df['cgpa_zscore'] = (df['cgpa'] -
df['cgpa'].mean())/df['cgpa'].std()

----
df.head()

#CGPA Score More then 3

df[df['cgpa_zscore'] > 3]

#CGPA Score Less then 3

df[df['cgpa_zscore'] < -3]

#Show or Merge Both CGPA

df[(df['cgpa_zscore'] > 3) | (df['cgpa_zscore'] < -
3)]


# Apply Trimming

new_df = df[(df['cgpa_zscore'] < 3) &
(df['cgpa_zscore'] > -3)]

----

new_df
```

Day 18: Curious Data Minds

Art of Storytelling in Data Science:

- Why Story telling is important for "Data Science" Interview?

Storytelling is important for data science interviews because:

1. **Communication**: It helps explain complex data findings to non-technical people.
2. **Context**: It puts data analysis into a meaningful story, explaining why it's important.
3. **Engagement**: Stories capture attention and persuade people better than just data.
4. **Simplicity**: It simplifies complex data concepts, making them easier to understand.
5. **Memorability**: Stories are more memorable, ensuring that insights stick with people.

In interviews, storytelling shows you can explain data clearly, understand its context, and influence decisions—valuable skills in data science.

https://eightify.app/summary/miscellaneous/why-jeff-bezos-banned-powerpoint-at-amazon-lex-fridman-podcast-clips

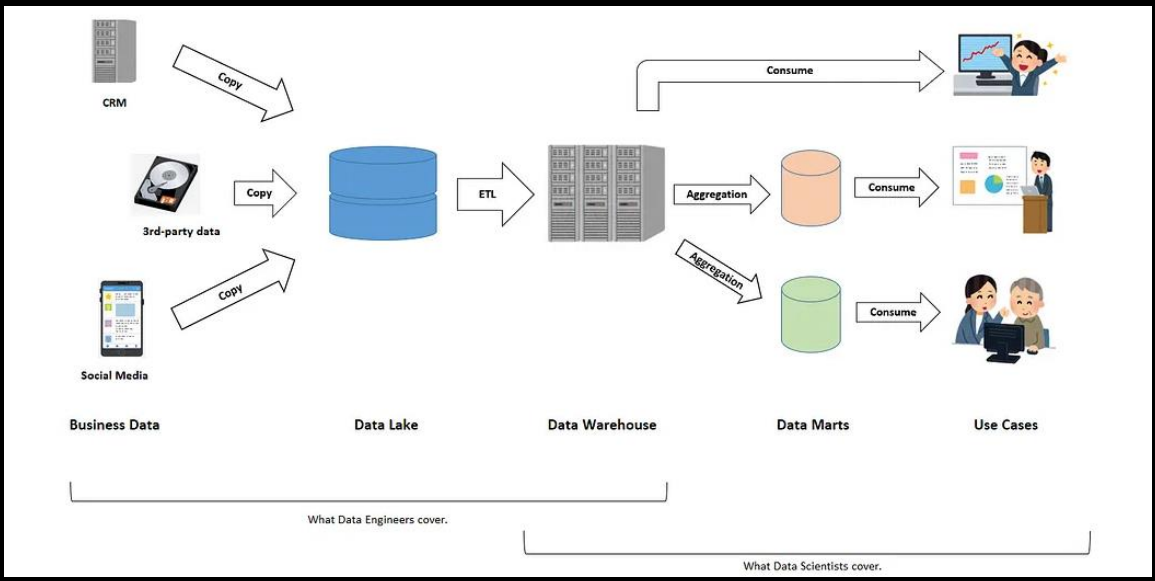
- Focus point to remember in Interview

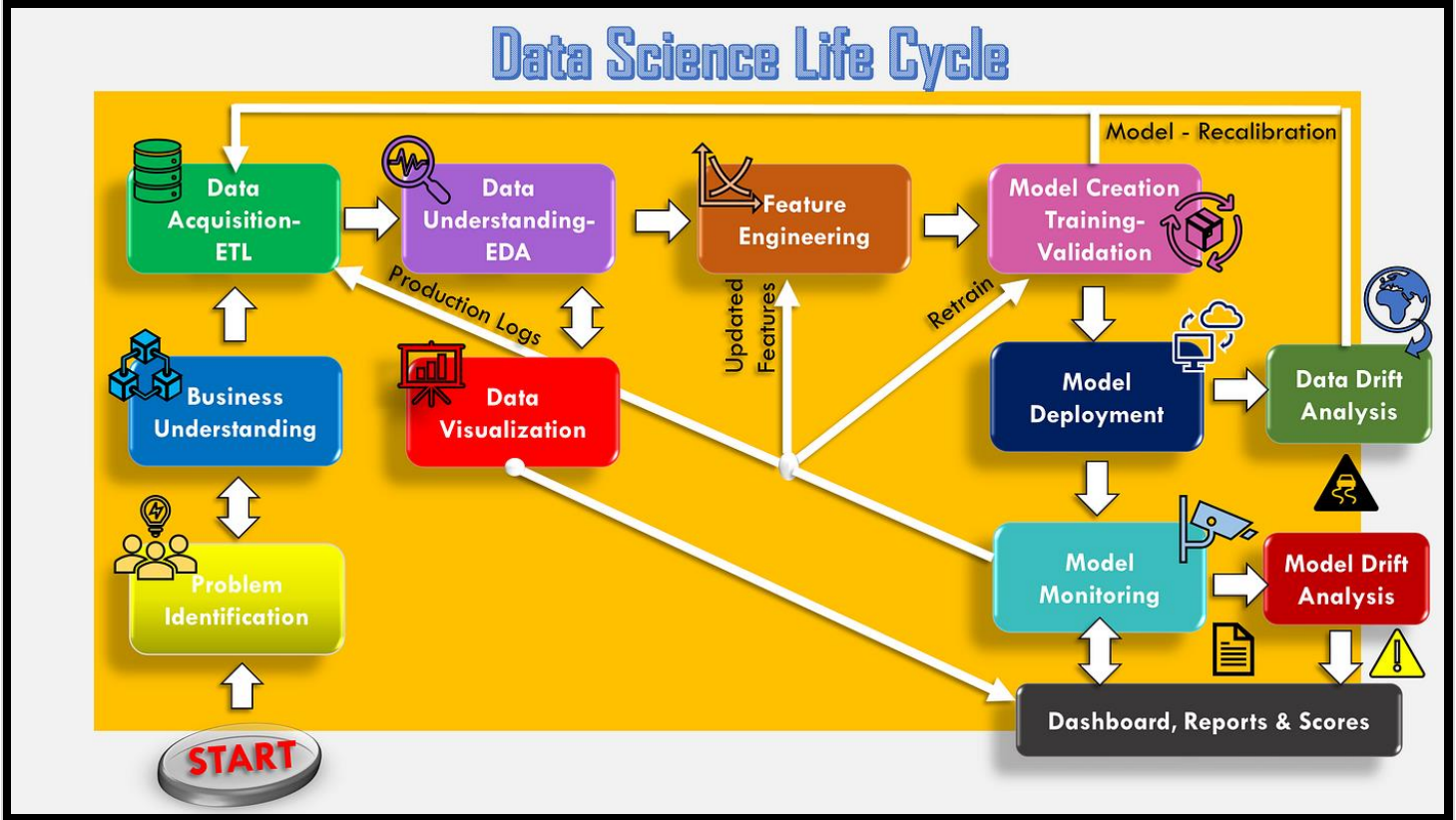
- **Tell interviewers what you know?**
- **Focus on Data Storytelling (About Projects)**

- How should be start the Data Story telling?

1. **Introduction**: Begin by introducing your project and why it's important.
2. **Hook**: Grab your interviewer's attention with an interesting fact or story related to your data.
3. **Problem Statement**: Clearly state the problem you're trying to solve or the question you're exploring.
4. **Objective**: Explain what you aim to achieve through your analysis.
5. **Preview of Insights**: Give a quick overview of the main findings or insights you'll be sharing.

- "Tell about your projects."
- "Discuss the architecture used."
- "Focus on the Life Cycle of Data Science Projects."
- "Focus on your roles in the team."
- "Tell about the challenges you faced and how you handled them." https://towardsdatascience.com/fundamentals-of-data-architecture-to-help-data-scientists-understand-architectural-diagrams-better-7bd26de41c66




Data Science Life Cycle
Model - Recalibration
Data Acquisition-ETL
Data Understanding-EDA
Feature Engineering
Model Creation Training-Validation
Production Logs
Updated Features
Retrain
Business Understanding
Data Visualization
Model Deployment
Data Drift Analysis
Problem Identification
Model Monitoring
Model Drift Analysis
Dashboard, Reports & Scores
START

Welcome to 30 Days ML | Day 18

Import Library

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Import Dataset

In [2]:
```
df = pd.read_csv('placement.csv')
```

In [3]:
```
df
```

Out[3]:

	cgpa	placement_exam_marks	placed
0	7.19	26.0	1
1	7.46	38.0	1
2	7.54	40.0	1
3	6.42	8.0	1
4	7.23	17.0	0
...	...	...	...
995	8.87	44.0	1
996	9.12	65.0	1
997	4.89	34.0	0
998	8.62	46.0	1
999	4.90	10.0	1

1000 rows × 3 columns

In [4]:
```
df.head()
```

Out[4]:

	cgpa	placement_exam_marks	placed
0	7.19	26.0	1
1	7.46	38.0	1
2	7.54	40.0	1
3	6.42	8.0	1
4	7.23	17.0	0

In [5]:
```
df.sample(5)
```

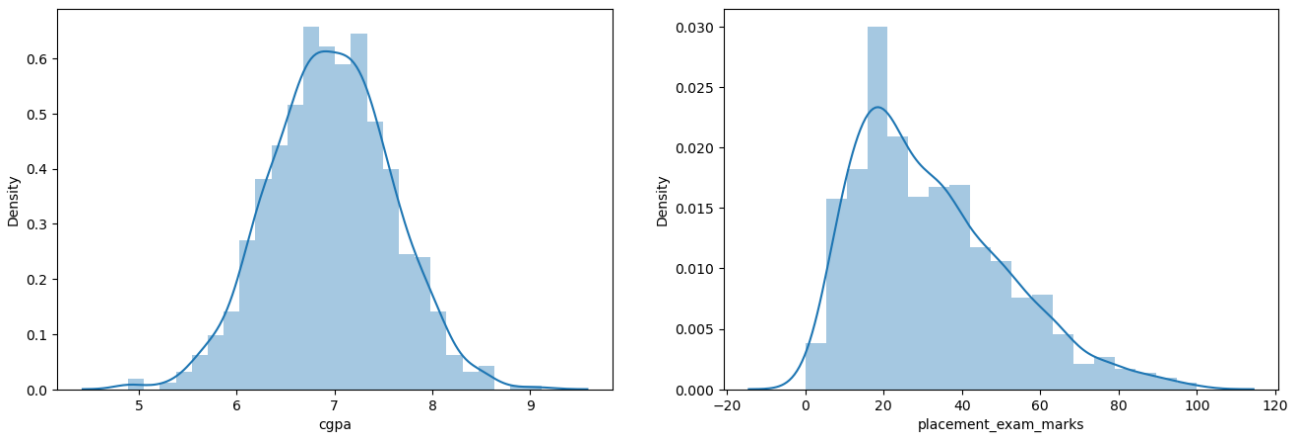
Out[5]:

	cgpa	placement_exam_marks	placed
655	7.36	34.0	0
339	7.32	18.0	1
274	7.13	4.0	1
138	7.53	8.0	1
83	7.38	20.0	1

Plot Show in CGPA and Placement Marks

In [6]:
```
plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.distplot(df['cgpa'])
plt.subplot(1,2,2)
sns.distplot(df['placement_exam_marks'])
plt.show()
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_11704\4260214945.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.githu
b.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df['cgpa'])
C:\Users\ASUS\AppData\Local\Temp\ipykernel_11704\4260214945.py:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.githu
b.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df['placement_exam_marks'])
```



Print Mean | Std | Min & Max Value

In [7]:
```
print("Mean value of cgpa",df['cgpa'].mean())
print("Std value of cgpa",df['cgpa'].std())
print("Min value of cgpa",df['cgpa'].min())
print("Max value of cgpa",df['cgpa'].max())
```

```
Mean value of cgpa 6.96124
Std value of cgpa 0.6158978751323894
Min value of cgpa 4.89
Max value of cgpa 9.12
```

Approach 1

Step 1:Finding the boundary values (Highest & Lowest)

In [8]:
```
print("Highest allowed",df['cgpa'].mean() + 3*df['cgpa'].std())
print("Lowest allowed",df['cgpa'].mean() - 3*df['cgpa'].std())
```

```
Highest allowed 8.808933625397168
Lowest allowed 5.113546374602832
```

Step 2: Finding the outliers

In [9]:
```
df[(df['cgpa'] > 8.80) | (df['cgpa'] < 5.11)]
```

Out[9]:

	cgpa	placement_exam_marks	placed
485	4.92	44.0	1
995	8.87	44.0	1
996	9.12	65.0	1
997	4.89	34.0	0
999	4.90	10.0	1

Step 3: Treat Outliers with Trimming

In [10]:
```
new_df = df[(df['cgpa'] < 8.80) & (df['cgpa'] > 5.11)]
new_df
```

Out[10]:

	cgpa	placement_exam_marks	placed
0	7.19	26.0	1
1	7.46	38.0	1
2	7.54	40.0	1
3	6.42	8.0	1
4	7.23	17.0	0
...	...	...	...
991	7.04	57.0	0
992	6.26	12.0	0
993	6.73	21.0	1
994	6.48	63.0	0
998	8.62	46.0	1

995 rows × 3 columns

Approach 2 : Calculating the Zscore

In [12]:
```
df['cgpa_zscore'] = (df['cgpa'] - df['cgpa'].mean())/df['cgpa'].std()
```

In [13]:
```
df
```

Out[13]:

	cgpa	placement_exam_marks	placed	cgpa_zscore
0	7.19	26.0	1	0.371425
1	7.46	38.0	1	0.809810
2	7.54	40.0	1	0.939701
3	6.42	8.0	1	-0.878782
4	7.23	17.0	0	0.436371
...	...	...	...	...
995	8.87	44.0	1	3.099150
996	9.12	65.0	1	3.505062
997	4.89	34.0	0	-3.362960
998	8.62	46.0	1	2.693239
999	4.90	10.0	1	-3.346724

1000 rows × 4 columns

CGPA Score More then 3

In [14]:
```
df[df['cgpa_zscore'] > 3]
```

Out[14]:

	cgpa	placement_exam_marks	placed	cgpa_zscore
995	8.87	44.0	1	3.099150
996	9.12	65.0	1	3.505062

CGPA Score Less then 3

In [15]:
```
df[df['cgpa_zscore'] < -3]
```

Out[15]:

	cgpa	placement_exam_marks	placed	cgpa_zscore
485	4.92	44.0	1	-3.314251
997	4.89	34.0	0	-3.362960
999	4.90	10.0	1	-3.346724

Show or Merge Both CGPA

In [16]:
```
df[(df['cgpa_zscore'] > 3) | (df['cgpa_zscore'] < -3)]
```

Out[16]:

	cgpa	placement_exam_marks	placed	cgpa_zscore
485	4.92	44.0	1	-3.314251
995	8.87	44.0	1	3.099150
996	9.12	65.0	1	3.505062
997	4.89	34.0	0	-3.362960
999	4.90	10.0	1	-3.346724

Apply Trimming

In [17]:
```
new_df = df[(df['cgpa_zscore'] < 3) & (df['cgpa_zscore'] > -3)]
```

In [18]:
```
new_df
```

Out[18]:

	cgpa	placement_exam_marks	placed	cgpa_zscore
0	7.19	26.0	1	0.371425
1	7.46	38.0	1	0.809810
2	7.54	40.0	1	0.939701
3	6.42	8.0	1	-0.878782
4	7.23	17.0	0	0.436371
...	...	...	...	...
991	7.04	57.0	0	0.127878
992	6.26	12.0	0	-1.138565
993	6.73	21.0	1	-0.375452
994	6.48	63.0	0	-0.781363
998	8.62	46.0	1	2.693239

995 rows × 4 columns

In [ ]: