Unit 1

INTRODUCTION TO MACHINE LEARNING

Basic Terminology

- **Algorithm** – A set of rules or instructions given to a machine to help it learn and make predictions or decisions.
- **Model** – A trained algorithm that makes predictions based on data.
- **Training Data** – The dataset used to train a machine learning model. It contains labeled or unlabeled data for supervised or unsupervised learning.
- **Test Data** – Data that is used to evaluate the performance of a trained machine learning model.
- **Features** – The input variables (independent variables) that are used by the model to make predictions.
- **Labels** – The output variable (dependent variable) that the model is trying to predict or classify.
- **Cross-validation** – A model validation technique used to assess the performance of a machine learning model. Common techniques include k-fold cross-validation. Cross-validation determines the accuracy of your machine learning model by partitioning the data into two different groups, called a training set and a testing set. The data is then randomly separated into a certain number of groups or subsets called folds. Each fold contains about the same amount of data.

- **Outliers:** In machine learning, an outlier is a data point that differs significantly from other data points in a dataset. Outliers can be caused by data entry errors, sensor errors, or rare occurrences
- **Over fitting** : A scenario where a model learns the details and noise in the training data to the extent that it negatively impacts its performance on new data. Over-fitting can be defined as a phenomenon where a machine learning model learns the training data too well, capturing not only the underlying patterns but also the noise and fluctuations present in that particular dataset. This results in a lack of generalization ability when confronted with new, previously unseen data.
- **Underfitting** – A scenario where a model is too simple to capture the underlying patterns of the data, resulting in poor performance. **Under fitting** occurs when a model is too simple and fails to capture the key patterns in the data. This leads to inaccurate predictions for both the training data and new data.
- **Bias** – The error introduced by approximating a real-world problem with a too simplistic model. Bias is simply defined as the inability of the model because of that there is some difference or error occurring between the model's predicted value and the actual value. These differences between actual or expected values and the predicted values are known as error or bias error or error due to bias. Bias is a systematic error that occurs due to wrong assumptions in the machine learning process. Let Y be the true value of a parameter, and let Y^ be an estimator of Y based on a sample of data. Then, the bias of the estimator Y^ is given by:
- Bias(Y^)=E(Y^)–Y
- where E(Y^) is the expected value of the estimator Y^. It is the measurement of the model that how well it fits the data.

- Variance is the measure of spread in data from its mean position. In machine learning variance is the amount by which the performance of a predictive model changes when it is trained on different subsets of the training data. More specifically, variance is the variability of the model that how much it is sensitive to another subset of the training dataset. i.e. how much it can adjust on the new subset of the training dataset. Let Y be the actual values of the target variable, and Y^ be the predicted values of the target variable. Then the variance of a model can be measured as the expected value of the square of the difference between predicted values and the expected value of the predicted values. Variance=$E[(Y\hat{}-E[Y\hat{}])\hat{}2]$ where E[Y^] is the expected value of the predicted values. Here expected value is averaged over all the training data.
- Variance testing low, training high(overfitting)
- Bias training less, testing high (underfitting)

Evaluation Metrics in Machine Learning

(For Classification Problems)

True positive (TP): It represents an outcome in which positive samples are accurately predicted as positive by the model.

True negative (TN): It represents an outcome in which negative samples are accurately predicted as negative samples by the model.

False positive (FP): It represents an outcome in which negative samples are incorrectly predicted as positive by the model.

False negative (FN): It represents an outcome in which positive samples are wrongly predicted as negative samples by the model.

Based on these 4 outcomes, a set of model performance metrics are given below.

True Positive Rate (TPR) (also known as Sensitivity): The probability that positive samples will predict positive.

$$True\ Positive\ Rate\ (TPR)\ or\ Sensitivity = \frac{TP}{TP + FN}, \quad TPR \in [0,1]$$

False Positive Rate (FPR): The probability that negative samples will predict positive.

$$False\ Positive\ Rate\ (FPR) = \frac{FP}{FP + TN}, \quad FPR \in [0,1]$$

Precision: It estimates positive sample predictions that are genuinely from the positive class label.

$$Precision = \frac{TP}{TP + FP}, \quad Precision \in [0,1]$$

Recall: It estimates positive sample predictions from all actual positives.

$$Recall = \frac{TP}{TP + FN}, \quad Recall \in [0,1]$$

F-measure: It balances precision and recalls both together to provide a single score.

$$F\text{-}measure = \frac{2 \cdot (Precision \cdot Recall)}{(Precision + Recall)}, \quad F-measure \in [0,1]$$

Accuracy: It gives the total correct predictions (TP+TN) made by the model.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \quad Accuracy \in [0,1]$$

The area under the receiver operating characteristic curve (auROC): It shows whether the model is capable of correctly discriminating between class labels. $auROC \in [0,1]$

Matthews correlation coefficient (MCC): It computes the correlation between actual and predicted labels and is calculated by the following formula.

$$MCC = \frac{(TP \cdot TN) - (FP \cdot FN)}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}}, \quad MCC \in [-1,1]$$

What is Entropy, Information Gain, Gini Index & Gain Ratio ?

1.Entropy(S):

- Entropy is a measure of impurity or uncertainty in a dataset.
- In the context of decision trees, entropy is used to quantify the amount of information disorder or unpredictability in the data before and after splitting based on an attribute.

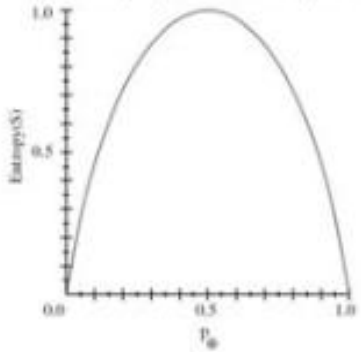
$$Entropy(S) = -\sum_{i=1}^{c} p_i \log_2(p_i)$$

Where:

S is the dataset at a given node.

c is the number of classes in the dataset.

Pi is the proportion of examples in class iii in dataset S.



- The **entropy is 0** if all members of S belong to the same class (highly pure dataset)
- The **entropy is 1** when the collection contains an equal number of positive and negative examples (mixed dataset/impure)
- If the collection contains **unequal numbers of positive and negative examples**, the **entropy is between 0 and 1**

Example:
Suppose S is a collection of 14 examples of some boolean concept, including 9 positive and 5 negative examples. Then the entropy of S relative to this boolean classification is

$$Entropy([9+, 5-]) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14)$$
$$= 0.940$$

2. Information Gain:

- Information gain is the reduction in entropy or uncertainty achieved by splitting the data on a particular attribute.
- The decision tree algorithm selects the attribute that maximizes information gain at each node.
- Constructing a decision tree is all about finding an attribute that returns the highest information gain and the smallest entropy.

$$Information\,Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|Sv|}{|S|} Entropy(Sv)$$

Where:

S is the dataset at a given node.

A is the attribute being considered for splitting.

Values(A) is the set of all possible values of attribute A.

Sv is the subset of S where attribute A has value v.

|S| is the total number of examples in S.

|Sv| is the number of examples in subset Sv.

Example: Information gain

Let, $Values(Wind) = \{Weak, Strong\}$

$S = [9+, 5-]$

$S_{Weak} = [6+, 2-]$

$S_{Strong} = [3+, 3-]$

Information gain of attribute *Wind*:

$Gain(S, Wind) = Entropy(S) - 8/14\ Entropy(S_{Weak}) - 6/14\ Entropy(S_{Strong})$

$= 0.94 - (8/14) * 0.811 - (6/14) * 1.00$

$= 0.048$

3. Gini Index:

- Gini index measures the impurity or the likelihood of an incorrect classification of a randomly chosen element in the dataset if it were randomly labeled according to the distribution of labels in the subset.
- It is used to evaluate splits in the dataset. A low Gini index suggests that a particular attribute provides good separation of the classes.
- Higher value of Gini index implies higher inequality, higher heterogeneity.

$$Gini(S) = 1 - \sum_{i=1}^{c} (pi)^2$$

Where:

S is the dataset at a given node.

c is the number of classes in the dataset.

pi is the proportion of examples in class i in dataset S.

- **Example Calculation**: Suppose we have a dataset S with 10 examples, where 6 examples belong to class A and 4 examples belong to class B.

Proportion of class A: $P_A = \frac{6}{10} = 0.6$

Proportion of class B: $P_B = \frac{4}{10} = 0.4$

$$Gini(S) = 1 - (0.6^2 + 0.4^2)$$

$$Gini(S) = 0.48$$

@data_science_school

4. Gain Ratio:

- Gain ratio is an extension of information gain that takes into account the intrinsic information of a split by normalizing the information gain using the split information.
- Gain ratio adjusts for the bias towards attributes with a large number of distinct values.

$$Gain\ Ratio(S, A) = \frac{Information\ Gain(S, A)}{Split\ Information(S, A)}$$

Where:

$$Split\ Information(S, A) = -\sum_{v \in Values(A)} \frac{|Sv|}{|S|} \log_2 \left(\frac{|Sv|}{|S|} \right)$$

What is Regularization?

Regularization is one of the most important concepts of machine learning. It is a technique to prevent the model from overfitting by adding extra information to it. Sometimes the machine learning model performs well with the training data but does not perform well with the test data. It means the model is not able to predict the output when deals with unseen data by introducing noise in the output, and hence the model is called overfitted. This problem can be deal with the help of a regularization technique.
This technique can be used in such a way that it will allow to maintain all variables or features in the model by reducing the magnitude of the variables. Hence, it maintains accuracy as well as a generalization of the model.

How does Regularization Work?

@data_science_learn

Regularization works by adding a penalty or complexity term to the complex model. Let's consider the simple linear regression equation:

$$y = \beta 0 + \beta 1x1 + \beta 2x2 + \beta 3x3 + \cdots + \beta nxn + b$$

In the above equation, Y represents the value to be predicted

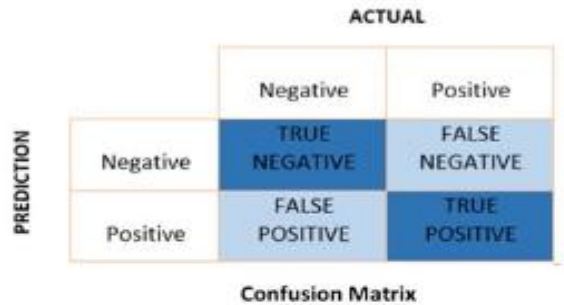
X1, X2, ...Xn are the features for Y. β0,β1,.....βn are the weights or magnitude attached to the features, respectively. Here represents the bias of the model, and b represents the intercept.

Linear regression models try to optimize the β0 and b to minimize the cost function. The equation for the cost function for the linear model is given below:

$$\sum_{i=1}^{M} (y_i - y'_i)^2 = \sum_{i=1}^{M} (y_i - \sum_{j=0}^{n} \beta_j * Xij)^2$$

What is Confusion Matrix?

It is a matrix of size 2*2 for binary classification with actual values one axis and predicted on another.

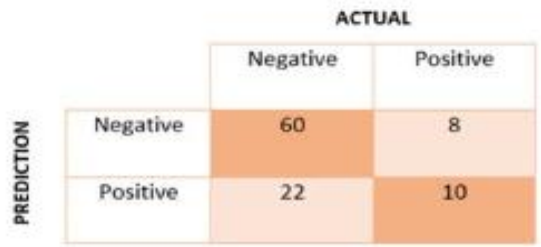


Confusion Matrix

The confusing terms in the confusion matrix: **true positive, true negative, false negative, and false positive** with an example.

EXAMPLE

A machine learning model is trained to predict tumor in patients. The test dataset consists of 100 people.



Confusion Matrix for tumor detection

True Positive (TP) — model correctly predicts the positive class (prediction and actual both are positive). In the above example, **10 people** who have tumors are predicted positively by the model.

True Negative (TN) — model correctly predicts the negative class (prediction and actual both are negative). In the above example, **60 people** who don't have tumors are predicted negatively by the model.

False Positive (FP) — model gives the wrong prediction of the negative class (predicted-positive, actual-negative). In the above example, **22 people** are predicted as positive of having a tumor, although they don't have a tumor. FP is also called a **TYPE I** error.

False Negative (FN) — model wrongly predicts the positive class (predicted-negative, actual-positive). In the above example, **8 people** who have tumors are predicted as negative. FN is also called a **TYPE II** error.

With the help of these four values, we can calculate True Positive Rate (TPR), False Negative Rate (FPR), True Negative Rate (TNR), and False Negative Rate (FNR).

$$TPR = \frac{TP}{Actual\ Positive} = \frac{TP}{TP + FN}$$

$$FNR = \frac{FN}{Actual\ Positive} = \frac{FN}{TP + FN}$$

$$TNR = \frac{TN}{Actual\ Negative} = \frac{TN}{TN + FP}$$

$$FPR = \frac{FP}{Actual\ Negative} = \frac{FP}{TN + FP}$$

Even if data is imbalanced, we can figure out that our model is working well or not. For that, **the values of TPR and TNR should be high, and FPR and FNR should be as low as possible.**

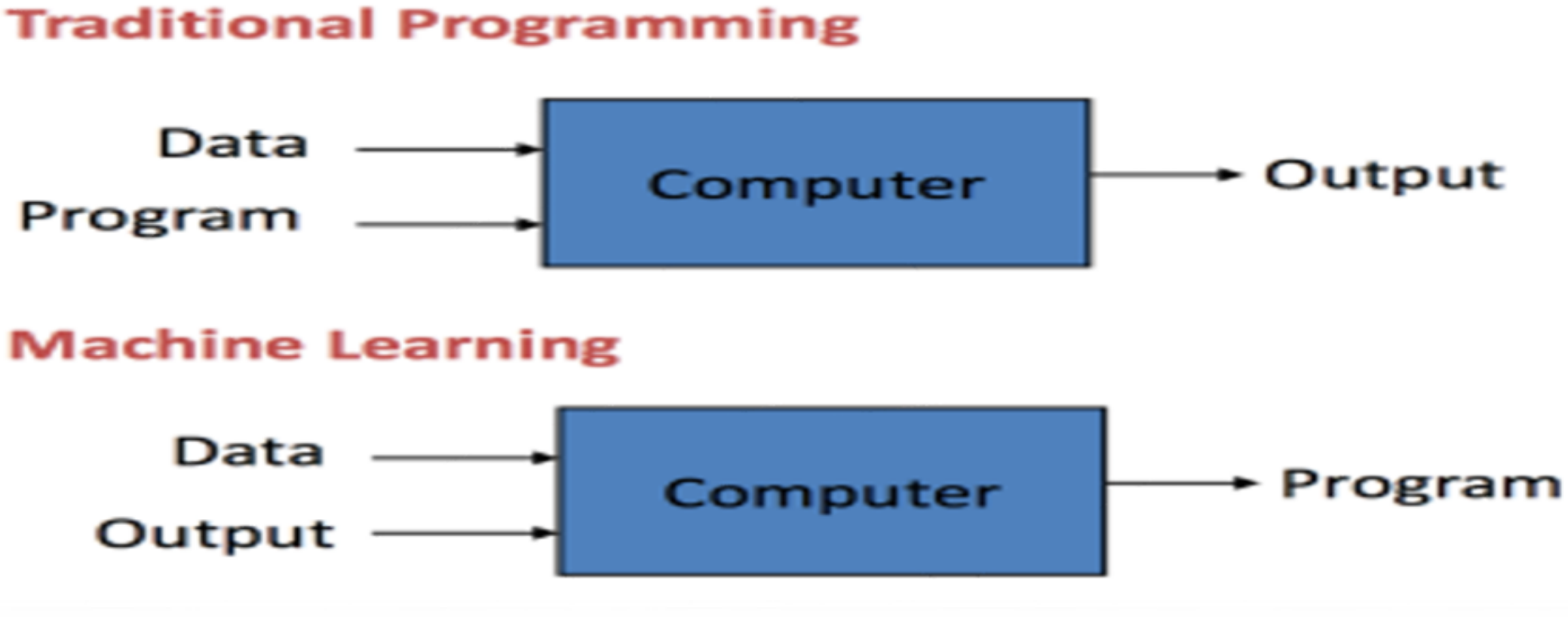
Introduction to AI

- Artificial Intelligence is composed of two words **Artificial** and **Intelligence**, where Artificial defines *"man-made,"* and intelligence defines *"thinking power"*, hence AI means *"a man-made thinking power.“*
- "It is a branch of computer science by which we can create intelligent machines which can behave like a human, think like humans, and able to make decisions."

How was Artificial Intelligence born?

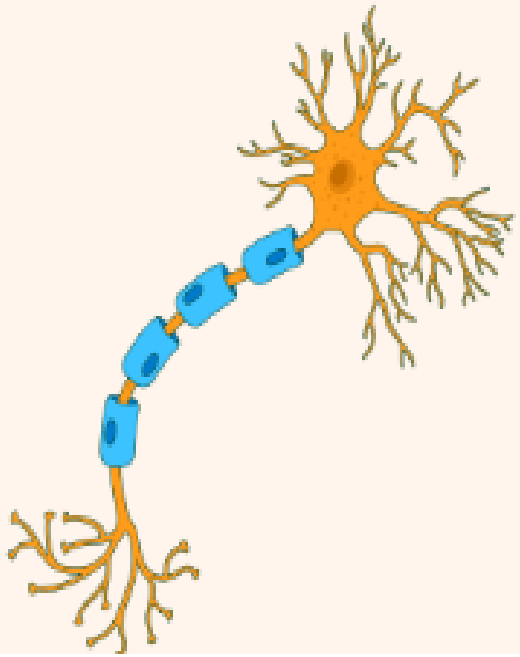
- The idea of creating machines that mimic human intelligence was present even in ancient times, with myths and legends about automatons and thinking machines.
- In 1943 Warren McCulloch and Walter Pitts presented their model of artificial neurons, considered the first artificial intelligence, even though the term did not yet exist.
- Later, in 1950, the British mathematician Alan Turing published an article entitled "Computing machinery and intelligence" in the magazine Mind where he asked the question: Can machines think? He proposed an experiment that came to be known as the Turing Test, which, according to the author, would make it possible to determine whether a machine could have intelligent behaviour similar to or indistinguishable from that of a human being.
- John McCarthy coined the term "artificial intelligence" in 1956 and drove the development of the first AI programming language, LISP, in the 1960s.

Traditional Programming v/s Machine Learning



History of artificial intelligence

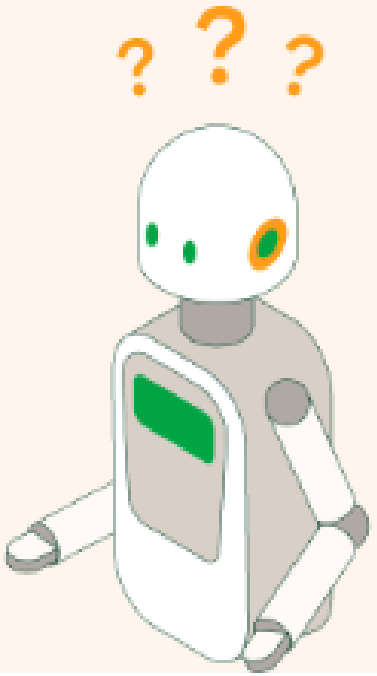
1943



American researchers Warren McCulloch and Walter Pitts present their model of **artificial neurons, considered the first artificial intelligence.**

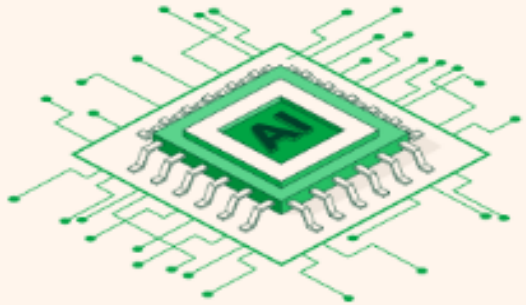
History of artificial intelligence

1950



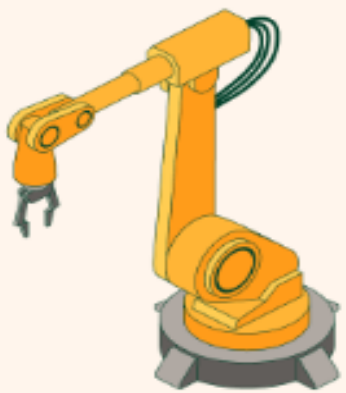
British mathematician Alan Turing proposes a **test for machine intelligence**. If a machine can fool humans into thinking it is human, then it has intelligence.

1956



American computer scientist John McCarthy **coins the term "artificial intelligence"** to describe "the science and engineering of making machines intelligent".

1961



General Motors installs the **first industrial robot**, Unimate, to replace humans in assembly tasks.

1964

Joseph Wizenbaum develops the **first natural language processing computer program**, ELIZA, which simulates human conversation.

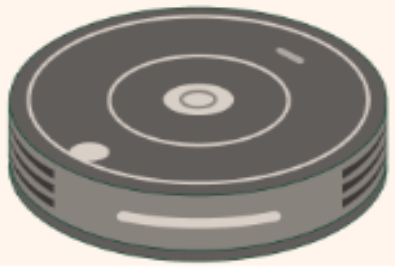
1966

Shakey, the first general-purpose mobile robot **capable of reasoning about its own actions**, is launched. It is considered the forerunner of autonomous cars.

1997

IBM's Deep Blue supercomputer **beats world chess champion** Garry Kasparov in one match.

2002



Roomba, the **first mass-produced robotic hoover** sold by iRobot, is launched, capable of roaming and cleaning in the home.

2011

Apple integrates Siri, a **virtual assistant** with a voice interface, into its iPhone 4S.

IBM's Watson system, capable of answering questions asked in natural language, **wins first prize** in the popular US TV quiz show Jeopardy!

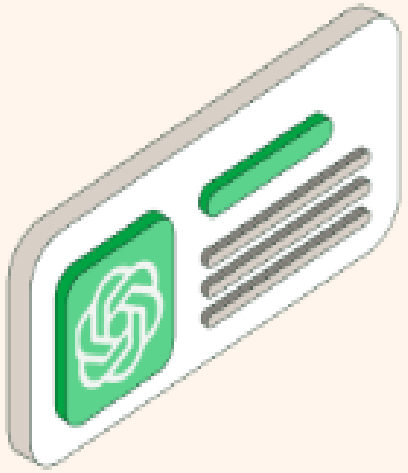
2014

The Eugene computer program **passes the Turing** Test by convincing a third of the judges participating in the experiment that it was a human being.

2016

DeepMind's AlphaGo programme, based on a deep neural network, **beats Lee Sodol, the world Go champion** in five games.

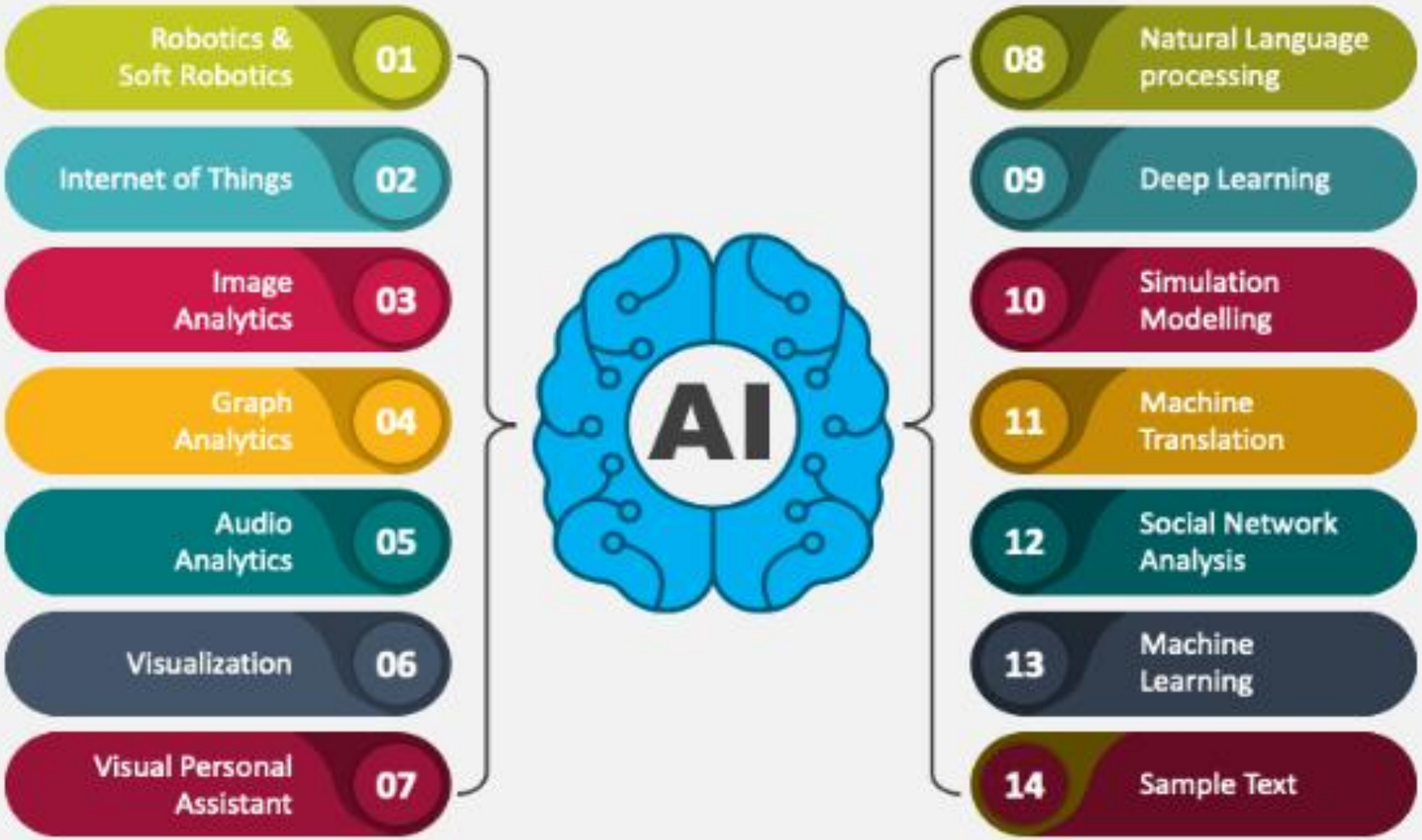
2022

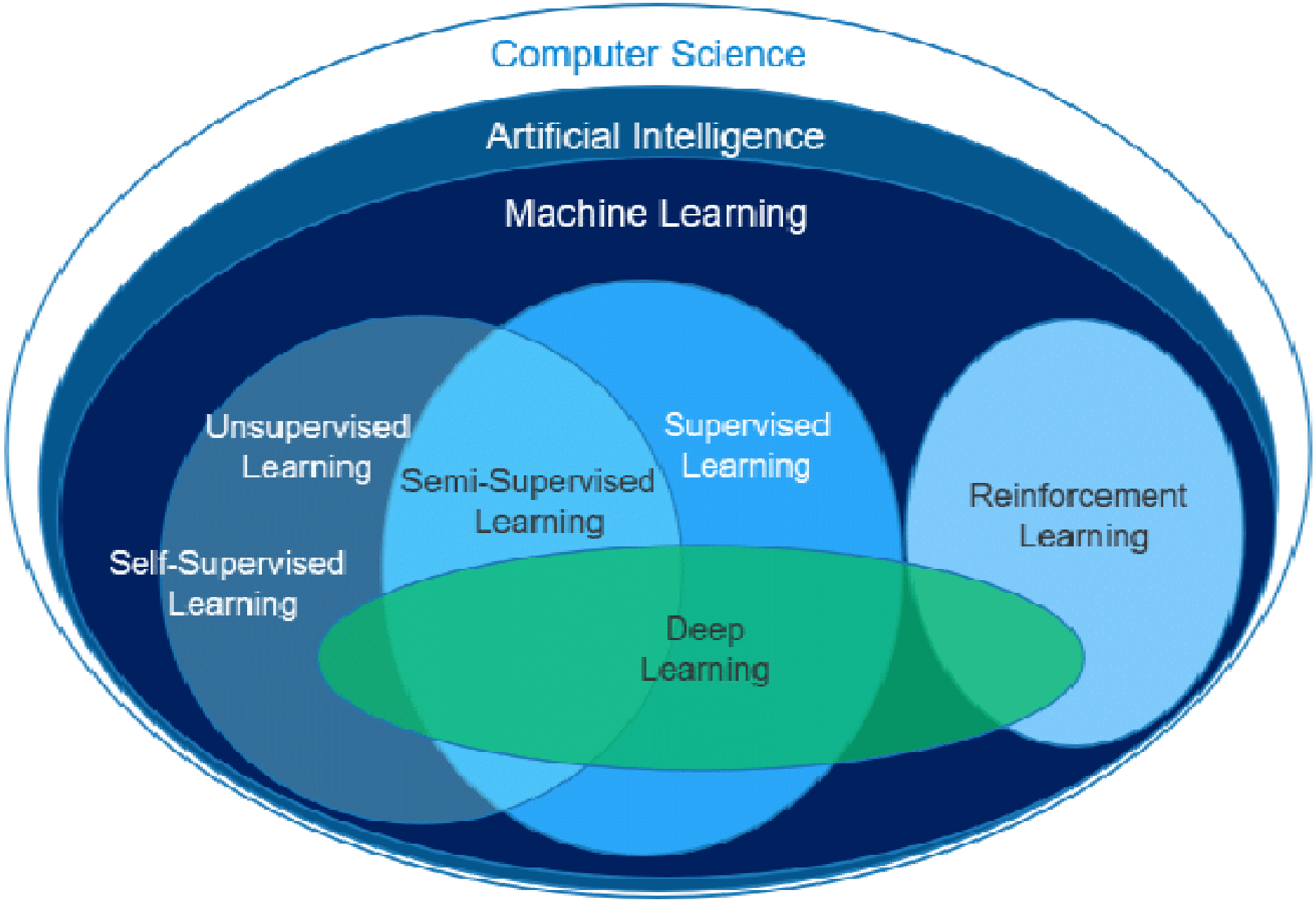


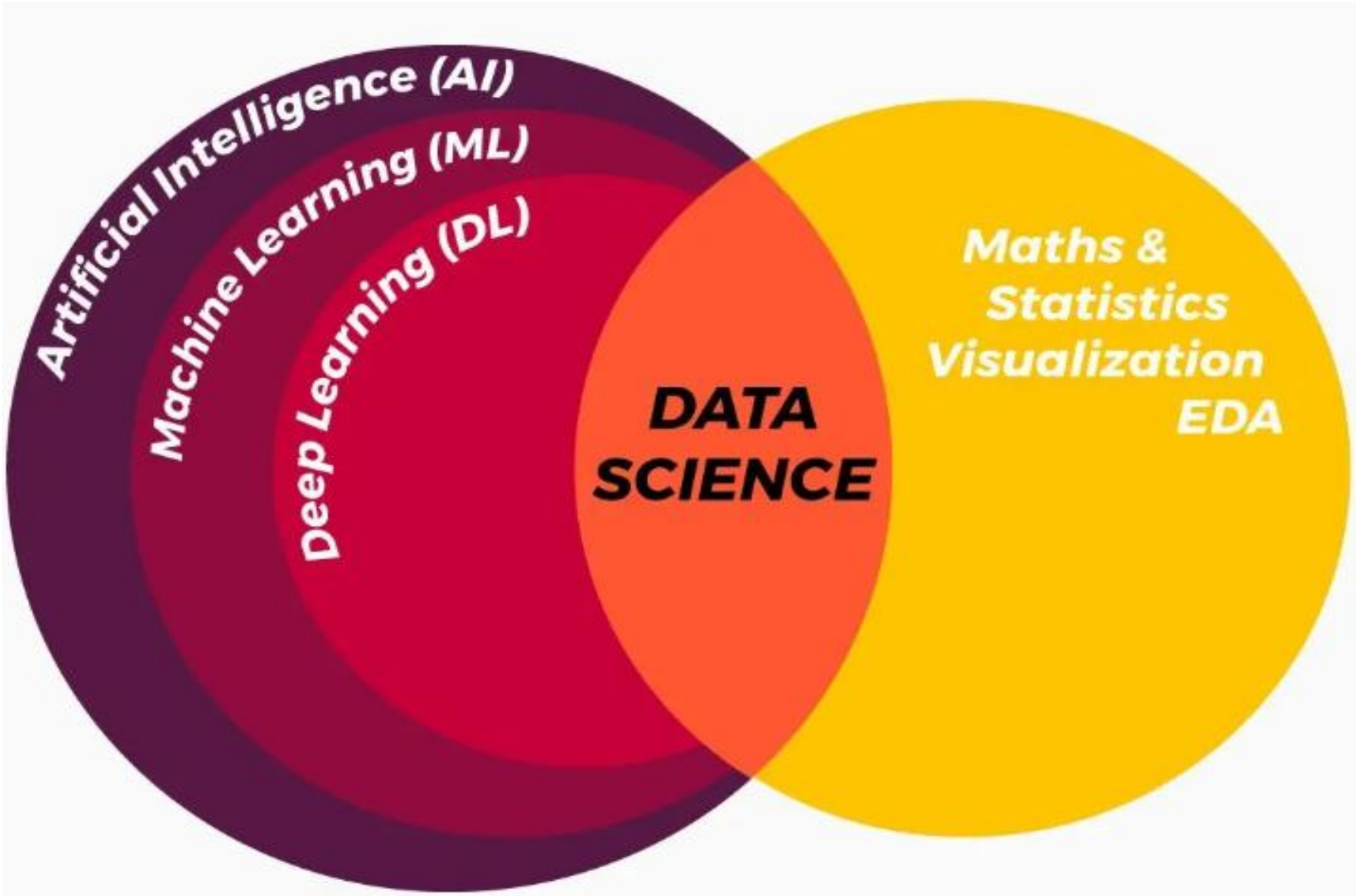
OpenAI launches ChatGPT, an **artificial intelligence chatbot** application trained to hold conversations and answer questions, to the public.

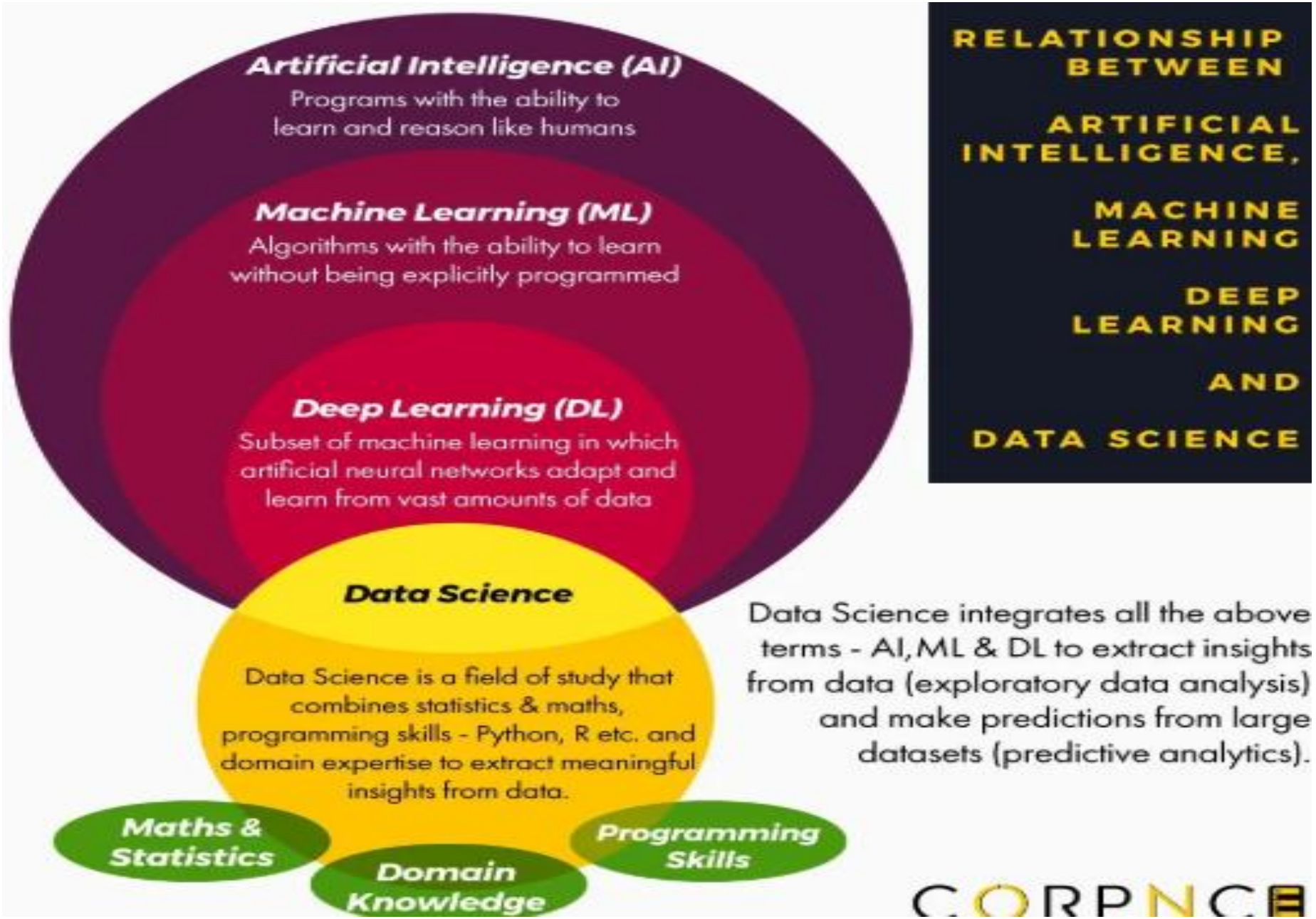
APPLICATIONS OF AI

Possible Applications for Artificial Intelligence

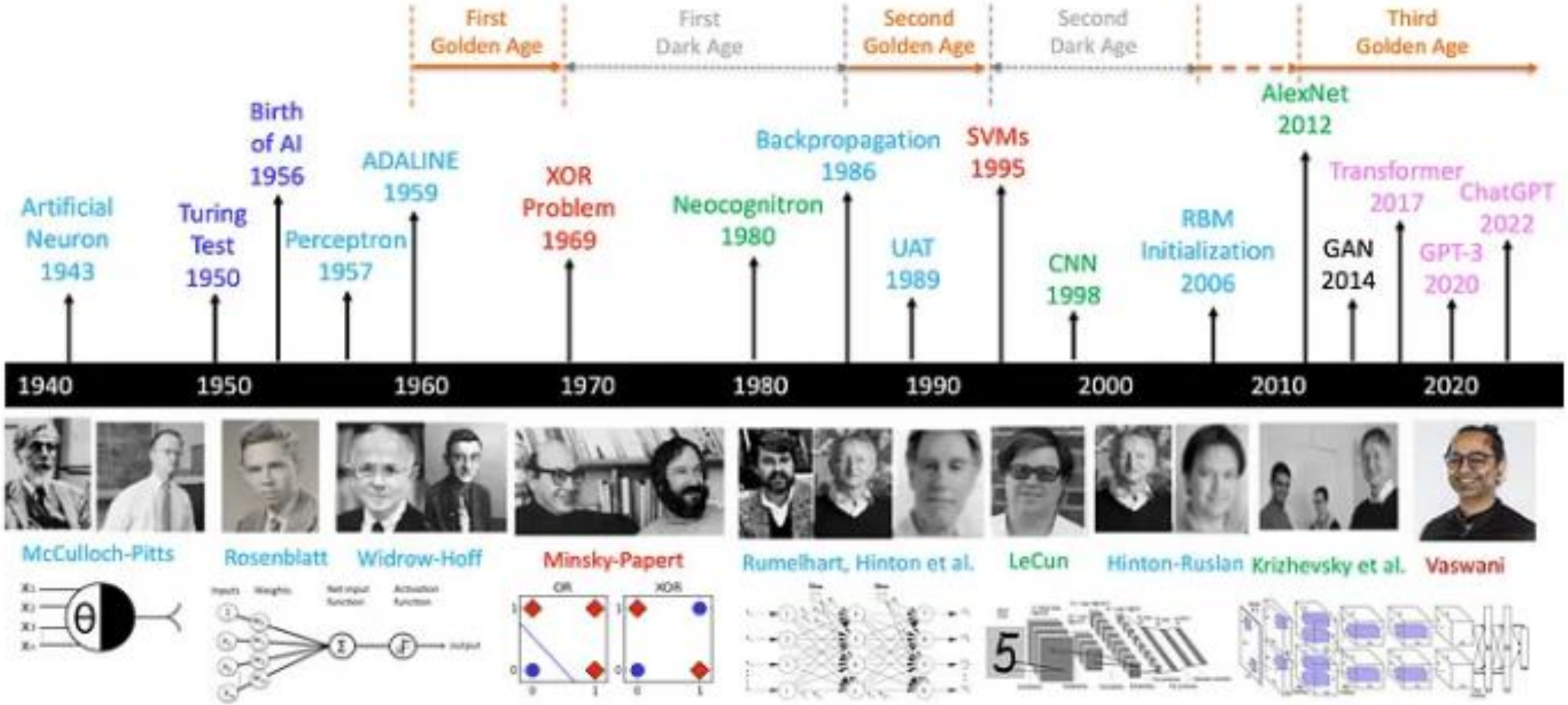



Computer Science
Artificial Intelligence
Machine Learning
Unsupervised Learning
Semi-Supervised Learning
Supervised Learning
Reinforcement Learning
Self-Supervised Learning
Deep Learning

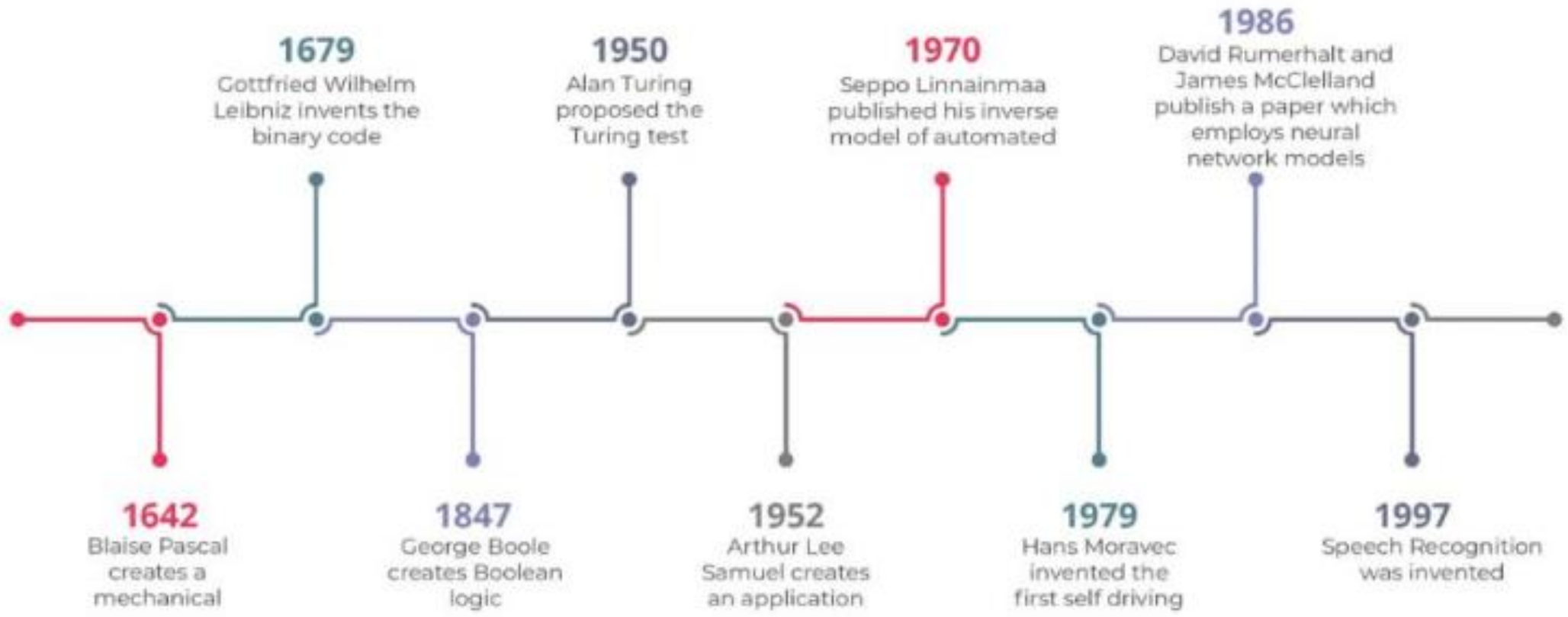

Artificial Intelligence (AI)
Machine Learning (ML)
Deep Learning (DL)
DATA SCIENCE
Maths &
Statistics
Visualization
EDA

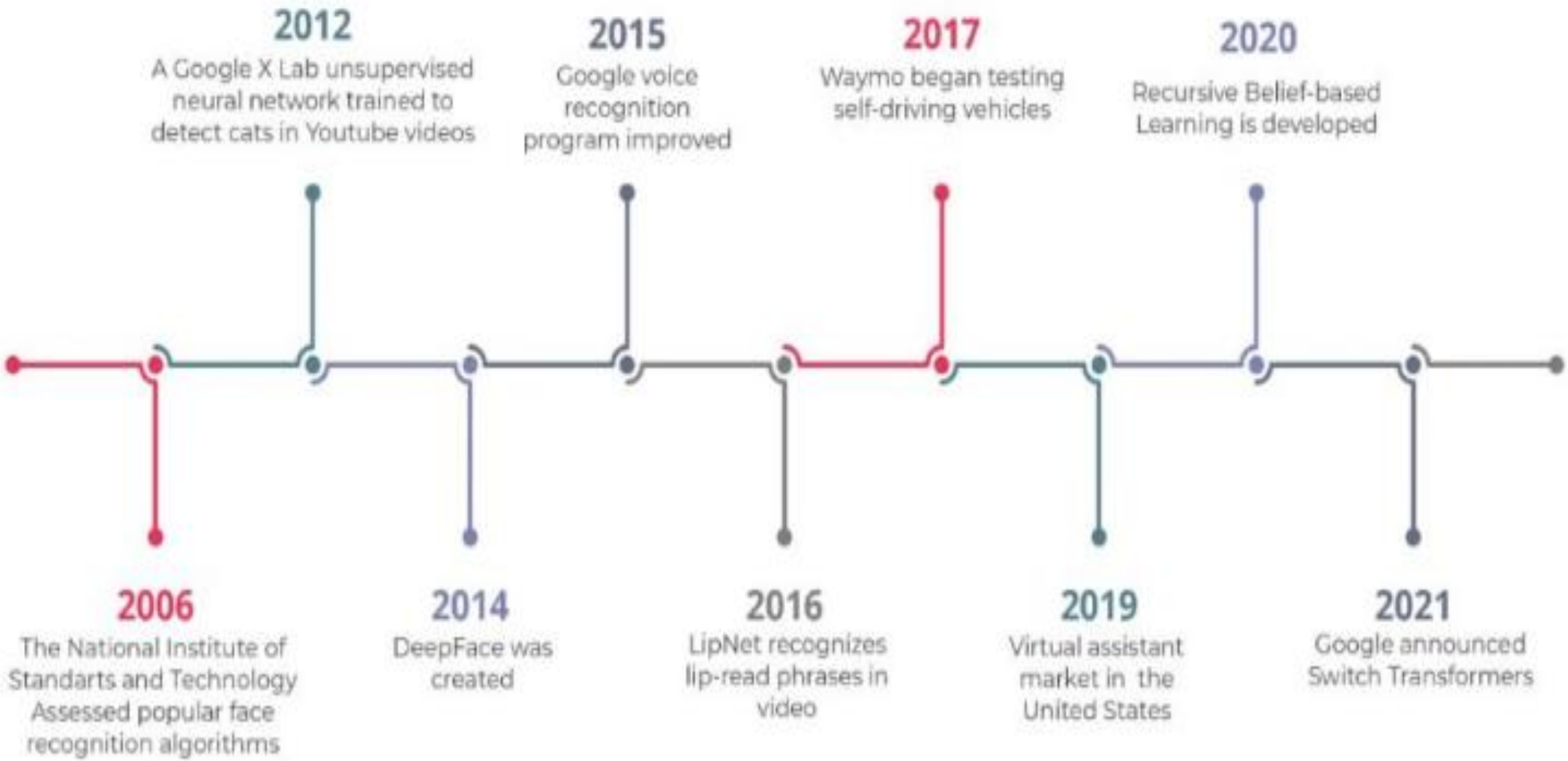

Artificial Intelligence (AI)
Programs with the ability to learn and reason like humans
Machine Learning (ML)
Algorithms with the ability to learn without being explicitly programmed
Deep Learning (DL)
Subset of machine learning in which artificial neural networks adapt and learn from vast amounts of data
Data Science
Data Science is a field of study that combines statistics & maths, programming skills - Python, R etc. and domain expertise to extract meaningful insights from data.
Maths & Statistics
Domain Knowledge
Programming Skills
RELATIONSHIP BETWEEN
ARTIFICIAL INTELLIGENCE,
MACHINE LEARNING
DEEP LEARNING
AND
DATA SCIENCE
Data Science integrates all the above terms - AI, ML & DL to extract insights from data (exploratory data analysis) and make predictions from large datasets (predictive analytics).

A Brief History of AI with Deep Learning



Walk along the machine learning timeline




2012
A Google X Lab unsupervised neural network trained to detect cats in Youtube videos
2015
Google voice recognition program improved
2017
Waymo began testing self-driving vehicles
2020
Recursive Belief-based Learning is developed
2006
The National Institute of Standarts and Technology Assessed popular face recognition algorithms
2014
DeepFace was created
2016
LipNet recognizes lip-read phrases in video
2019
Virtual assistant market in the United States
2021
Google announced Switch Transformers

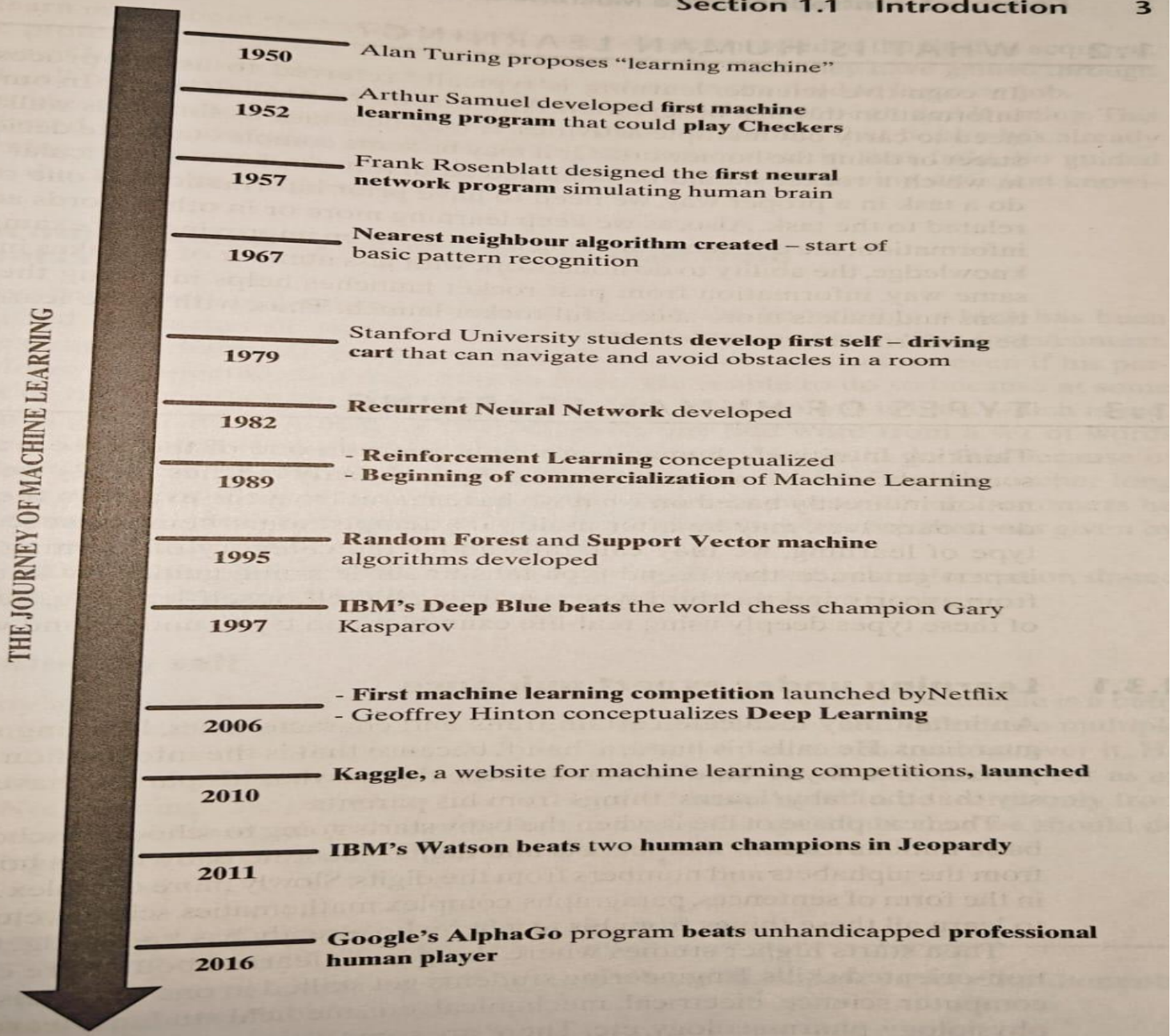
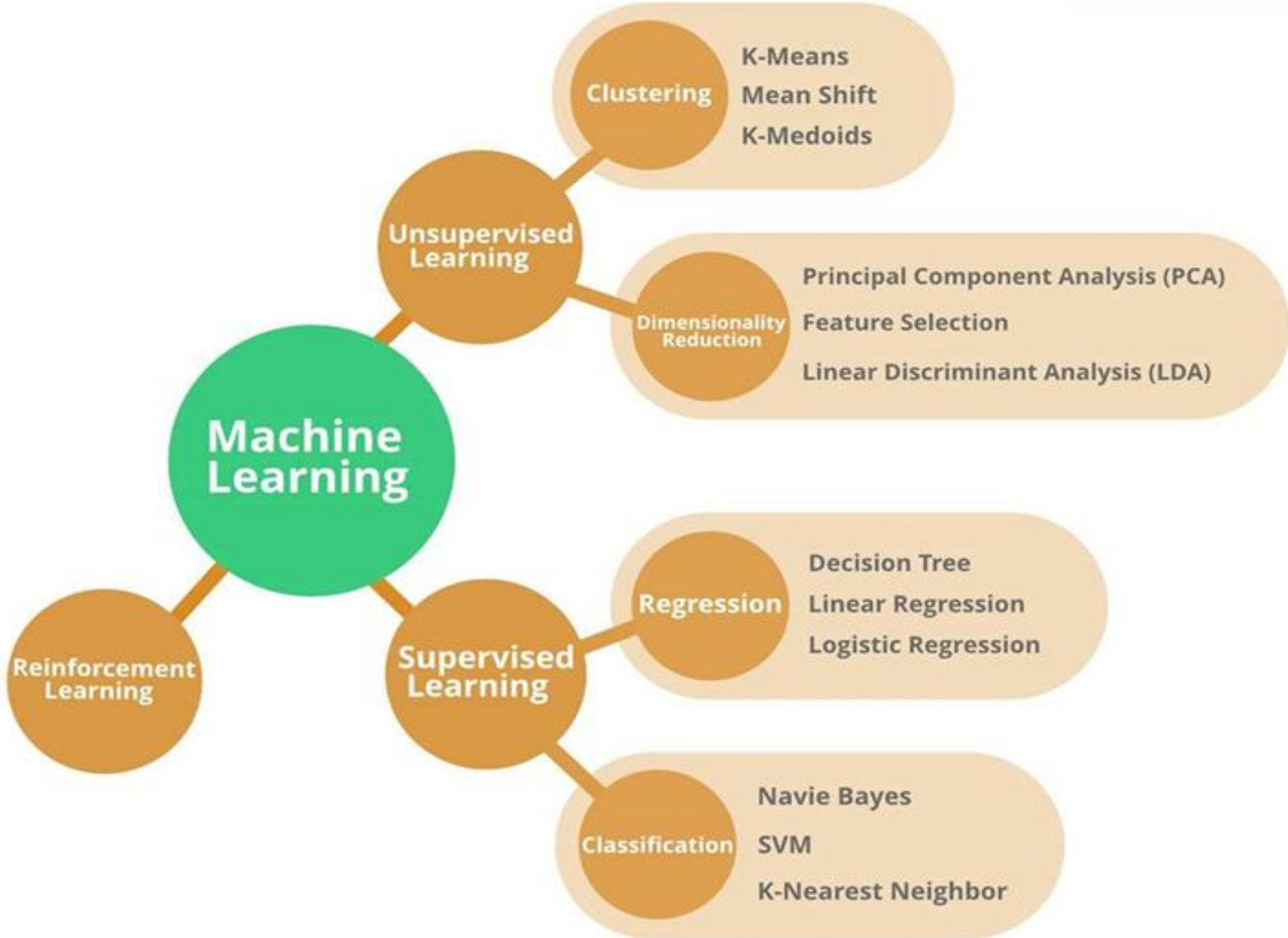
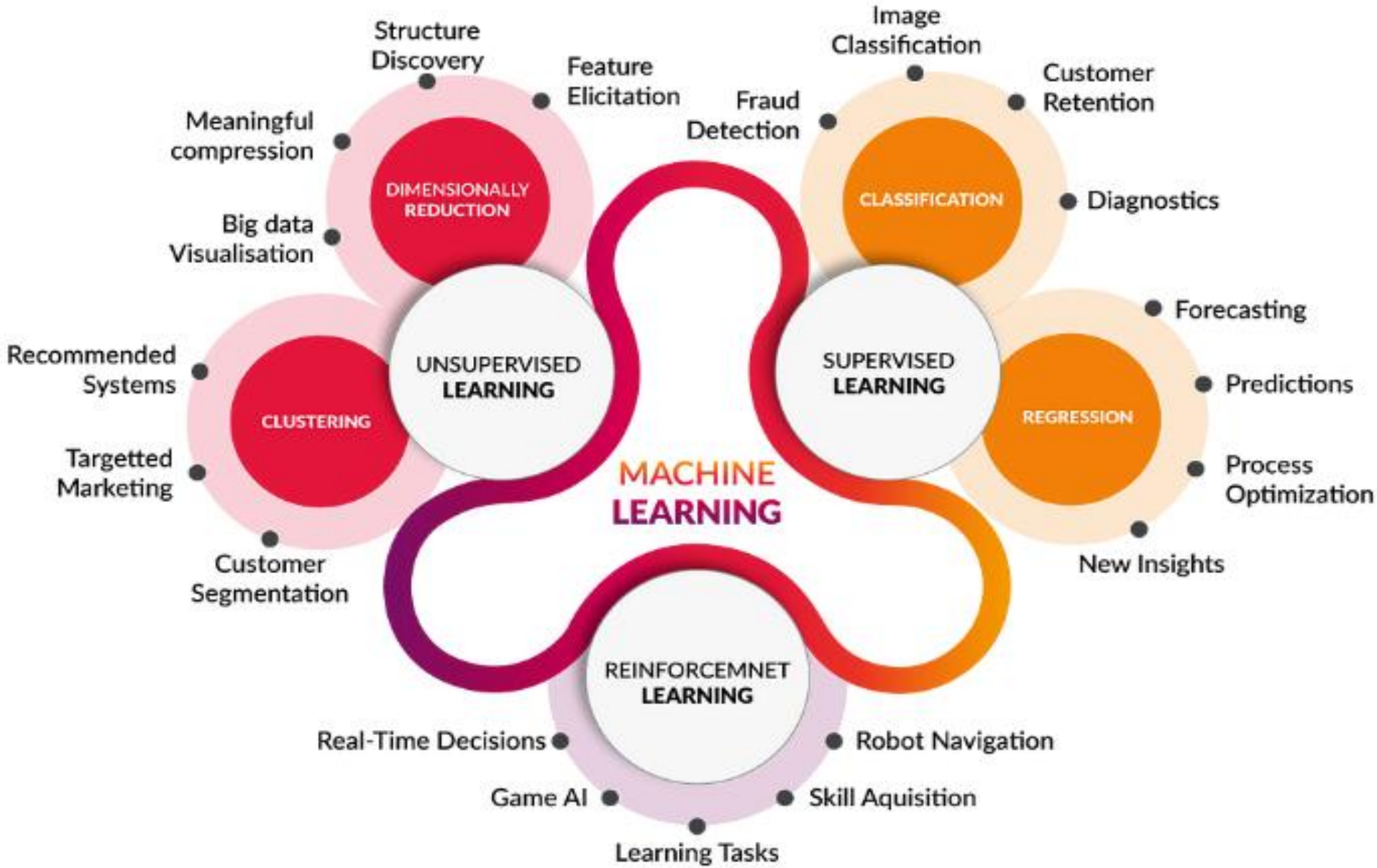


FIG. 1.1
Evolution of machine learning

Definition of ML

- Tom M. Mitchell provided a widely quoted, more formal definition of the algorithms studied in the machine learning field: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.“
- This definition of the tasks in which machine learning is concerned offers a fundamentally operational definition rather than defining the field in cognitive terms.
- This follows Alan Turing's proposal in his paper "Computing Machinery and Intelligence", in which the question "Can machines think?" is replaced with the question "Can machines do what we (as thinking entities) can do?"


Machine Learning
Unsupervised Learning
Clustering
K-Means
Mean Shift
K-Medoids
Dimensionality Reduction
Principal Component Analysis (PCA)
Feature Selection
Linear Discriminant Analysis (LDA)
Supervised Learning
Regression
Decision Tree
Linear Regression
Logistic Regression
Classification
Navie Bayes
SVM
K-Nearest Neighbor
Reinforcement Learning


MACHINE LEARNING
UNSUPERVISED LEARNING
SUPERVISED LEARNING
REINFORCEMNET LEARNING
DIMENSIONALLY REDUCTION
CLUSTERING
CLASSIFICATION
REGRESSION
Structure Discovery
Feature Elicitation
Meaningful compression
Big data Visualisation
Recommended Systems
Targetted Marketing
Customer Segmentation
Image Classification
Customer Retention
Fraud Detection
Diagnostics
Forecasting
Predictions
Process Optimization
New Insights
Real-Time Decisions
Robot Navigation
Game AI
Skill Aquisition
Learning Tasks

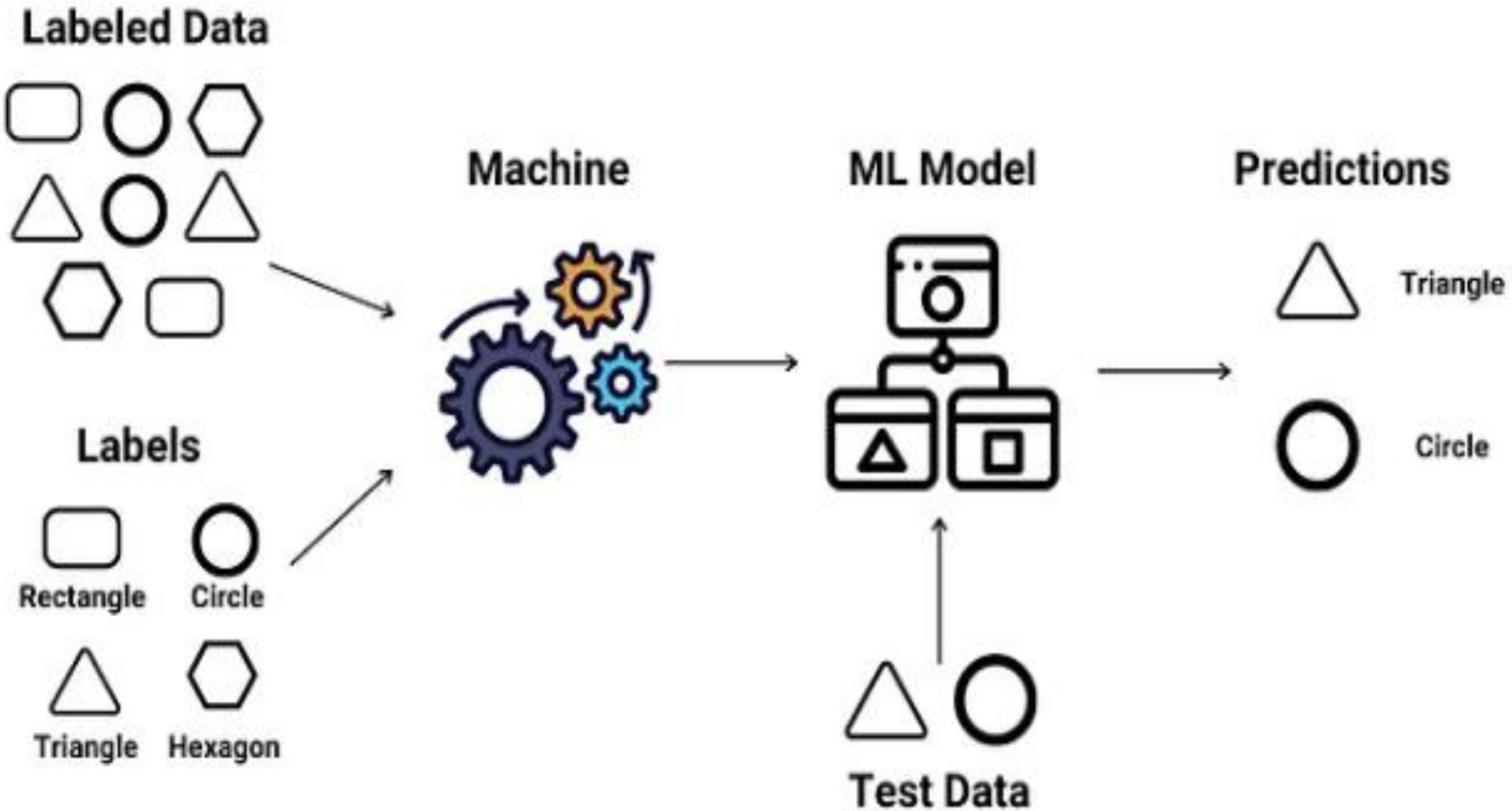
Difference between Supervised vs Unsupervised Machine Learning

Supervised Learning	Unsupervised Learning
SL algorithms are trained using labeled data	UL algorithms are trained using unlabeled data
SL model takes direct feedback to check if it is predicting correct output or not	UL model does not take any feedback
SL model predicts the output	UL model finds the hidden patterns in data
In SL, input data is provided to the model along with the output	In UL, only input data is provided to the model
The goal of SL is so that it can predict the output when it is given new data	The goal of UL is to find the hidden patterns and useful insights from the
SL needs supervision	UL does not need any supervision
SL can be categorized in Classification and Regression problems	Unsupervised Learning can be classified in Clustering and Associations problems
SL can be used where we know the input as well as corresponding outputs	UL can be used where we have only input data and no corresponding output data
SL model produces an accurate result	UL model may give less accurate result
SL is not close to true AI as in this, we first train the model for each data, and then only it can predict the correct output	UL is more close to the true AI as it learns similarly as a child learns daily routine things by his experiences
It includes various algorithms such as Linear Regression, Logistic Regression, Support Vector Machine, Multi-class Classification, Decision tree, Bayesian Logic, etc	It includes various algorithms such as Clustering, KNN, and Apriori algorithm

Supervised Machine Learning

- Supervised learning is the types of machine learning in which machines are trained using well “labelled” training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output.
- There are two types of supervised learning techniques:
- **Classification** where our result set consist of categories
- Some examples are: *Whether a patient has cancer or not, Which companies gonna go bankruptcy this year, etc.*
- **Regression** where the results are continuous values.
- E.g. *Predicting house prices, How long is it gonna take you to get home*

Supervised Learning



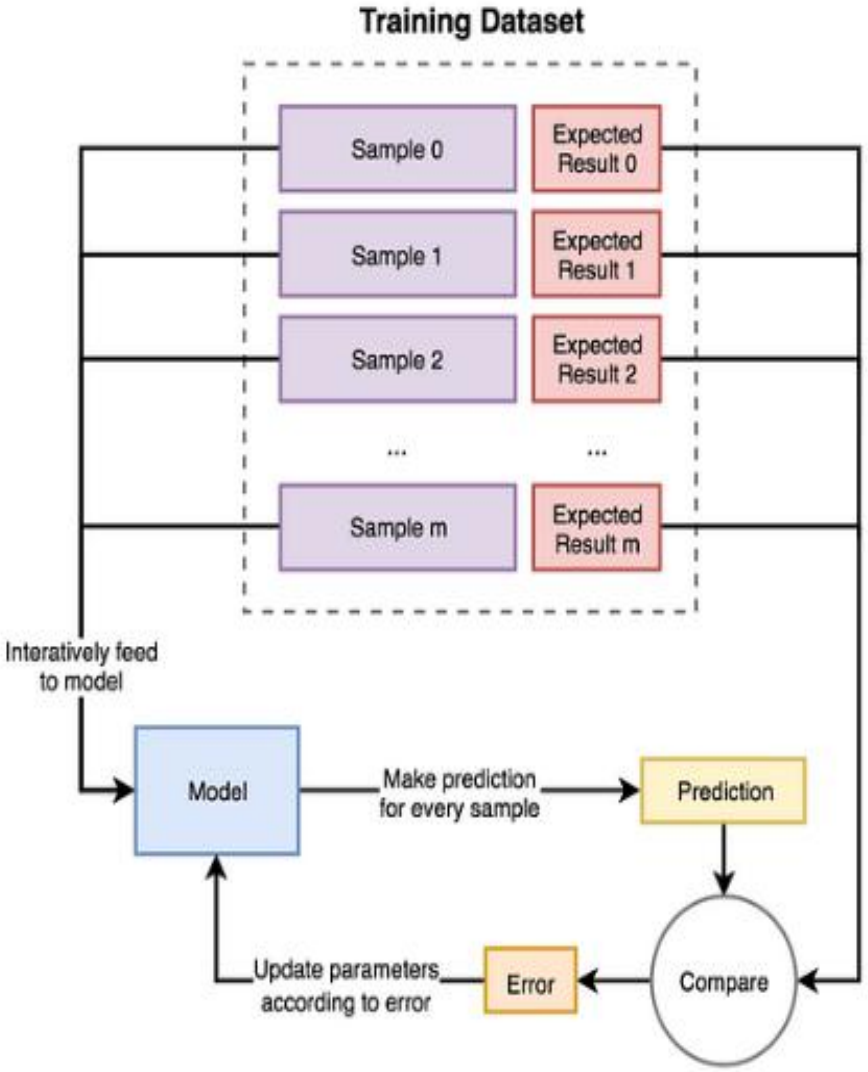


Diagram demonstrating how Supervised Learning works.

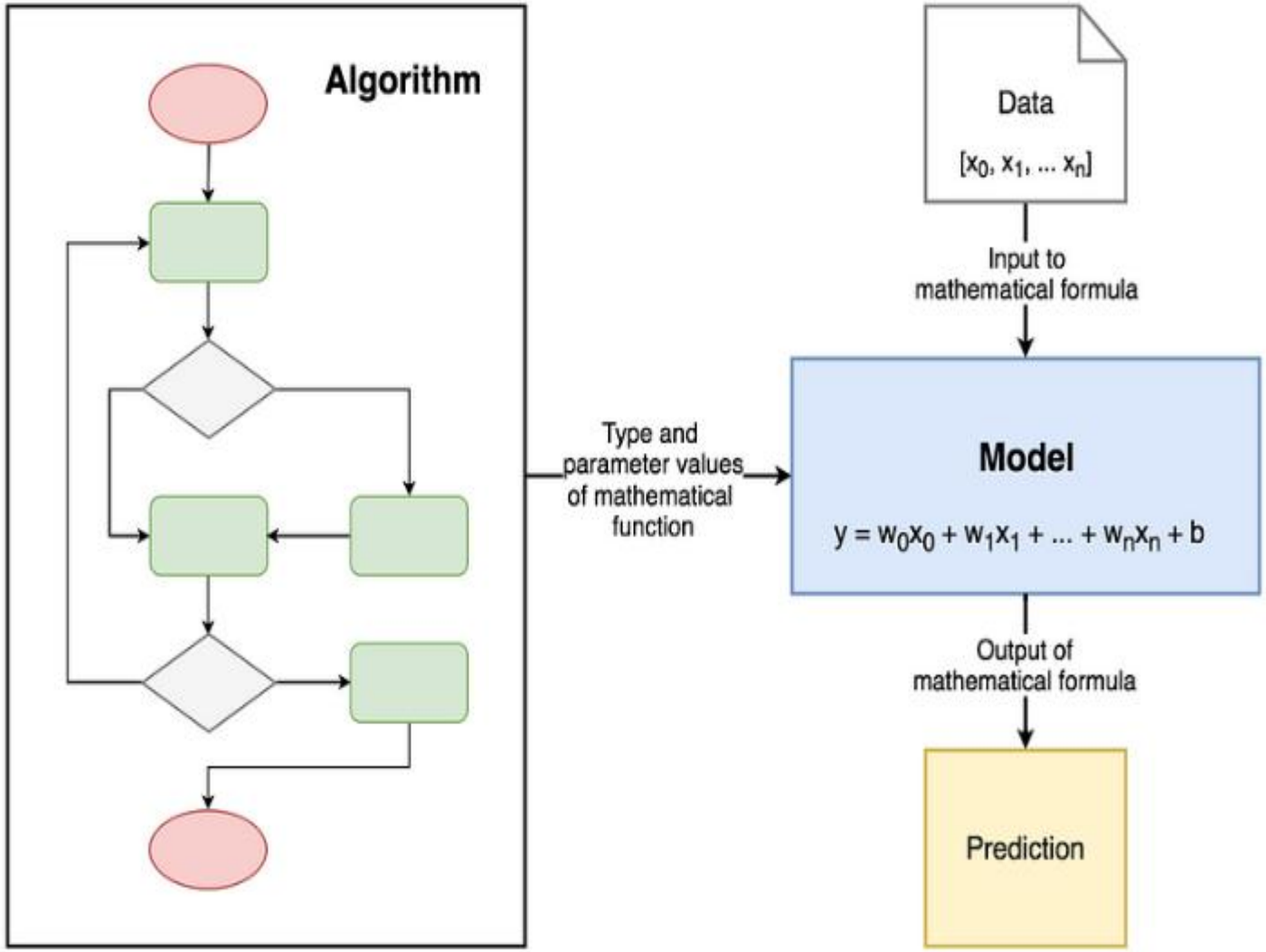
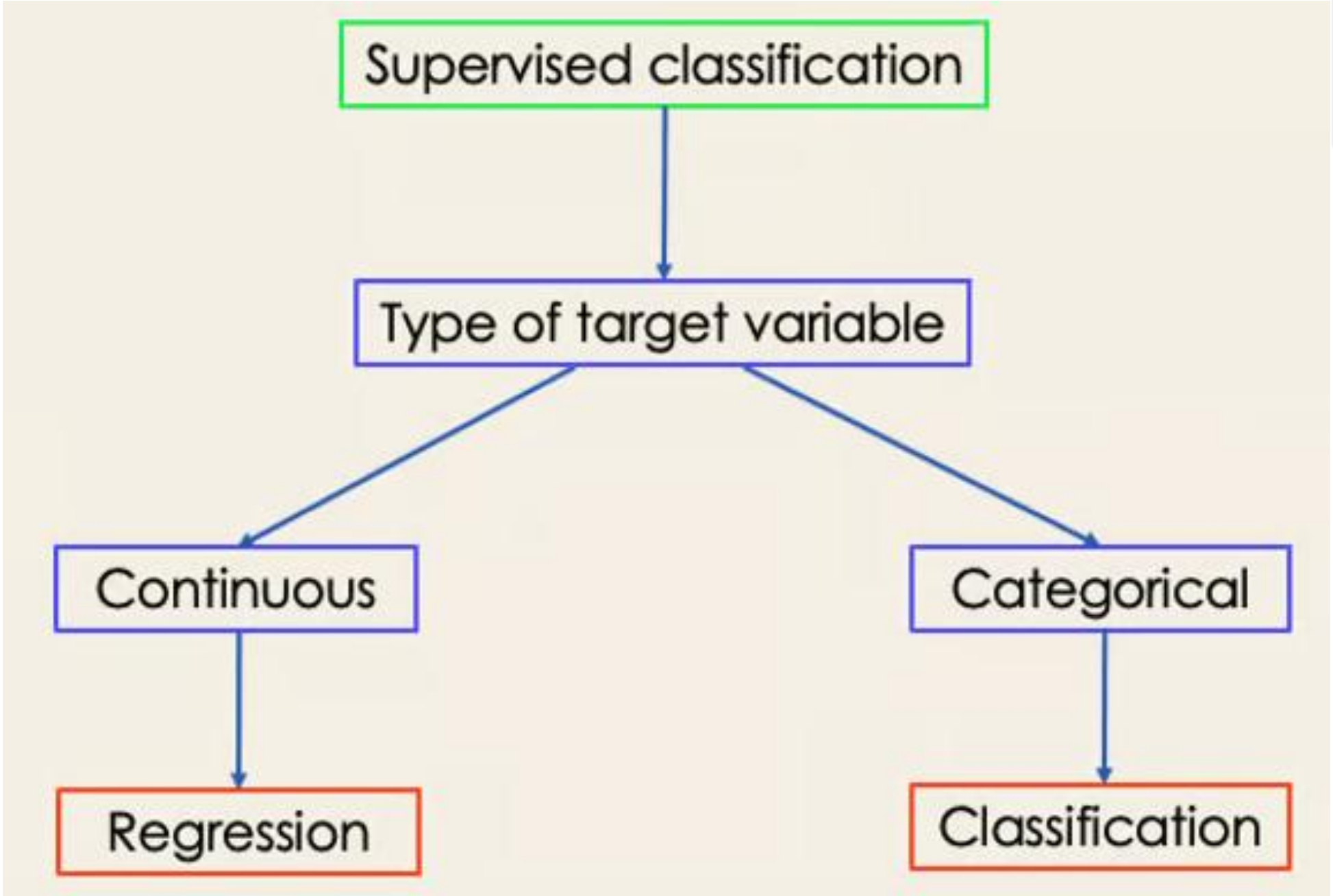
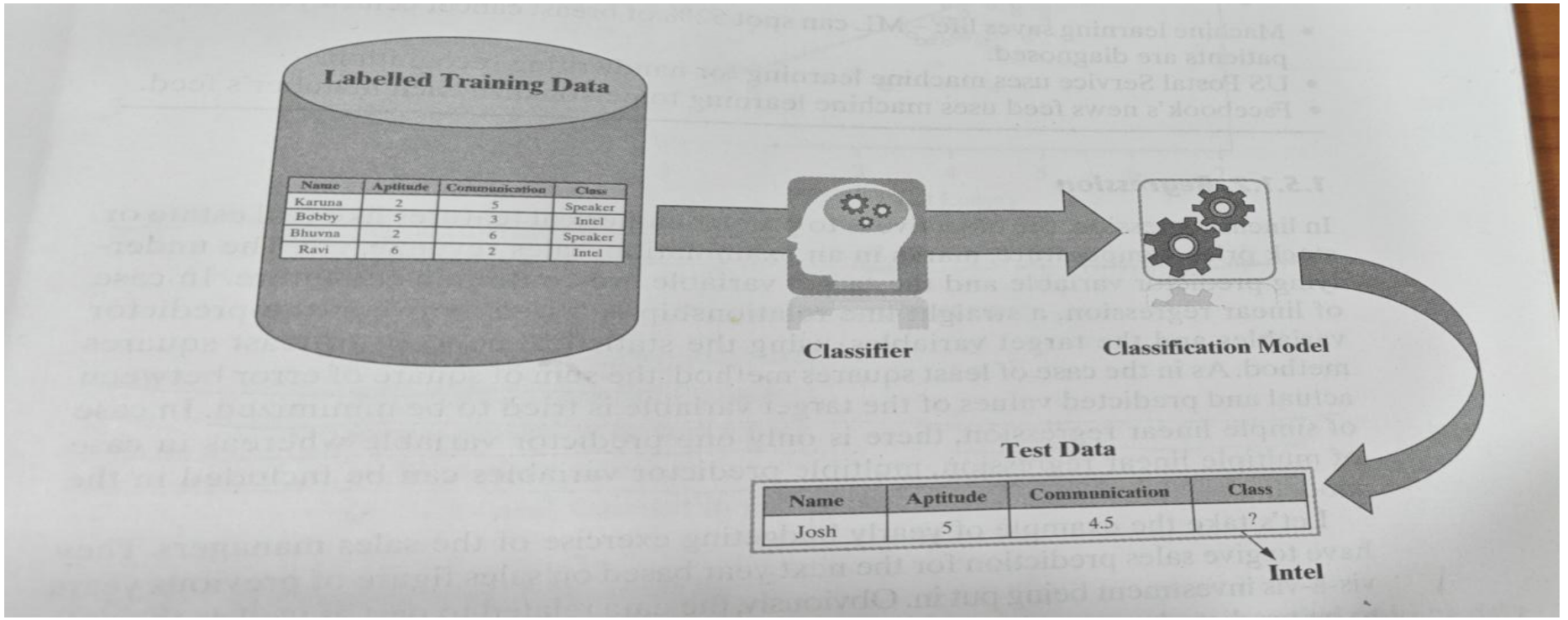


Diagram visualizing difference between Machine Learning Algorithm and Machine Learning Model.


Supervised classification
Type of target variable
Continuous
Categorical
Regression
Classification

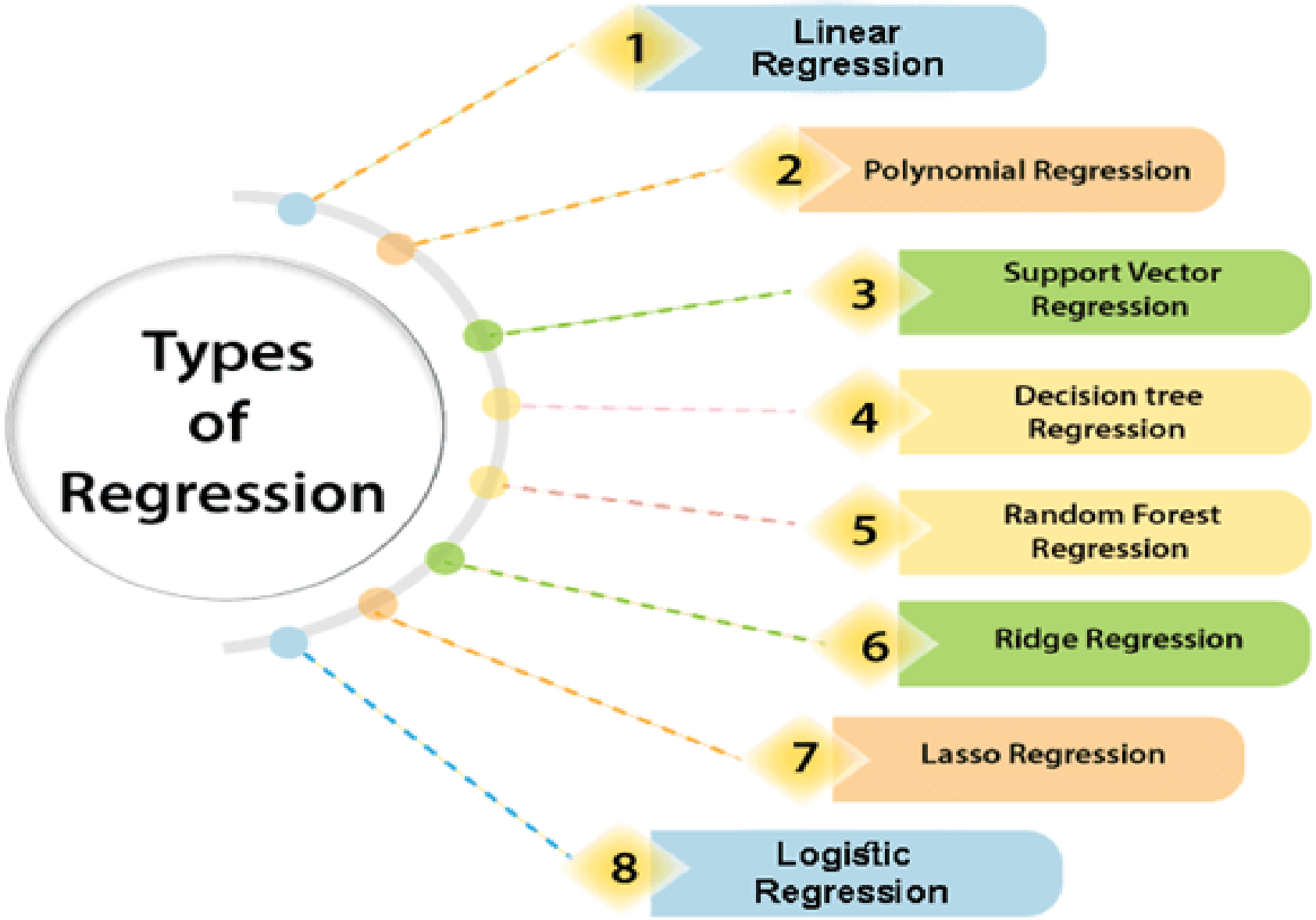
Classification

- Classification models classify our outputs to certain categories. If the number of categories are only two then it is specially called binary classification. For greater number of categories it is called multi-class classification.



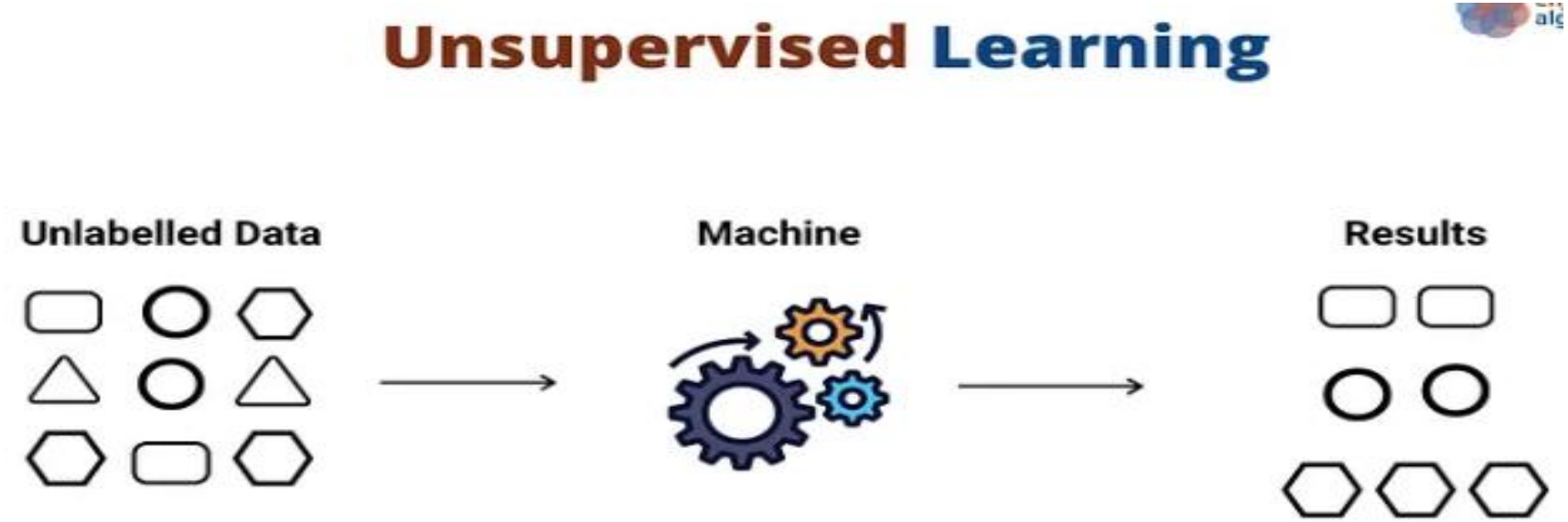
Regression

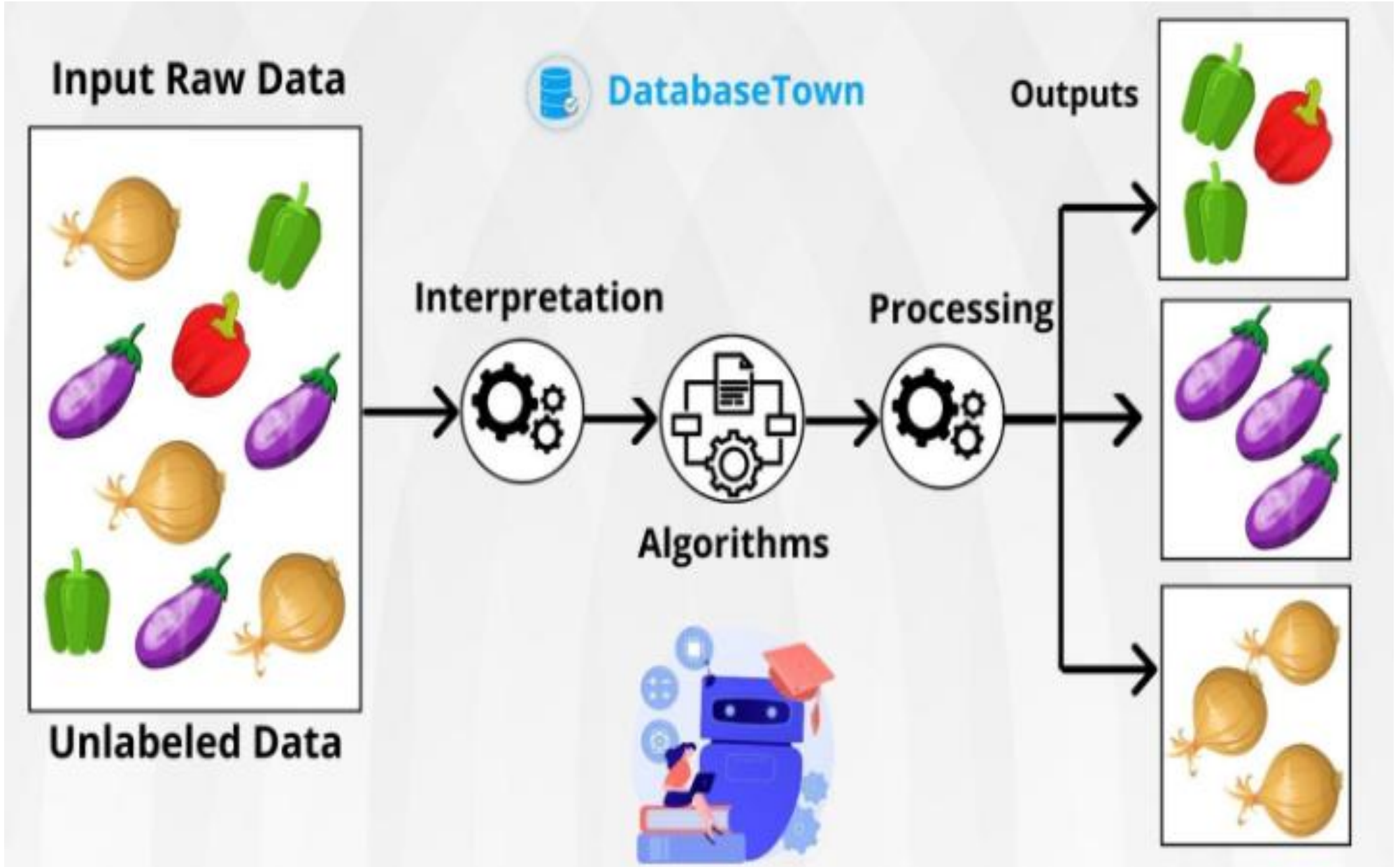
- Regression in machine learning consists of mathematical methods that allow data scientists to predict a continuous outcome (y) based on the value of one or more predictor variables (x).
- It helps in establishing a relationship among the variables by estimating how one variable affects the other.
- It is mainly used for prediction, forecasting, time series modeling, and determining the causal-effect relationship between variables.
- *"Regression shows a line or curve that passes through all the datapoints on target-predictor graph in such a way that the vertical distance between the datapoints and the regression line is minimum."*


Types of Regression
1 Linear Regression
2 Polynomial Regression
3 Support Vector Regression
4 Decision tree Regression
5 Random Forest Regression
6 Ridge Regression
7 Lasso Regression
8 Logistic Regression

Unsupervised Machine Learning

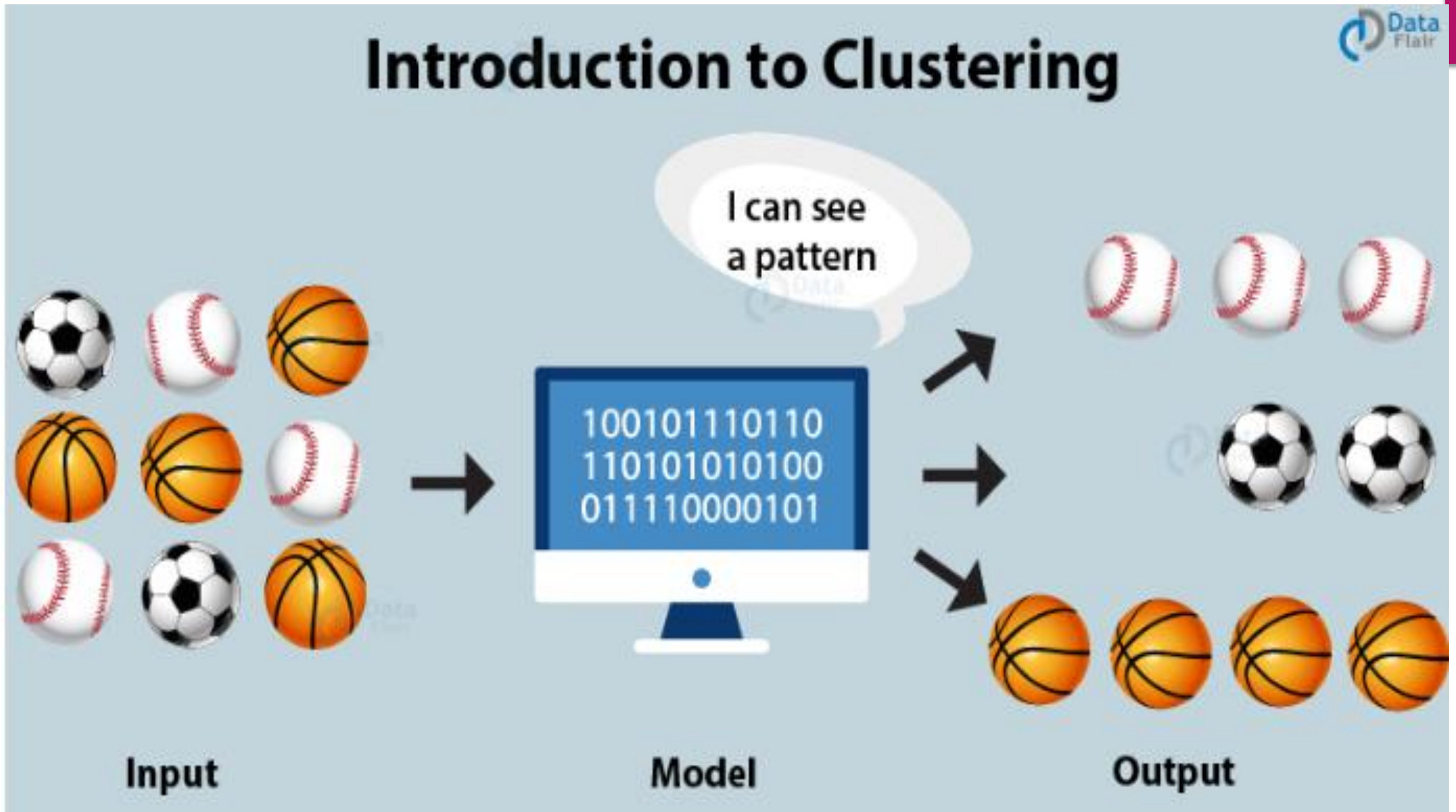
- Unsupervised learning is the case where we fit the model without known outputs. Our goal does not involve to predict any labels here.
- Rather we expect our model to enlighten us by finding unseen patterns within the data set. This might be in the way of grouping the data in various ways to our pleasure.

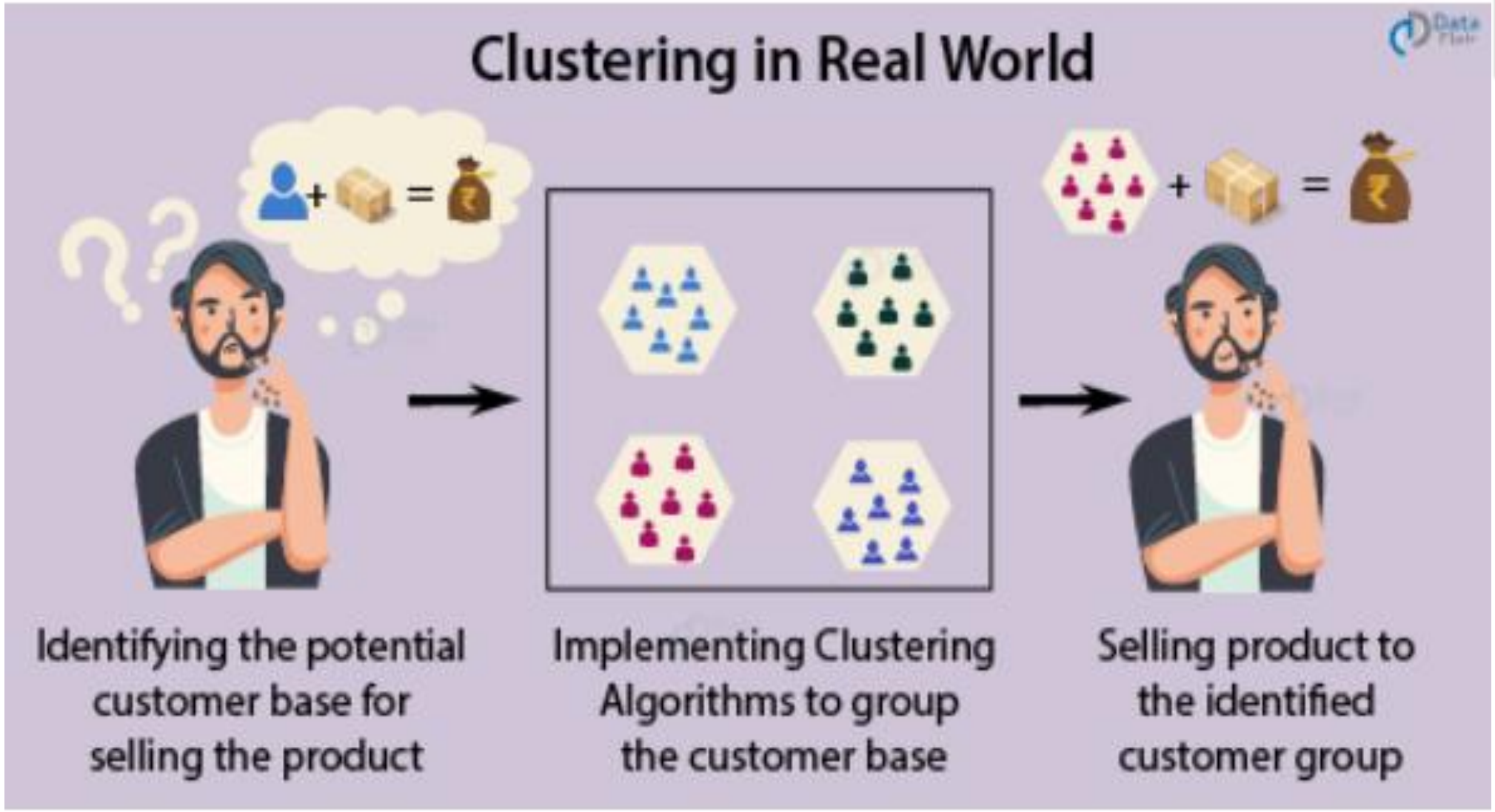



Input Raw Data
DatabaseTown
Outputs
Interpretation
Processing
Algorithms
Unlabeled Data

Types of Unsupervised Learning Techniques

- **Clustering** where data is grouped in a meaningful way
- **Dimensionality Reduction** where high-dimensional data is represented with low-dimensional data
- **Association** where the relationships between variables in a big dataset is discovered


Introduction to Clustering
Data Flair
I can see a pattern
100101110110
110101010100
011110000101
Input
Model
Output


Clustering in Real World
Identifying the potential customer base for selling the product
Implementing Clustering Algorithms to group the customer base
Selling product to the identified customer group

Types of Clustering Algorithms

- In total, there are five distinct types of clustering algorithms. They are as follows –
- Partitioning Based Clustering
- Hierarchical Clustering
- Model-Based Clustering
- Density-Based Clustering
- Fuzzy Clustering

Partitioning Clustering

- In this type of clustering, the algorithm subdivides the data into a subset of k groups. These k groups or clusters are pre-defined. It divides the data into clusters by satisfying these two requirements – Firstly, Each group should consist of at least one point. Secondly, each point must belong to exactly one group. K-Means Clustering is the most popular type of partitioning clustering method.

Hierarchical Clustering

- The basic notion behind this type of clustering is to create a hierarchy of clusters. As opposed to Partitioning Clustering, it does not require pre-definition of clusters upon which the model is to be built. There are two ways to perform Hierarchical Clustering. The first approach is a bottom-up approach, also known as Agglomerative Approach and the second approach is the Divisive Approach which moves hierarchy of clusters in a top-down approach. As a result of this type of clustering, we obtain a tree-like representation known as a dendogram.

Density-Based Models

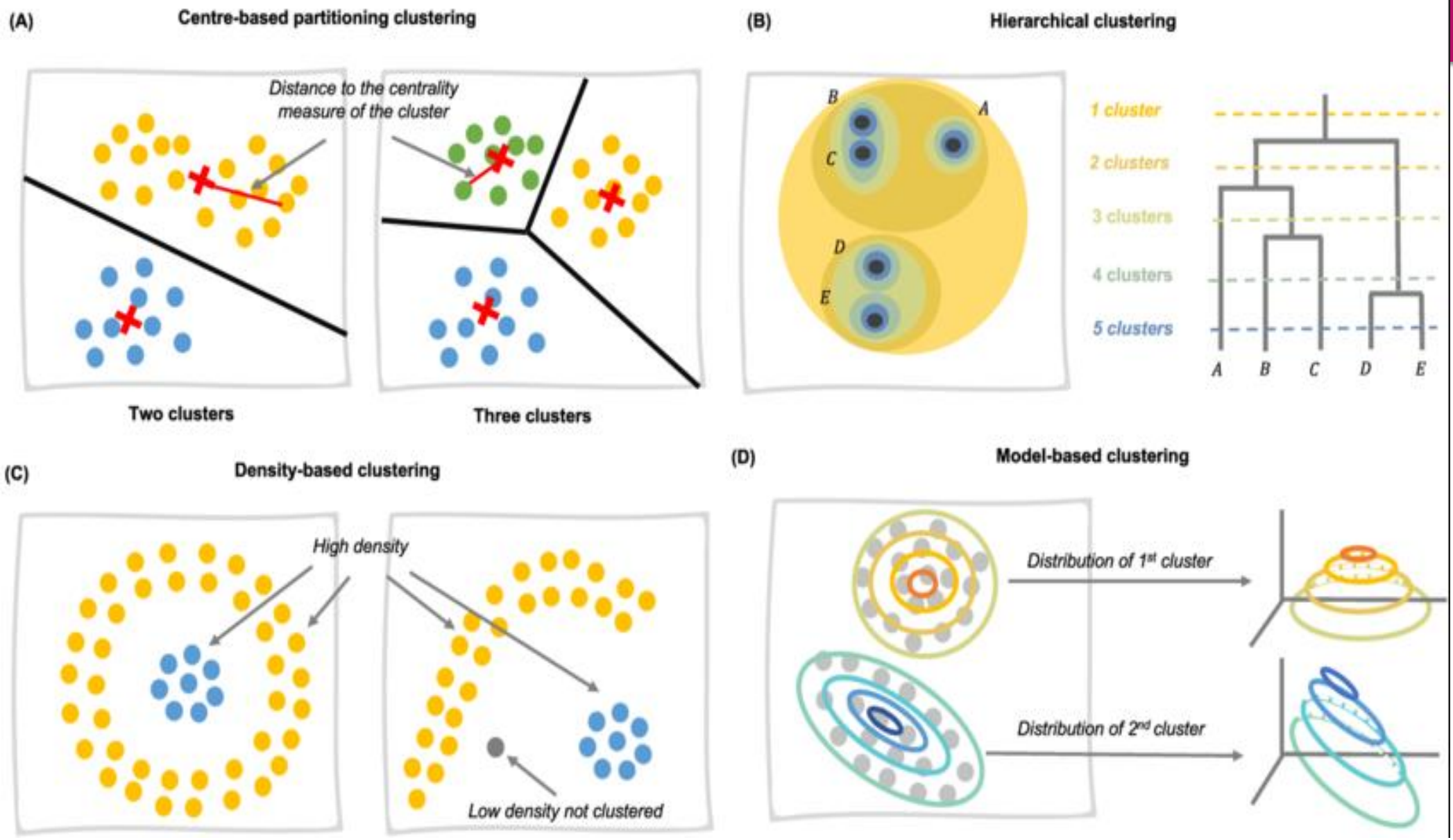
- In these type of clusters, there are dense areas present in the data space that are separated from each other by sparser areas. These type of clustering algorithms play a crucial role in evaluating and finding non-linear shape structures based on density. The most popular density-based algorithm is DBSCAn which allows spatial clustering of data with noise. It makes use of two concepts – Data Reachability and Data Connectivity.

Model-Based Clustering

- In this type of clustering technique, the data observed arises from a distribution consisting of a mixture of two or more cluster components. Furthermore, each component cluster has a density function having an associated probability or weight in this mixture.

Fuzzy Clustering

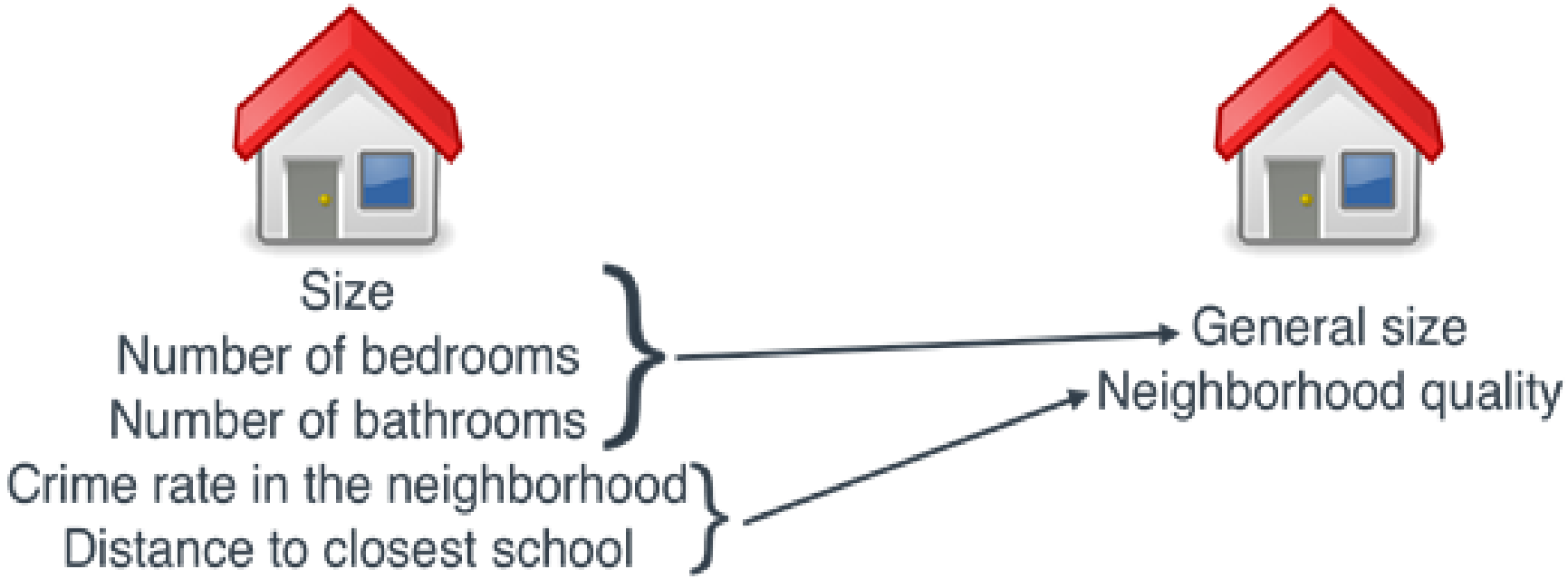
- In this type of clustering, the data points can belong to more than one cluster. Each component present in the cluster has a membership coefficient that corresponds to a degree of being present in that cluster. Fuzzy Clustering method is a soft method of clustering.


(A)
Centre-based partitioning clustering
Distance to the centrality measure of the cluster
Two clusters
Three clusters
(B)
Hierarchical clustering
B
A
C
D
E
1 cluster
2 clusters
3 clusters
4 clusters
5 clusters
A
B
C
D
E
(C)
Density-based clustering
High density
Low density not clustered
(D)
Model-based clustering
Distribution of 1st cluster
Distribution of 2nd cluster

Dimensionality Reduction

- Dimensionality reduction technique tries to construct low-dimensional version of a high-dimensional data.
- This is an important technique since dealing with high-dimensional data is troublesome for both humans to interpret models and machine learning algorithms to learn patterns.
- This technique often used in preprocessing data stage, where complex input data is reduced to its most useful information carrying parts.

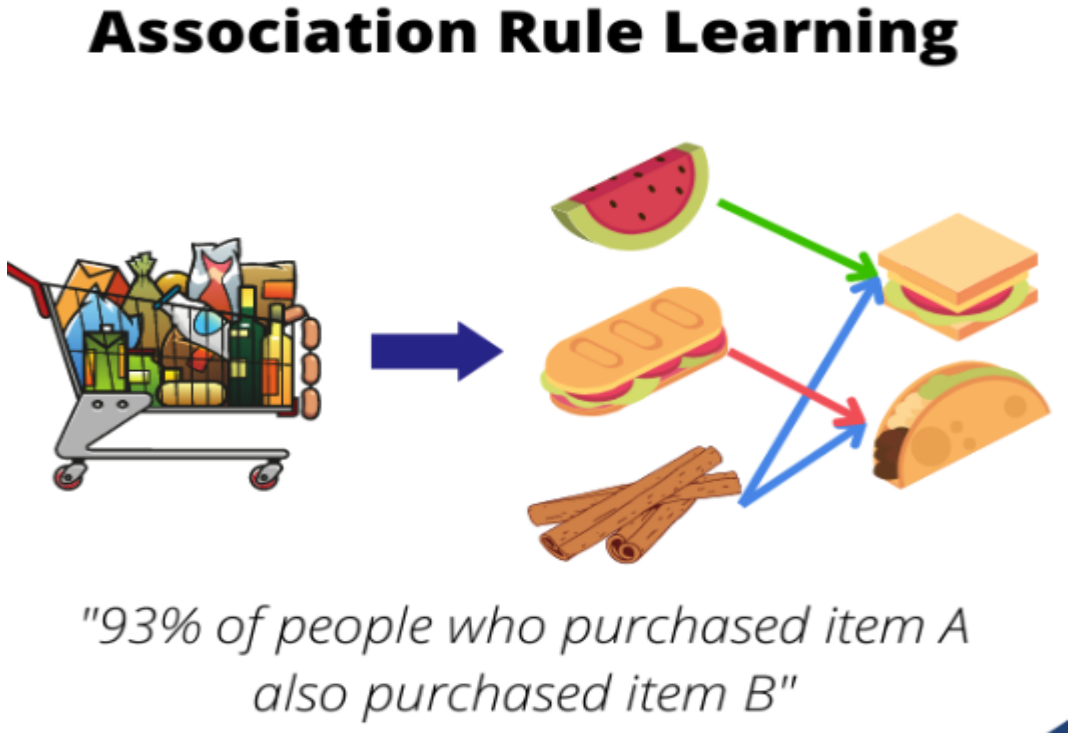
Dimensionality reduction



Using dimensionality reduction to reduce the number of features in a housing dataset, without losing much information.

Association

- Association is about finding relationships between variables in large datasets. I think an example will make it a lot clearer.

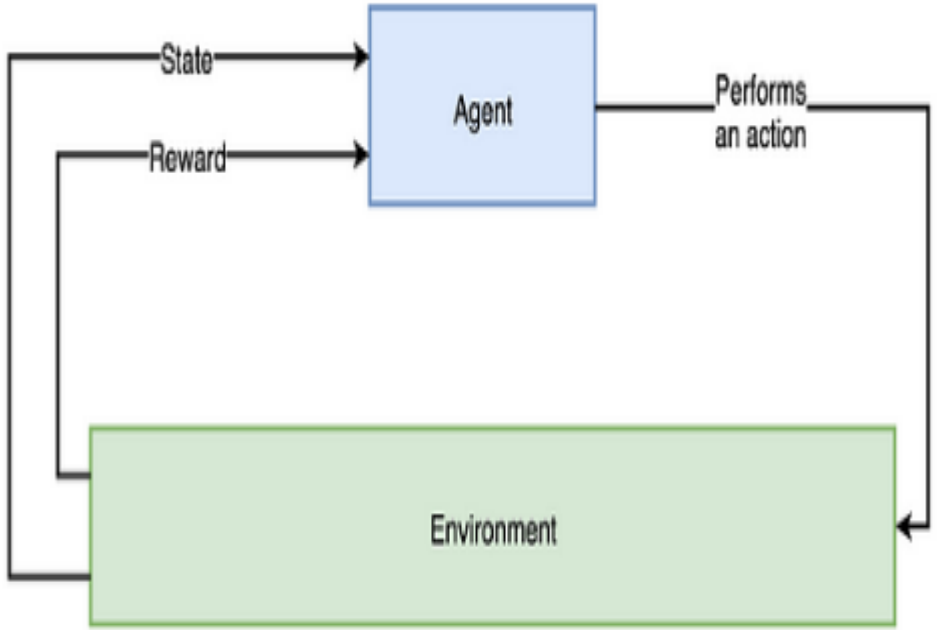


Reinforcement Learning

- Branch of Machine Learning algorithms which produces so-called **agents**. The agent role is slightly different than classic model. It's to **receive information from the environment and react to it** by performing an **action**.
- The information is fed to an agent in form of numerical data, called **state,** which is stored and then used for choosing right action. As a result, an agent receives a **reward** that can be either positive or negative. The reward is a feedback that can be used by an agent to update its parameters.
- Training of an agent is a process of **trial and error**. It needs to find itself in various situations and get punished every time it takes the wrong action in order to learn.

Continued….

- Reinforcement learning is a machine learning model that focuses on how the agents learn to interact with an environment to maximize cumulative rewards.
- Unlike supervised learning, where the agents learn from labeled examples, or in case of unsupervised learning which finds patterns in unlabeled data, reinforcement learning relies on trial and error learning through interactions with the environment.

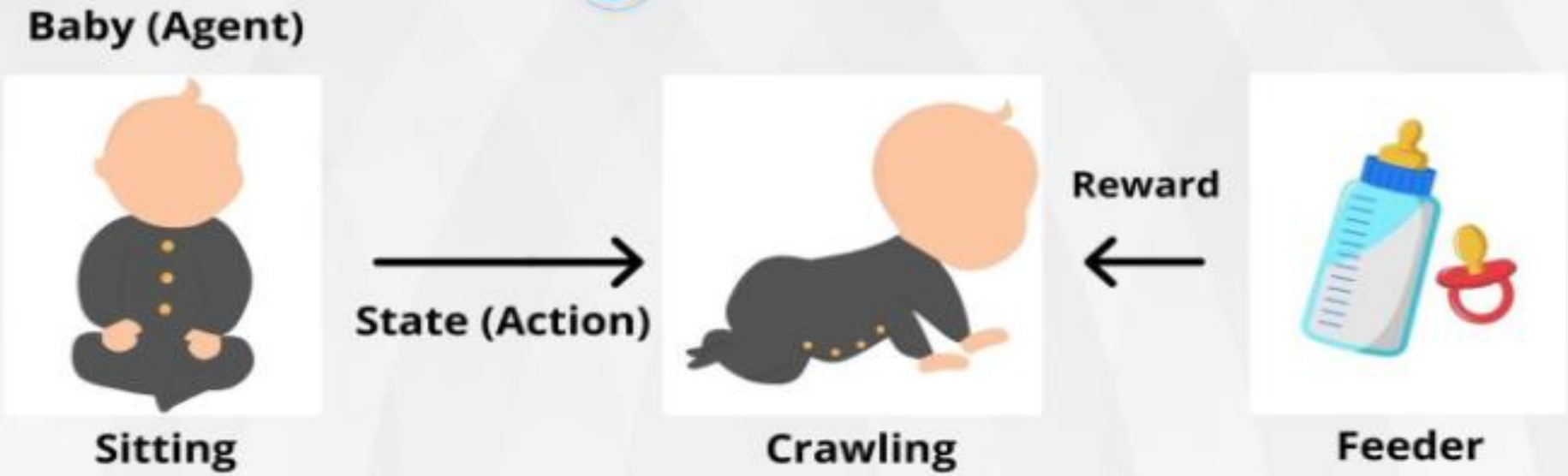


Interaction between Agent and Environment.

REINFORCEMENT LEARNING

Reinforcement learning is a machine learning paradigm that focuses on how agents learn to interact with an environment to maximize cumulative rewards.

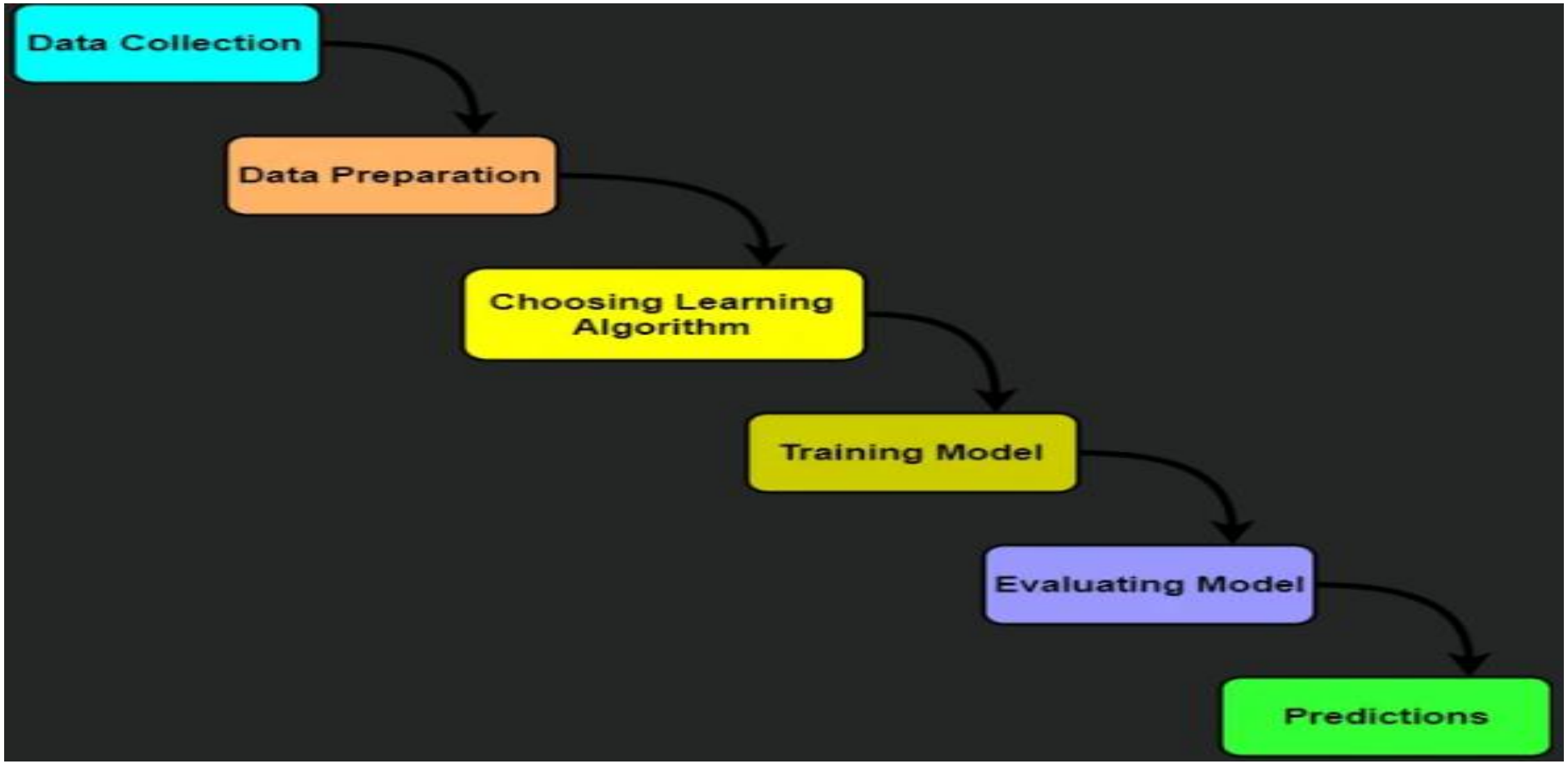
DatabaseTown

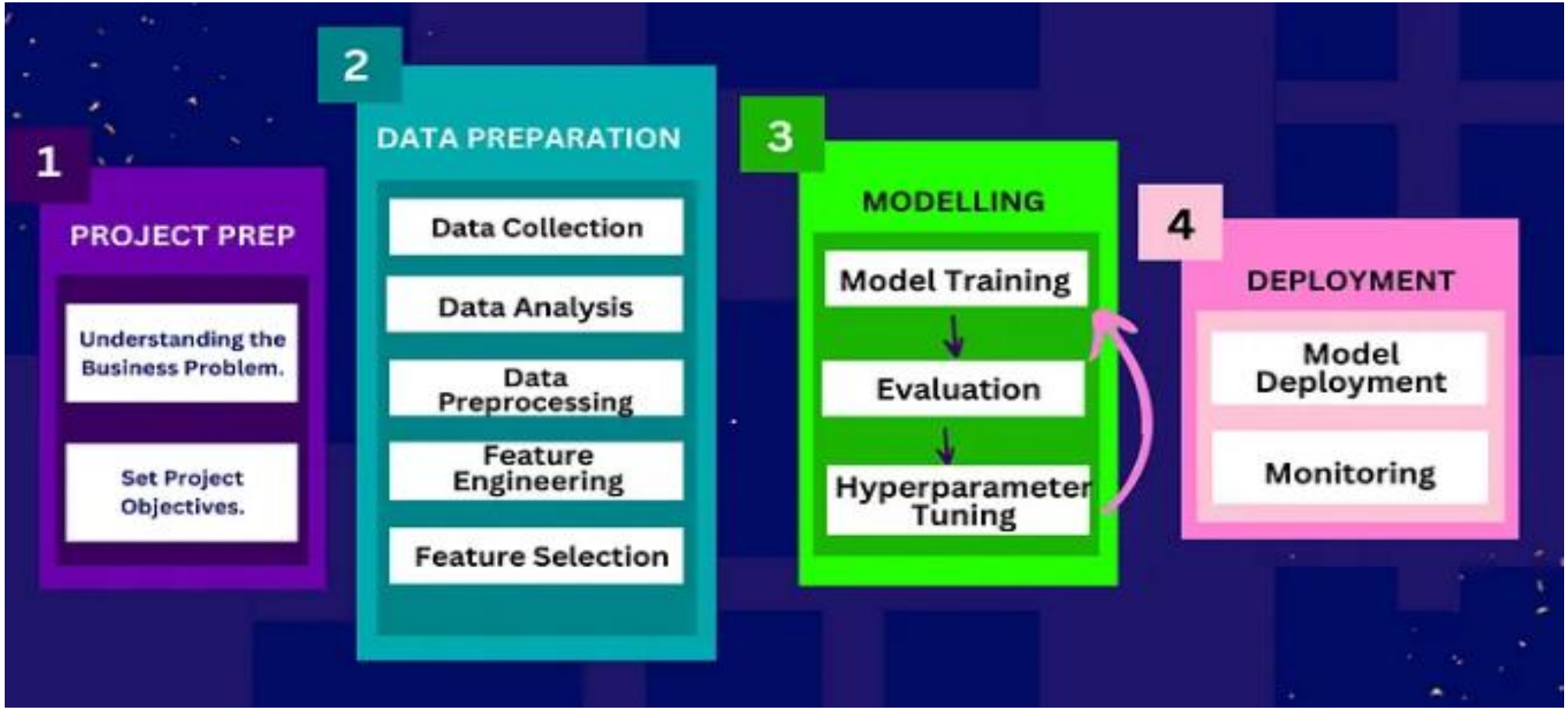


Algorithms and Approaches in Reinforcement Learning

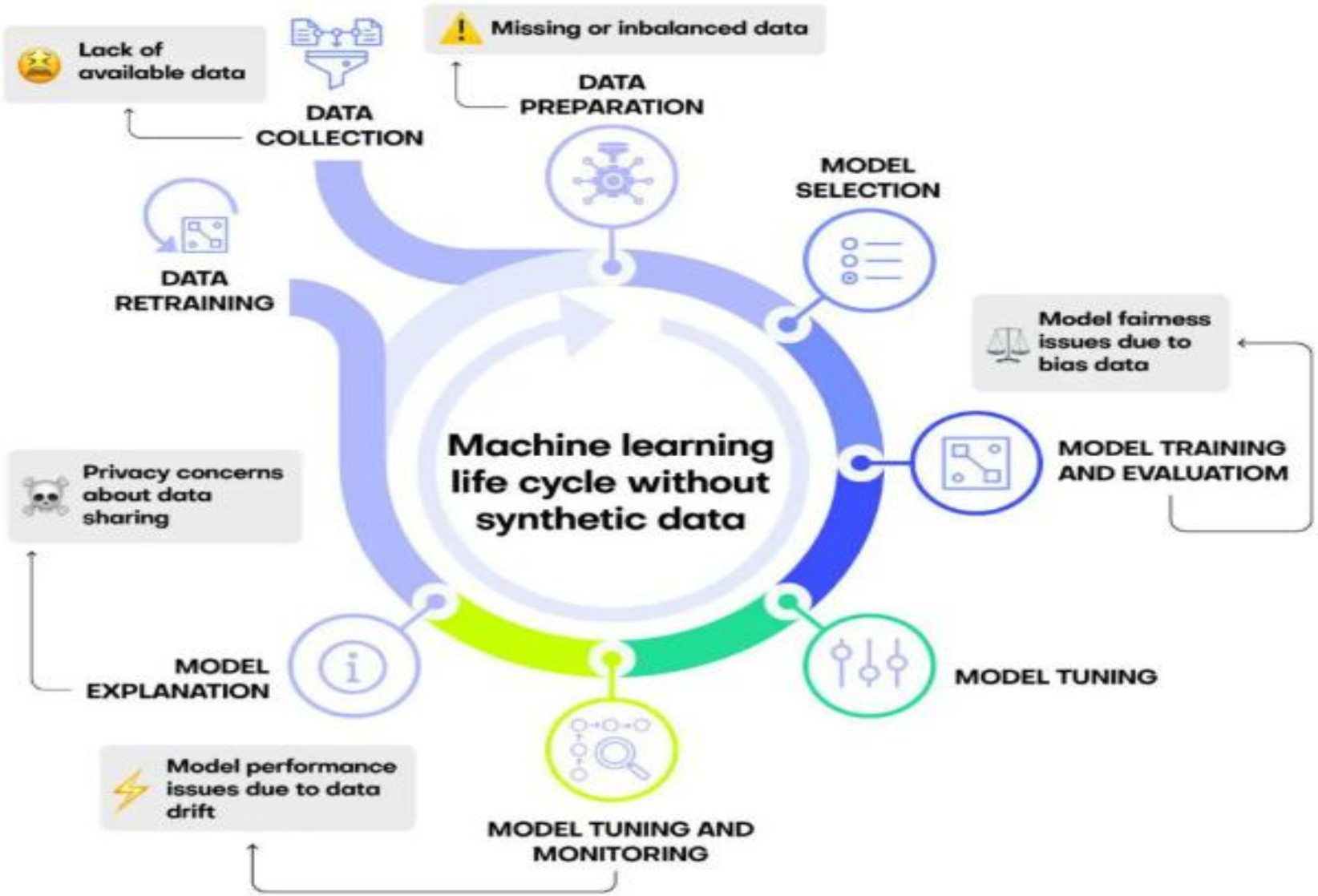
- Q-learning
- Deep Q-networks (DQN)
- Policy Gradients Methods
- Proximal Policy Optimization (PPO)

ML Workflow


Data Collection
Data Preparation
Choosing Learning Algorithm
Training Model
Evaluating Model
Predictions



Machine Learning Workflow Summary


Lack of available data
DATA COLLECTION
Missing or inbalanced data
DATA PREPARATION
MODEL SELECTION
DATA RETRAINING
Model fairness issues due to bias data
MODEL TRAINING AND EVALUATIOM
Machine learning life cycle without synthetic data
Privacy concerns about data sharing
MODEL EXPLANATION
MODEL TUNING
Model performance issues due to data drift
MODEL TUNING AND MONITORING

Data Preprocessing

- Data preprocessing is the process of evaluating, filtering, manipulating, and encoding data so that a machine learning algorithm can understand it and use the resulting output.
- The major goal of data preprocessing is to eliminate data issues such as missing values, improve data quality, and make the data useful for machine learning purposes.
- Data in the actual world is quite messy – full of errors, noise, partial information, and missing values. It's also inconsistent, often compiled from many sources using data mining and warehousing techniques.
- In machine learning, the general rule is that the more data you have, the better machine learning models you can train. However, the data needs to be of high quality.
- This is why data preprocessing makes up a significant portion of the daily jobs of data practitioners, who dedicate around 80% of their time to data preprocessing and management.

Data Preprocessing Steps in Machine Learning

- As the adage goes, “If garbage goes in, garbage comes out.”
- Acquire the Dataset
- Import Libraries
- Import Datasets
- Check for Missing Values
- Encode the Data
- Scaling
- Split Dataset Into Training, Evaluation and Validation Sets

Data Preprocessing Best Practices

1. Data Cleaning

The goal here is to identify the simplest solution to correct quality concerns, such as removing incorrect data, filling in missing data, or ensuring the raw data is appropriate for feature engineering.

2. Data Reduction

Raw data collections often contain duplicate data resulting from diverse methods of defining events, as well as material that just doesn’t work for your machine learning architecture or project scope.

Data reduction techniques, such as principal component analysis, are used to convert raw data into a simplified format that is appropriate for certain use cases.

3. Data Transformation

Data scientists consider how different components of the data should be structured to achieve the best results. This might entail arranging unstructured data, merging salient variables where it makes sense, or determining which ranges to focus on.

Continued…

4. Data Enrichment

In this stage, data practitioners use various feature engineering libraries on the data to achieve the needed changes. The end result should be a data set arranged in such a way that it strikes the best balance between training time for a new model and compute requirements.

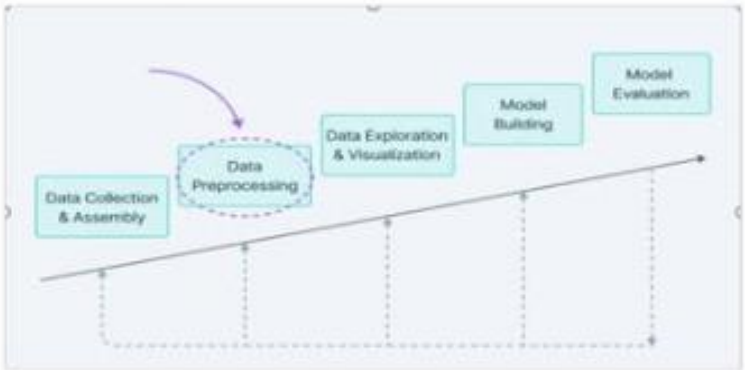
5. Data Validation

Data validation starts with separating data into two sets. The first set is used to train a machine learning or deep learning algorithm. The second one serves as the test data, which is used to assess the correctness and robustness of the final model. This second stage helps to identify any issues with the hypothesis used in data cleaning and feature engineering.

Data Preprocessing - Cheatsheet

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

Data Preprocessing includes the steps we need to follow to transform or encode data so that it may be easily parsed by the machine. The main agenda for a model to be accurate and precise in predictions is that the algorithm should be able to easily interpret the data's features.



Why do we need Data Preprocessing?

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model. It involves below steps:

- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scaling

1) Importing Libraries

In order to perform data preprocessing using Python, we need to import some predefined Python libraries. These libraries are used to perform some specific jobs.

There are many libraries like Numy, Matplotlib, Pandas etc. that we will use for data preprocessing, which are:

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

2) Preparing Dataset

To create a machine learning model, the first thing we required is a dataset as a machine learning model completely works on data. The collected data for a particular problem in a proper format is known as the **dataset**. To use the dataset in our code, we usually put it into a CSV **file**. However, sometimes, we may also need to use an HTML or xlsx file.

CSV stands for "**Comma-Separated Values**" files; it is a file format which allows us to save the tabular data, such as spreadsheets. It is useful for huge datasets and can use these datasets in programs. We can download datasets online from various sources such as:

- https://www.superdatascience.com/pages/machine-learning,
- https://www.kaggle.com/uciml/datasets,
- https://archive.ics.uci.edu/ml/index.php etc.

Now we need to import the datasets which we have collected for our machine learning project. To import the dataset here an example dataset (Dataset.csv), we will use read_csv() function of pandas library, which is used to read a csv file and performs various operations on it. Using this function, we can read a csv file locally as well as through an URL.

We can use read_csv function as below:

```
In [6]: data_set= pd.read_csv('Dataset.csv')
        data_set
```

Out[6]:

	Country	Age	Salary	Purchased
0	india	34.0	92000.0	Yes
1	Sri lanka	22.0	26000.0	Yes
2	China	31.0	74000.0	Yes
3	Sri lanka	29.0	NaN	No
4	China	55.0	98000.0	Yes
5	India	24.0	30000.0	No

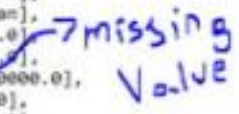
Here, **data_set** is a name of the variable to store our dataset, and inside the function, we have passed the name of our dataset.

Extracting dependent and independent variables: In machine learning, it is important to distinguish the matrix of features (independent variables) and dependent variables from dataset.

Extracting independent variable: To extract an independent variable, we will use **iloc[ ]** method of Pandas library. It is used to extract the required rows and columns from the dataset.

```
In [17]: X = data_set.iloc[:,:-1].values
         X

Out[17]: array([['India', 34.0, 92000.0],
                ['Sri lanka', 22.0, 25000.0],
                ['China', 31.0, 74000.0],
                ['Sri lanka', 29.0, nan],
                ['China', 55.0, 98000.0],
                ['India', 24.0, 30000.0],
                ['Sri lanka', 28.0, 40000.0],
                ['India', nan, 60000.0],
                ['China', 51.0, 89000.0],
                ['India', 44.0, 78000.0],
                ['Sri lanka', 21.0, 20000.0],
                ['China', 25.0, 30000.0],
                ['India', 33.0, 45000.0],
                ['India', 42.0, 65000.0],
                ['Sri lanka', 33.0, 22000.0]], dtype=object)
```



In the above code, the first colon(:) is used to take all the rows, and the second colon(:) is for all the columns. Here we have used :-1, because we don't want to take the last column as it contains the dependent variable. So by doing this, we will get the matrix of features.

Extracting dependent variable:

To extract dependent variables, again, we will use Pandas .iloc[] method.

```
In [9]: y= data_set.iloc[:,3].values
        y

Out[9]: array(['Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'No', 'No', 'Yes', 'Yes',
               'No', 'Yes', 'Yes', 'Yes', 'No'], dtype=object)
```

Here all the rows with the last column only. It will give the array of dependent variables.

3) Handling Missing data:

Data Cleaning

Data Cleaning is particularly done as part of data preprocessing to clean the data by filling missing values, smoothing the noisy data, resolving the inconsistency, and removing outliers.

Missing values

Here are a few ways to solve this issue:

- **By deleting the particular row:** The first way is used to commonly deal with null values. In this way, we just delete the specific row or column which consists of null values. But this way is not so efficient and removing data may lead to loss of information which will not give the accurate output.
- **Fill in the missing values:** There are many methods to achieve this, such as filling in the values manually, predicting the missing values using regression method, or numerical methods like attribute mean.
    - **By calculating the mean:** In this way, we will calculate the mean of that column or row which contains any missing value and will put it on the place of missing value. This strategy is useful for the features which have numeric data such as age, salary, year, etc. Here, we will use this approach.

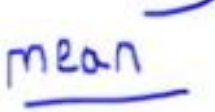
```
In [28]: age_mean= data_set.iloc[:,1]
         age_mean.mean()

Out[28]: 33.714285714285715
```

To handle missing values, we will use **Scikit-learn** library in our code, which contains various libraries for building machine learning models. Here we will use **Imputer** class of **sklearn.preprocessing** library. Below is the code for it:

```
In [20]: X

Out[20]: array([['India', 34.0, 92000.0],
                ['Sri lanka', 22.0, 25000.0],
                ['China', 31.0, 74000.0],
                ['Sri lanka', 29.0, 54857.142857142855],
                ['China', 55.0, 98000.0],
                ['India', 24.0, 30000.0],
                ['Sri lanka', 28.0, 40000.0],
                ['India', 33.714285714285715, 60000.0],
                ['China', 51.0, 89000.0],
                ['India', 44.0, 78000.0],
                ['Sri lanka', 21.0, 20000.0],
                ['China', 25.0, 30000.0],
                ['India', 33.0, 45000.0],
                ['India', 42.0, 65000.0],
                ['Sri lanka', 33.0, 22000.0]], dtype=object)
```



It involves removing a random error or variance in a measured variable. It can be done with the help of the following techniques:

- **Binning**: It is the technique that works on sorted data values to smoothen any noise present in it. The data is divided into equal-sized bins, and each bin/bucket is dealt with independently. All data in a segment can be replaced by its mean, median or boundary values.
- **Regression**: This data mining technique is generally used for prediction. It helps to smoothen noise by fitting all the data points in a regression function. The linear regression equation is used if there is only one independent attribute; else Polynomial equations are used.
- **Clustering**: Creation of groups/clusters from data having similar values. The values that don't lie in the cluster can be treated as noisy data and can be removed.

Removing outliers

Clustering techniques group together similar data points. The tuples that lie outside the cluster are outliers/inconsistent data.

4) Encoding Categorical data:

Categorical data is data which has some categories such as, in our dataset; there are two categorical variable, **Country**, and **Purchased**. Since ML model completely works on mathematics and numbers, but if our dataset would have a categorical variable, then it may create trouble while building the model. So it is necessary to encode these categorical variables into numbers.

For Country variable:

Firstly, we will convert the country variables into categorical data. So to do this, we will use **LabelEncoder()** class from **preprocessing** library.

```
In [32]: # Catgorical data   for Country Variable
         from sklearn.preprocessing import LabelEncoder
         label_encoder_x= LabelEncoder()
         X[:, 0]= label_encoder_x.fit_transform(X[:, 0])
         X[:, 0]

Out[32]: array([3, 2, 0, 2, 0, 1, 2, 1, 0, 1, 2, 0, 1, 1, 2], dtype=object)
```

In above code, we have imported **LabelEncoder** class of **sklearn library**. This class has successfully encoded the variables into digits.

But in our case, there are three country variables, and as we can see in the above output, these variables are encoded into 0, 1, and 2. By these values, the machine learning model may assume that there is some correlation between these variables which will produce the wrong output. So to remove this issue, we will use **dummy encoding**.

Dummy Variables:

Dummy variables are those variables which have values 0 or 1. The 1 value gives the presence of that variable in a particular column, and rest variables become 0. With dummy encoding, we will have a number of columns equal to the number of categories.

In our dataset, we have 3 categories so it will produce three columns having 0 and 1 values. For Dummy Encoding, we will use **OneHotEncoder** class of **preprocessing** library.

```
In [10]: #for Country Variable
         from sklearn.preprocessing import LabelEncoder, OneHotEncoder
         label_encoder_x= LabelEncoder()
         X[:, 0]= label_encoder_x.fit_transform(X[:, 0])

         #Encoding for dummy variables
         onehot_encoder= OneHotEncoder(categories='auto', sparse=False)
         X= onehot_encoder.fit_transform(X)
         X

Out[10]: array([[0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
                [0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
                [1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.],
```

Now all the variables are encoded into numbers 0 and 1 and divided into three columns.

For Purchased Variable:

```
In [21]: labelencoder_y= LabelEncoder()
         y_encoded=labelencoder_y.fit_transform(y)
         y_encoded

Out[21]: array([1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0], dtype=int64)
```

For the second categorical variable, we will only use labelencoder object of **LableEncoder** class. Here we are not using **OneHotEncoder** class because the purchased variable has only two categories yes or no, and which are automatically encoded into 0 and 1.

5) Splitting the Dataset into the Training set and Test set

In machine learning data preprocessing, we divide our dataset into a training set and test set. This is one of the crucial steps of data preprocessing as by doing this, we can enhance the performance of our machine learning model.

Suppose, if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models.

If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So we always try to make a machine learning model which performs well with the training set and also with the test dataset. Here, we can define these datasets as:

Training Set: A subset of dataset to train the machine learning model, and we already know the output.

Test set: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

For splitting the dataset, we will use the below lines of code:

```
In [ ]: from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test= train_test_split(X, y, test_size= 0.2, random_state=0)
```

- In the above code, the first line is used for splitting arrays of the dataset into random train and test subsets.
- In the second line, we have used four variables for our output that are
  - **x_train:** features for the training data
  - **x_test:** features for testing data
  - **y_train:** Dependent variables for training data
  - **y_test:** Independent variable for testing data

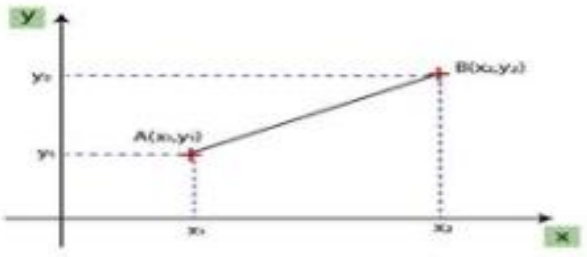
- In **train_test_split() function**, we have passed four parameters in which first two are for arrays of data, and **test_size** is for specifying the size of the test set. The test_size maybe .5, .3, or .2, which tells the dividing ratio of training and testing sets.
- The last parameter **random_state** is used to set a seed for a random generator so that you always get the same result, and the most used value for this is 42.

6) Feature Scaling

Feature scaling is the final step of data preprocessing in machine learning. It is a technique to standardize the independent variables of the dataset in a specific range. In feature scaling, we put our variables in the same range and in the same scale so that no any variable dominate the other variable.

In the example, age and salary column values are not on the same scale. A machine learning model is based on **Euclidean distance**, and if we do not scale the variable, then it will cause some issue in our machine learning model.

Euclidean distance is given as:

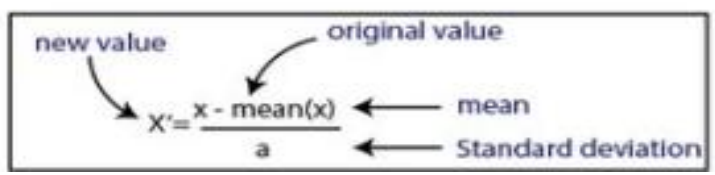


Euclidean Distance Between A and B= $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$

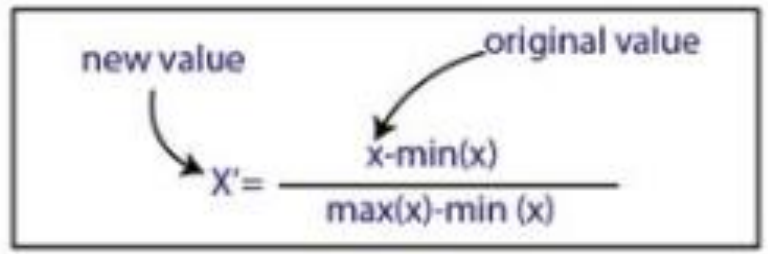
If we compute any two values from age and salary, then salary values will dominate the age values, and it will produce an incorrect result. So to remove this issue, we need to perform feature scaling for machine learning.

There are two ways to perform feature scaling in machine learning:

Standardization



Normalization



Here, we will use the **standardization method** for our dataset.

For feature scaling, we will import ***StandardScaler*** class of ***sklearn.preprocessing*** library as:

Now, we will create the object of **StandardScaler** class for independent variables or features. And then we will fit and transform the training and test dataset.

```
In [23]: # Feature scaling
         from sklearn.preprocessing import StandardScaler
         st_x= StandardScaler()
         X= st_x.fit_transform(X)

         X
```

```
Out[23]: array([[-0.60302269, -0.70710678, -0.70710678,  3.74165739, -0.26726124,
                 -0.26726124, -0.26726124, -0.26726124, -0.26726124,
                 -0.26726124, -0.39223227, -0.26726124,  3.74165739, -0.26726124,
                 -0.26726124, -0.26726124, -0.26726124, -0.26726124, -0.26726124,
```

As we can see in the above output, all the variables are scaled.

Note: Here, we have not scaled the dependent variable because there are only two values 0 and 1. But if these variables will have more range of values, then we will also need to scale those variables.

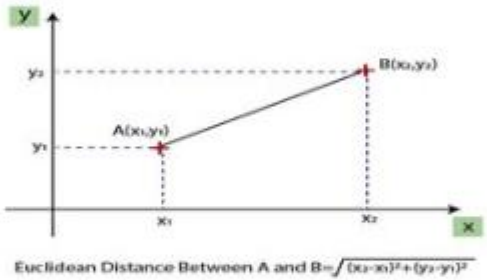
Feature Scaling in Machine Learning

(ALL-IN-ONE CHEAT SHEET)

Feature scaling is the final step of data preprocessing in machine learning. It is a technique to standardize the independent variables of the dataset in a specific range. In feature scaling, we put our variables in the same range and in the same scale so that no any variable dominate the other variable.

In the example, age and salary column values are not on the same scale. A machine learning model is based on **Euclidean distance**, and if we do not scale the variable, then it will cause some issue in our machine learning model.

Euclidean distance is given as:

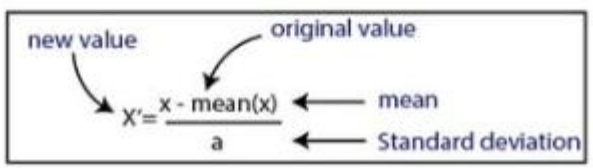


Euclidean Distance Between A and B = $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$

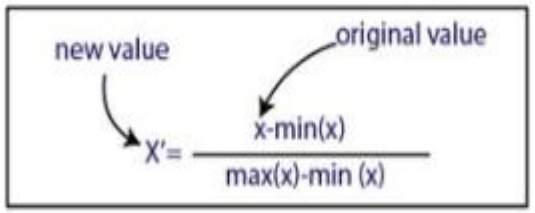
If we compute any two values from age and salary, then salary values will dominate the age values, and it will produce an incorrect result. So to remove this issue, we need to perform feature scaling for machine learning.

There are two ways to perform feature scaling in machine learning:

Standardization



Normalization



Here, we will use the **standardization method** for our dataset.

For feature scaling, we will import ***StandardScaler*** class of ***sklearn.preprocessing*** library as:

Now, we will create the object of **StandardScaler** class for independent variables or features. And then we will fit and transform the training and test dataset.

```
In [23]: # Feature scaling
         from sklearn.preprocessing import StandardScaler
         st_x= StandardScaler()
         X= st_x.fit_transform(X)

         X
```

```
Out[23]: array([[-0.68382269, -0.70710678, -0.70710678,  3.74165739, -0.26726124,
                 -0.26726124, -0.26726124, -0.26726124, -0.26726124, -0.26726124,
                 -0.26726124, -0.39223227, -0.26726124,  3.74165739, -0.26726124,
                 -0.26726124, -0.26726124, -0.26726124, -0.26726124, -0.26726124,
```

As we can see in the above output, all the variables are scaled.

Note: Here, we have not scaled the dependent variable because there are only two values 0 and 1. But if these variables will have more range of values, then we will also need to scale those variables.

Data Normalization in Machine Learning

(ALL-IN-ONE CHEAT SHEET)

Data normalization is a preprocessing step applied to data before it is fed into a machine learning algorithm. The goal of normalization is to transform the data into a standard scale without distorting differences in the ranges of values.

Why Normalize Data?

1. **Improves Model Performance**: Many machine learning algorithms perform better or converge faster when features are on a relatively similar scale and/or close to normally distributed.
2. **Prevents Dominance of Certain Features**: Features with larger ranges can dominate the distance calculations and the learning process.
3. **Reduces Training Time**: Normalized data can lead to faster convergence during the training phase.

Common Normalization Techniques

1. **Min-Max Scaling (Rescaling)**
   - Formula: X′=X−XminXmax−XminX' = \frac{X - X_{min}}{X_{max} - X_{min}}X′=Xmax−XminX−Xmin
   - This scales the data to a fixed range, usually 0 to 1.
   - Sensitive to outliers as they will be compressed in the range.
2. **Z-Score Normalization (Standardization)**
   - Formula: X′=X−μσX' = \frac{X - \mu}{\sigma}X′=σX−μ
   - Here, μ\muμ is the mean and σ\sigmaσ is the standard deviation of the feature.
   - This technique centers the data around 0 with a standard deviation of 1.
3. **Max Abs Scaling**
   - Formula: X′=X∣Xmax∣X' = \frac{X}{|X_{max}|}X′=∣Xmax∣X
   - Scales the data by the maximum absolute value of the feature.
   - Useful when data contains positive and negative values.
4. **Robust Scaling**
   - Formula: X′=X−medianIQRX' = \frac{X - \text{median}}{IQR}X′=IQRX−median
   - Here, IQR is the Interquartile Range (75th percentile - 25th percentile).
   - Useful for data with many outliers.

When to Use Normalization?

- **Algorithms that rely on distance metrics**: K-nearest neighbors, K-means clustering, and support vector machines.
- **Gradient-based algorithms**: Neural networks, logistic regression, and linear regression.
- **PCA and other dimension reduction techniques**: Normalizing data ensures that the principal components are not biased towards features with larger magnitudes.

Practical Implementation

Using libraries like Scikit-Learn in Python, you can easily apply normalization techniques:

```
python
Copy code
from sklearn.preprocessing import MinMaxScaler, StandardScaler, RobustScaler

# Example data
data = [[-1, 2], [-0.5, 6], [0, 10], [1, 18]]

# Min-Max Scaling
scaler = MinMaxScaler()
data_minmax = scaler.fit_transform(data)

# Z-Score Normalization
scaler = StandardScaler()
data_standardized = scaler.fit_transform(data)

# Robust Scaling
scaler = RobustScaler()
data_robust = scaler.fit_transform(data)

print("Min-Max Scaled Data:\n", data_minmax)
print("Standard Scaled Data:\n", data_standardized)
print("Robust Scaled Data:\n", data_robust)
```

Summary

Data normalization is a crucial step in the data preprocessing pipeline, especially for machine learning algorithms that are sensitive to the scale of the data. By transforming the features into a common scale, we can improve model performance, reduce training time, and ensure that no single feature disproportionately affects the learning process.

DIFFERENCE BETWEEN

Normalization and Standardization

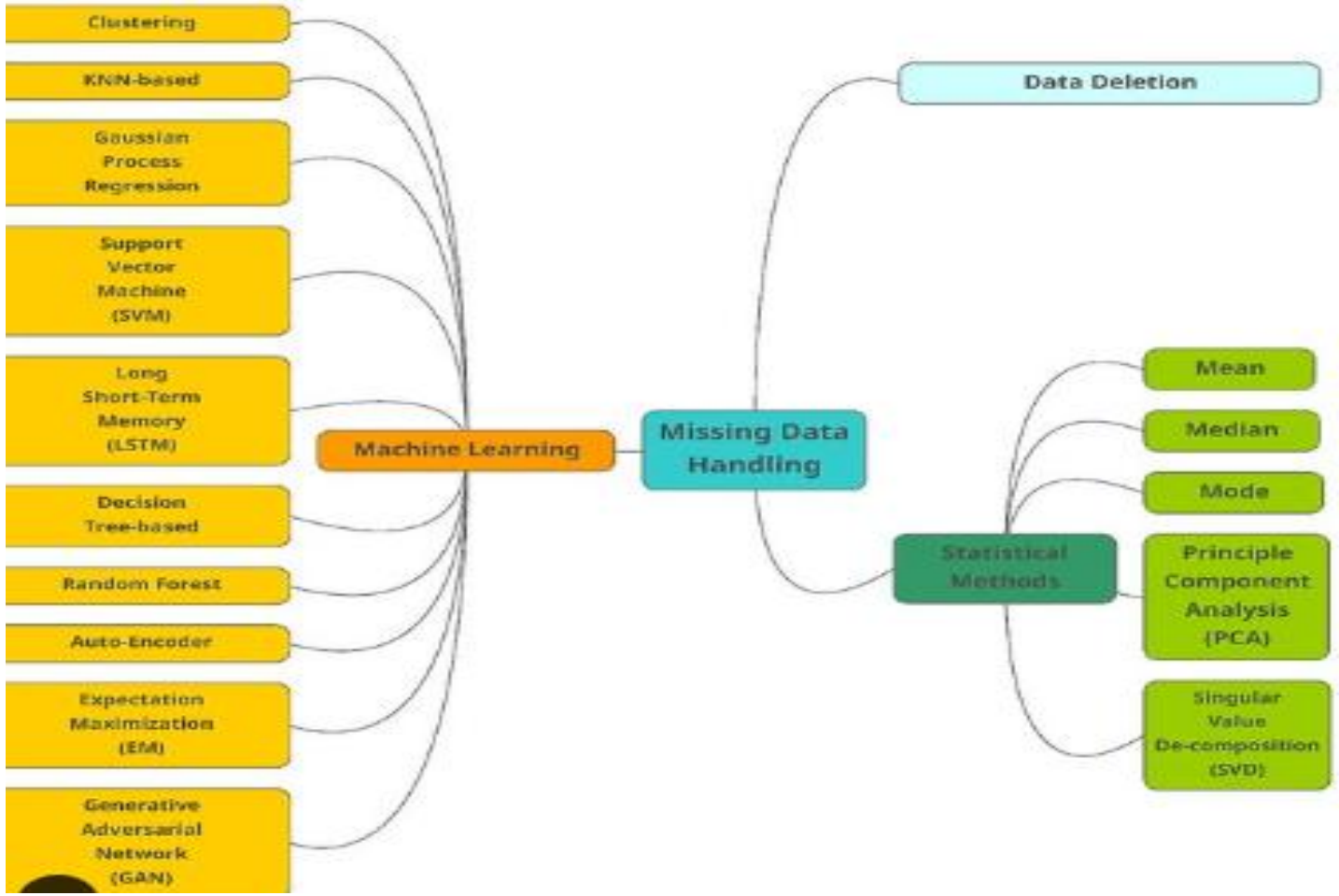
S.NO.	Normalization	Standardization
1.	Minimum and maximum value of features are used for scaling	Mean and standard deviation is used for scaling.
2.	It is used when features are of different scales.	It is used when we want to ensure zero mean and unit standard deviation.
3.	Scales values between [0, 1] or [-1, 1].	It is not bounded to a certain range.
4.	It is really affected by outliers.	It is much less affected by outliers.
5.	Scikit-Learn provides a transformer called `MinMaxScaler` for Normalization.	Scikit-Learn provides a transformer called `StandardScaler` for standardization.
6.	This transformation squishes the n-dimensional data into an n-dimensional unit hypercube.	It translates the data to the mean vector of original data to the origin and squishes or expands.
7.	It is useful when we don't know about the distribution	It is useful when the feature distribution is Normal or Gaussian.
8.	It is a often called as Scaling Normalization	It is a often called as Z-Score Normalization.

Feature Encoding Techniques

Method	Definition	Pros	Cons
One-hot encoding	replace the categorical variable by different boolean variables (0/1) to indicate whether or not certain label is true for that observation	keep all information of that variable	1. expand feature space dramatically if too many labels in that variable 2. does not add additional value to make the variable more predictive
Ordinal-encoding	replace the labels by some ordinal number if ordinal is meaningful	straightforward	does not add additional value to make the variable more predictive
Count/frequency encoding	replace each label of the categorical variable by the count/frequency within that category	/	1. may yield same encoding for two different labels (if they appear same times) and lose valuable info. 2. may not add predictive power
Mean encoding	replace the label by the mean of the target for that label. (the target must be 0/1 valued or continuous)	1. Capture information within the label, therefore rendering more predictive features 2. Create a monotonic relationship between the variable and the target 3. Do not expand the feature space	Prone to cause over-fitting

Top Techniques to Handle Missing Values

(Every data scientist should know)



Feature Selection - Cheatsheet

Definition: Feature Selection is the process of selecting a subset of relevant features for use in machine learning model building.

It is not always the truth that the more data, the better the result will be. Including irrelevant features (the ones that are just unhelpful to the prediction) and redundant features (irrelevant in the presence of others) will only make the learning process overwhelmed and easy to cause overfitting.

With feature selection, we can have:

- simplification of models to make them easier to interpret
- shorter training times and lesser computational cost
- lesser cost in data collection
- avoid the curse of dimensionality
- enhanced generalization by reducing overfitting

We should keep in mind that different feature subsets render optimal performance for different algorithms. So it's not a separate process along with the machine learning model training. Therefore, if we are selecting features for a linear model, it is better to use selection procedures targeted to those models, like importance by regression coefficient or Lasso. And if we are selecting features for trees, it is better to u... tree derived importance.

1. Filter Method:

Filter methods select features based on a performance measure regardless of the ML algorithm later employed.

Univariate filters evaluate and rank a single feature according to a certain criteria, while multivariate filters evaluate the entire feature space. Filter methods are:

- selecting variable regardless of the model
- less computationally expensive
- usually give lower prediction performance

As a result, filter methods are suited for a first step quick screen and removal of irrelevant features.

Method	Definition
Variance	removing features that show the same value for the majority/all of the observations (constant/quasi-constant features)
Correlation	remove features that are highly correlated with each other
Chi-Square	Compute chi-squared stats between each non-negative feature and class
Mutual Information Filter	Mutual information measures how much information the presence/absence of a feature contributes to making the correct prediction on Y.
Univariate ROC-AUC or MSE	builds one decision tree per feature, to predict the target, then make predictions and ranks the features according to the machine learning metric (roc-auc or mse)
Information Value (IV)	a byproduct of WOE. IV = Σ(Proportion of Good Outcomes - Proportion of Bad Outcomes) * WOE

...encoding (see section 3.3.2) and IV often go hand in hand in scorecard development. The two conc... ...n derived from logistic regression and is kind of standard practice in credit card industry. IV is a popul... and widely used measure as there are very convenient rules of thumb for variables selection associated with IV as below:

Information Value (IV)	Predictive Power
< 0.02	useless for prediction
0.02 to 0.1	weak predictor
0.1 to 0.3	medium predictor
0.3 to 0.5	strong predictor
> 0.5	suspicious or too good to be true

However, all these filtering methods fail to consider the interaction between features and may reduce our predict power. Personally I only use variance and correlation to filter some absolutely unnecessary features.

2. Wrapper Method:

Wrappers use a search strategy to search through the space of possible feature subsets and evaluate each subset by the quality of the performance on a ML algorithm. Practically any combination of search strategy and algorithm can be used as a wrapper. It is featured as:

- use ML models to score the feature subset
- train a new model on each subset
- very computationally expensive
- usually provide the best performing subset for a give ML algorithm, but probably not for another
- need an arbitrary defined stopping criteria

The most common **search strategy** group is Sequential search, including Forward Selection, Backward Elimination and Exhaustive Search. Randomized search is another popular choice, including Evolutionary computation algorithms such as genetic, and Simulated annealing.

Another key element in wrappers is **stopping criteria**. When to stop the search? In general there're three:

- performance increase
- performance decrease
- predefined number of features is reached

2.1 Forward Selection:

Step forward feature selection starts by evaluating all features individually and selects the one that generates the best performing algorithm, according to a pre-set evaluation criteria. In the second step, it evaluates all possible combinations of the selected feature and a second feature, and selects the pair that produce the best performing algorithm based on the same pre-set criteria.

The pre-set criteria can be the roc_auc for classification and the r squared for regression for example.

This selection procedure is called greedy, because it evaluates all possible single, double, triple and so on feature combinations. Therefore, it is quite computationally expensive, and sometimes, if feature space is big, even unfeasible.

There is a special package for python that implements this type of feature selection: mlxtend.

2.2 Backward Elimination:

Step backward feature selection starts by fitting a model using all features. Then it removes one feature. It will remove the one that produces the highest performing algorithm (least statistically significant) for a certain evaluation criteria. In the second step, it will remove a second feature, the one that again produces the best performing algorithm. And it proceeds, removing feature after feature, until a certain criteria is met.

The pre-set criteria can be the roc_auc for classification and the r squared for regression for example.

2.3 Exhaustive Feature Selection:

In an exhaustive feature selection the best subset of features is selected, over all possible feature subsets, by optimizing a specified performance metric for a certain machine learning algorithm. For example, if the classifier is a logistic regression and the dataset consists of **4** features, the algorithm will evaluate all **15** feature combinations as follows:

- all possible combinations of 1 feature
- all possible combinations of 2 features
- all possible combinations of 3 features
- all the 4 features

and select the one that results in the best performance (e.g., classification accuracy) of the logistic regression classifier.

This exhaustive search is very computationally expensive. In practice for this computational cost, it is rarely used.

3. Embedded Method:

Embedded Method combine the advantages of the filter and wrapper methods. A learning algorithm takes advantage of its own variable selection process and performs feature selection and classification at same time. Common embedded methods include Lasso and various types of tree-based algorithms. It is featured as:

- perform feature selection as part of the model building process
- consider interactions between features
- less computationally expensive as it only train the model once, compared to Wrappers
- usually provide the best performing subset for a give ML algorithm, but probably not for another

3.1 Regularization with Lasso:

Regularization consists in adding a penalty to the different parameters of the machine learning model to reduce the freedom of the model. Hence, the model will be less likely to fit the noise of the training data so less likely to be overfitting.

In linear model regularization, the penalty is applied over the coefficients that multiply each of the predictors. For linear models there are in general 3 types of regularization:

- L1 regularization (Lasso)
- L2 regularization (Ridge)
- L1/L2 (Elastic net)

From the different types of regularization, **Lasso (L1)** has the property that is able to shrink some of the coefficients to zero. Therefore, that feature can be removed from the model.

Both for linear and logistic regression we can use the Lasso regularization to remove non-important features. Keep in mind that increasing the penalization will increase the number of features removed. Therefore, you will need to keep an eye and monitor that you don't set a penalty too high so that to remove even important features, or too low and then not remove non-important features.

Having said this, if the penalty is too high and important features are removed, you should notice a drop in the performance of the algorithm and then realize that you need to decrease the regularization.

Regularization is a large topic. For for information you can refer to here:

3.2 Random Forest Importance:

Random forests are one of the most popular machine learning algorithms. They are so successful because they provide in general a good predictive performance, low overfitting and easy interpretability. This interpretability is given by the fact that it is straightforward to derive the importance of each variable on the tree decision. In other words, it is easy to compute how much each variable is contributing to the decision.

Random forest is a bagging algorithm consists a bunch of base estimators (decision trees), each of them built over a random extraction of the observations from the dataset and a random extraction of the features. Not every tree sees all the features or all the observations, and this guarantees that the trees are **de-correlated** and therefore **less prone to over-fitting.**

Each tree is also a sequence of yes-no questions based on a single or combination of features. At each split, the question divides the dataset into 2 buckets, each of them hosting observations that are more similar among themselves and different from the ones in the other bucket. Therefore, the importance of each feature is derived by how "**pure**" each of the buckets is.

For classification, the measure of impurity is either the **Gini impurity** or the **information gain/entropy**. For regression the measure of impurity is **variance**. Therefore, when training a tree, it is possible to compute how much each feature decreases the impurity. The more a feature decreases the impurity, the more important the feature is. In random forests, the impurity decrease from each feature can be averaged across trees to determine the final importance of the variable.

Selecting features by using tree derived feature importance is a very straightforward, fast and generally accurate way of selecting good features for machine learning. In particular, if you are going to build tree methods.

However, correlated features will show in a tree similar and lowered importance, compared to what their importance would be if the tree was built without correlated counterparts.

Limitation

- correlated features show similar importance
- correlated features importance is lower than real importance, when tree is build without its correlated counterparts
- high carinal variable tend to show higher importance

3.3 Gradient Boosted Trees Importance:

Similarly to selecting features using Random Forests derived feature importance, we can select features based on the importance derived by gradient boosted trees. And we can do that in one go, or in a recursive manner, depending on how much time we have, how many features are in the dataset, and whether they are correlated or not.

4. Feature Shuffling:

A popular method of feature selection consists in random shuffling the values of a specific variable and determining how that permutation affects the performance metric of the machine learning algorithm. In other words, the idea is to permute the values of each feature, one at the time, and measure how much the permutation decreases the accuracy, or the roc_auc, or the mse of the machine learning model. If the variables are important, this is, highly predictive, a random permutation of their values will decrease dramatically any of these metrics. Contrarily, non-important / non-predictive variables, should have little to no effect on the model performance metric we are assessing.

5. Hybrid Method:

5.1 Recursive Feature Elimination:

This method consists of the following steps:

1. Rank the features according to their importance derived from a machine learning algorithm: it can be tree importance, or LASSO / Ridge, or the linear / logistic regression coefficients.
2. Remove one feature -the least important- and build a machine learning algorithm utilizing the remaining features.
3. Calculate a performance metric of your choice: roc-auc, mse, rmse, accuracy.
4. If the metric decreases by more of an arbitrarily set threshold, then that feature is important and should be kept. Otherwise, we can remove that feature.
5. Repeat steps 2-4 until all features have been removed (and therefore evaluated) and the drop in performance assessed.

The method combines the selection process like wrappers and feature importance derivation from ML models like embedded methods so it's called hybrid.

The difference between this method and the step backwards feature selection lies in that it does not remove all features first in order to determine which one to remove. It removes the least important one, based on the machine learning model derived importance. And then, it makes an assessment as to whether that feature should be removed or not. So it removes each feature only once during selection, whereas step backward feature selection removes all the features at each step of selection.

This method is therefore faster than wrapper methods and generally better than embedded methods. In practice it works extremely well. It does also account for correlations (depending on how stringent you set the arbitrary performance drop threshold). On the downside, the drop in performance assessed to decide whether the feature should be kept or removed, is set arbitrarily. The smaller the drop the more features will be selected, and vice versa.

Example: Recursive Feature Elimination with Random Forests Importance

As we talked about in section 4.3.2, Random Forests assign equal or similar importance to features that are highly correlated. In addition, when features are correlated, the importance assigned is lower than the importance attributed to the feature itself, should the tree be built without the correlated counterparts.

Therefore, instead of eliminating features based on importance **at one time** (from all initial features), we may get a better selection by removing one feature **recursively**, and recalculating the importance on each round.

In this situation, when a feature that is highly correlated to another one is removed, then, the importance of the remaining feature increases. This may lead to a better subset feature space selection. On the downside, building several random forests is quite time consuming, in particular if the dataset contains a high number of features.

5.2 Recursive Feature Addition:

This method consists of the following steps:

1. Rank the features according to their importance derived from a machine learning algorithm: it can be tree importance, or LASSO / Ridge, or the linear / logistic regression coefficients.
2. Build a machine learning model with only 1 feature, the most important one, and calculate the model metric for performance.
3. Add one feature -the most important- and build a machine learning algorithm utilizing the added and any feature from previous rounds.
4. Calculate a performance metric of your choice: roc-auc, mse, rmse, accuracy.
5. If the metric increases by more than an arbitrarily set threshold, then that feature is important and should be kept. Otherwise, we can remove that feature.
6. Repeat steps 2-5 until all features have been removed (and therefore evaluated) and the drop in performance assessed.

The difference between this method and the step forward feature selection is similar. It does not look for all features first in order to determine which one to add, so it's faster than wrappers.

Data Preprocessing

- Data preprocessing is the process of preparing data for machine learning algorithms. It involves cleaning, organizing, and reducing data to improve the accuracy and reliability of the dataset.
- Data preprocessing steps

  ✅ Data cleaning: Detects and fixes errors in the data, such as missing values, duplicates, and outliers

  ✅ Data integration: Combines data from multiple sources to create a comprehensive dataset

  ✅ Data reduction: Reduces the volume of data while retaining the most important patterns and trends

  ✅ Encoding categorical variables: Converts categorical data into numerical values so that machine learning algorithms can use it

  ✅ Feature scaling: Standardizes the range of features in a dataset so that no single feature dominates the model

  ✅ Feature engineering: Creates new features from existing ones to help models capture complex patterns in the data

- Data preprocessing can help machine learning models produce more accurate and reliable results.

What is correlation?

- Correlation is a measure of how two or more variables are related in machine learning.
- Correlation is a way to understand how variables change together, or co-vary. It’s used to explore data, select features, and evaluate models.
- Correlation coefficient
  The correlation coefficient is a number that quantifies the strength and direction of the correlation. It ranges from -1 to 1, with values closer to 1 or -1 representing stronger correlations, and values closer to 0 indicating little connection between the variables.
- Interpretation
  A correlation coefficient close to 1 indicates a strong positive correlation, while a correlation coefficient close to -1 indicates a strong negative correlation. A correlation coefficient close to 0 indicates no linear correlation.

Correlation Continues

- Importance
  Understanding the correlation between features in a dataset can help build more accurate models that make better predictions.
- Limitations
  However, correlation can have limitations, and may not be well suited to capturing complex real-world relationships. For example, correlations that have held in the past may not continue to hold in the future.

Correlation in Machine Learning

A correlation is a statistical measure of the relationship between two variables. The measure is best used in variables that demonstrate a linear relationship between each other. The correlation coefficient is a value that indicates the strength of the relationship between variables. The coefficient can take any values from -1 to 1. The interpretations of the values are:

- **-1:** Perfect negative correlation. The variables tend to move in opposite directions (i.e., when one variable increases, the other variable decreases).
- **0:** No correlation. The variables do not have a relationship with each other.
- **1:** Perfect positive correlation. The variables tend to move in the same direction (i.e., when one variable increases, the other variable also increases).

$$r_{xy} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

- r_{xy} – the correlation coefficient of the linear relationship between the variables x and y
- x_i – the values of the x-variable in a sample
- $\bar{x}$ – the mean of the values of the x-variable
- y_i – the values of the y-variable in a sample
- $\bar{y}$ – the mean of the values of the y-variable

Example:

	S&P 500	Apple
2013	1691.75	68.96
2014	1977.80	100.11
2015	1884.09	109.06
2016	2151.13	112.18
2017	2519.36	154.12

			Step 3		Step 4	Step 5	
	S&P 500	Apple	a	b	a x b	a^2	b^2
2013	1691.75	68.96	- 353.08	- 39.93	14,096.91	124,662.66	1,59
2014	1977.80	100.11	- 67.03	- 8.78	588.22	4,492.48	77
2015	1884.09	109.06	- 160.74	0.17	- 27.97	25,836.07	0.03
2016	2151.13	112.18	106.30	3.29	350.16	11,300.52	10.85
2017	2519.36	154.12	474.53	45.23	21,465.08	225,182.62	2,046.11
Mean	2044.83	108.89	Sums		36,472.40	391,474.35	3,728.10

$$r_{xy} = \frac{36{,}272.40}{\sqrt{391{,}474.35 \times 3{,}728.10}} = 0.95$$

The coefficient indicates that the prices of the S&P 500 and Apple Inc. have a high positive correlation. This means that their respective prices tend to move in the same direction.

Pearson Correlation

Correlation between sets of data is a measure of how well they are related. The most common measure of correlation in stats is the Pearson Correlation. The full name is the Pearson Product Moment Correlation (PPMC). It shows the linear relationship between two sets of data.

$$r = \frac{N \times \sum XY - (\sum X \sum Y)}{\sqrt{[N \sum x^2 - (\sum x)^2][N \sum y^2 - (\sum y)^2]}}$$

r	Pearson Correlation Coefficient
n	Quantity of Information
$\sum X$	Total of the First Variable Value
$\sum Y$	Total of the Second Variable Value
$\sum XY$	Sum of the Product of & Second Value
$\sum X^2$	Sum of the Squares of the First Value
$\sum Y^2$	Sum of the Squares of the Second Value

Example:

Calculate the linear correlation coefficient for the following data. X = 4, 8 ,12, 16 and Y = 5, 10, 15, 20

x	y	x^2	y^2	XY
4	5	16	25	20
8	10	64	100	80
12	15	144	225	180
16	20	256	400	320
Σ x = 40	Σ y =50	480	750	600

$$r(xy) = \frac{(4 \times 600) - (40 \times 50)}{\sqrt{4(480) - 40^2}\sqrt{4(750) - 50^2}} = 1$$

Spearman Rank Correlation

Spearman's measures the strength of a monotonic relationship, your data has to be monotonically related. Basically, this means that if one variable increases (or decreases), the other variable also increases (or decreases).

Spearman's returns a value from -1 to 1, where:

+1 = a perfect positive correlation between ranks

-1 = a perfect negative correlation between ranks

0 = no correlation between ranks.

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$$

Example: The scores for nine students in physics and math are as follows:

Physics: 35, 23, 47, 17, 10, 43, 9, 6, 28

Mathematics: 30, 33, 45, 23, 8, 49, 12, 4, 31

Compute the student's ranks in the two subjects and compute the Spearman rank correlation.

Physics	Rank	Math	Rank
35	3	30	5
23	5	33	3
47	1	45	2
17	6	23	6
10	7	8	8
43	2	49	1
9	8	12	7
6	9	4	9
28	4	31	4

Physics	Rank	Math	Rank	d	d squared
35	3	30	5	2	4
23	5	33	3	2	4
47	1	45	2	1	1
17	6	23	6	0	0
10	7	8	8	1	1
43	2	49	1	1	1
9	8	12	7	1	1
6	9	4	9	0	0
28	4	31	4	0	0

= 1 – (6*12)/(9(81-1)) = 0.9

Covariance in Machine Learning

- Covariance is a statistical term that measures the relationship between two random variables in a data set.
- Covariance measures how the movement of one variable affects the other. It’s a measure of the variance between the two variables.
- Covariance can be calculated by multiplying the correlation between the two variables by the standard deviation of each variable.
- A positive covariance means the variables move in the same direction, while a negative covariance means they move in opposite directions. The greater the covariance value, the more connected the variables are.
- Covariance only analyzes the linear relationship between two variables, while correlation can involve two or more variables. Covariance is also measured in units, while correlation is not.
- Covariance is useful for defining the type of relationship between two variables.

Covariance in Machine Learning

Covariance: A measure of the relationship between random variables

In mathematics and statistics, covariance is a measure of the relationship between two random variables. The metric evaluates how much – to what extent – the variables change together. In other words, it is essentially a measure of the variance between two variables. The variance can take any positive or negative values. The values are interpreted as follows:

- **Positive covariance**: Indicates that two variables tend to move in the same direction.
- **Negative covariance**: Reveals that two variables tend to move in inverse directions.

Population Covariance Formula

$$Cov(x,y) = \frac{\Sigma(x_i-\bar{x})(y_i-\bar{y})}{N}$$

Sample Covariance

$$Cov(x,y) = \frac{\Sigma(x_i-\bar{x})(y_i-y)}{N-1}$$

- x_i = data value of x
- y_i = data value of y
- $\bar{x}$ = mean of x
- $\bar{y}$ = mean of y
- N = number of data values.

Evaluation Metrics in Machine Learning

(For Classification Problems)

True positive (TP): It represents an outcome in which positive samples are accurately predicted as positive by the model.

True negative (TN): It represents an outcome in which negative samples are accurately predicted as negative samples by the model.

False positive (FP): It represents an outcome in which negative samples are incorrectly predicted as positive by the model.

False negative (FN): It represents an outcome in which positive samples are wrongly predicted as negative samples by the model.

Based on these 4 outcomes, a set of model performance metrics are given below.

True Positive Rate (TPR) (also known as Sensitivity): The probability that positive samples will predict positive.

$$True\ Positive\ Rate\ (TPR)\ or\ Sensitivity = \frac{TP}{TP + FN}, \quad TPR \in [0,1]$$

False Positive Rate (FPR): The probability that negative samples will predict positive.

$$False\ Positive\ Rate\ (FPR) = \frac{FP}{FP + TN}, \quad FPR \in [0,1]$$

Precision: It estimates positive sample predictions that are genuinely from the positive class label.

$$Precision = \frac{TP}{TP + FP}, \quad Precision \in [0,1]$$

Recall: It estimates positive sample predictions from all actual positives.

$$Recall = \frac{TP}{TP + FN}, \quad Recall \in [0,1]$$

F-measure: It balances precision and recalls both together to provide a single score.

$$F\text{-}measure = \frac{2 \cdot (Precision \cdot Recall)}{(Precision + Recall)}, \quad F-measure \in [0,1]$$

Accuracy: It gives the total correct predictions (TP+TN) made by the model.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \quad Accuracy \in [0,1]$$

The area under the receiver operating characteristic curve (auROC): It shows whether the model is capable of correctly discriminating between class labels. $auROC \in [0,1]$

Matthews correlation coefficient (MCC): It computes the correlation between actual and predicted labels and is calculated by the following formula.

$$MCC = \frac{(TP \cdot TN) - (FP \cdot FN)}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}}, \quad MCC \in [-1,1]$$

Linear Discriminant Analysis (LDA)

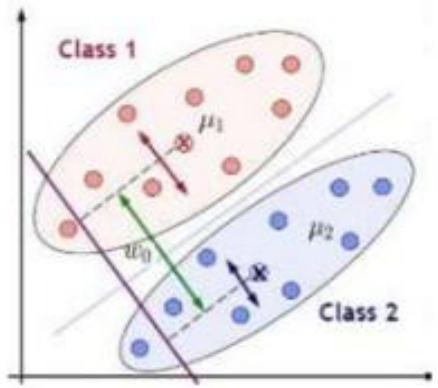
(ALL-IN-ONE CHEAT SHEET)

For **multiclass classification**, LDA is the preferred linear technique.

Representation:

LDA representation consists of **statistical properties** calculated for **each class**: **means** and the **covariance matrix:**

$$\mu_k = \frac{1}{n_k}\sum_{i=1}^{n} x_i \quad \text{and} \quad \sigma^2 = \frac{1}{n-K}\sum_{i=1}^{n}(x_i - \mu_k)^2$$



LDA assumes **Gaussian** data and attributes of **same σ^2**.

Predictions are made using **Bayes Theorem:**

$$P(Y = k|X = x) = \frac{P(k)\times P(x|k)}{\sum_{l=1}^{K} P(l)\times P(x|l)}$$

to obtain a discriminate function (latent variable) for each class k, estimating $P(x|k)$ with a Gaussian distribution:

$$D_k(x) = x \times \frac{\mu_k}{\sigma^2} - \frac{{\mu_k}^2}{2\sigma^2} + \ln(P(k))$$

The class with largest **discriminant value** is the **output class**.

Variations:

- **Quadratic DA**: Each class uses its own variance estimate
- **Regularized DA**: Regularization into the variance estimate

Data preparation:

- Review and modify univariate distributions to be Gaussian
- Standardize data to $\mu = 0, \sigma = 1$ to have same variance
- Remove noise such as outliers

Advantages:

+ Can be used for dimensionality reduction by keeping the latent variables as new variables

Usecase example:

- Prediction of customer churn

Dimensionality Reduction

Dimensionality Reduction

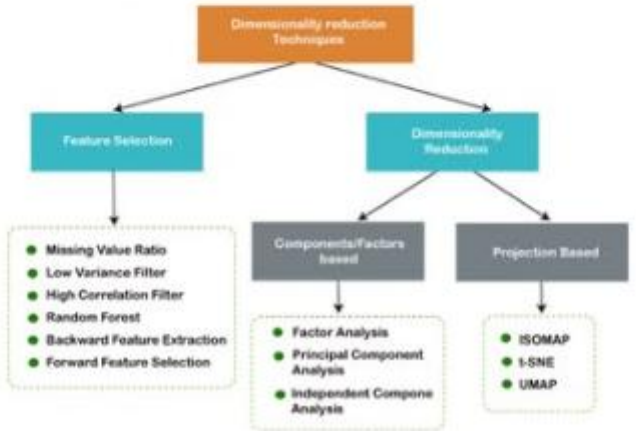
(ALL-IN-ONE CHEAT SHEET)

The number of input features, variables, or columns present in a given dataset is known as dimensionality, and the process to reduce these features is called dimensionality reduction.

A dataset contains a huge number of input features in various cases, which makes the predictive modeling task more complicated. Because it is very difficult to visualize or make predictions for the training dataset with a high number of features, for such cases, dimensionality reduction techniques are required to use.

Dimensionality reduction technique can be defined as, ***"It is a way of converting the higher dimensions dataset into lesser dimensions dataset ensuring that it provides similar information."*** These techniques are widely used in machine learning for obtaining a better fit predictive model while solving the classification and regression problems.

It is commonly used in the fields that deal with high-dimensional data, such as **speech recognition, signal processing, bioinformatics, etc. It can also be used for data visualization, noise reduction, cluster analysis**, etc.



The Curse of Dimensionality:

Handling the high-dimensional data is very difficult in practice, commonly known as the *curse of dimensionality*. If the dimensionality of the input dataset increases, any machine learning algorithm and model becomes more complex. As the number of features increases, the number of samples also gets increased proportionally, and the chance of overfitting also increases. If the machine learning model is trained on high-dimensional data, it becomes overfitted and results in poor performance.

Hence, it is often required to reduce the number of features, which can be done with dimensionality reduction.

Advantages of Dimensionality Reduction:

Some benefits of applying dimensionality reduction technique to the given dataset are given below:

- By reducing the dimensions of the features, the space required to store the dataset also gets reduced.
- Less Computation training time is required for reduced dimensions of features.
- Reduced dimensions of features of the dataset help in visualizing the data quickly.
- It removes the redundant features (if present) by taking care of multicollinearity.

Disadvantages of Dimensionality Reduction:

There are also some disadvantages of applying the dimensionality reduction, which are given below:

- Some data may be lost due to dimensionality reduction.
- In the PCA dimensionality reduction technique, sometimes the principal components required to consider are unknown.

Dimensionality Reduction Techniques

Feature Selection:

Feature selection is the process of selecting the subset of the relevant features and leaving out the irrelevant features present in a dataset to build a model of high accuracy. In other words, it is a way of selecting the optimal features from the input dataset.

Three methods are used for the feature selection:

1. Filters Methods

In this method, the dataset is filtered, and a subset that contains only the relevant features is taken. Some common techniques of filters method are:

- **Correlation**
- **Chi-Square Test**
- **ANOVA**
- **Information Gain, etc.**

2. Wrappers Methods

The wrapper method has the same goal as the filter method, but it takes a machine learning model for its evaluation. In this method, some features are fed to the ML model, and evaluate the performance. The performance decides whether to add those features or remove to increase the accuracy of the model. This method is more accurate than the filtering method but complex to work. Some common techniques of wrapper methods are:

- Forward Selection
- Backward Selection
- Bi-directional Elimination

3. Embedded Methods: Embedded methods check the different training iterations of the machine learning model and evaluate the importance of each feature. Some common techniques of Embedded methods are:

- **LASSO**
- **Elastic Net**
- **Ridge Regression, etc.**

Feature Extraction:

Feature extraction is the process of transforming the space containing many dimensions into space with fewer dimensions. This approach is useful when we want to keep the whole information but use fewer resources while processing the information.

Some common feature extraction techniques are:

a. Principal Component Analysis

b. Linear Discriminant Analysis

c. Kernel PCA

d. Quadratic Discriminant Analysis

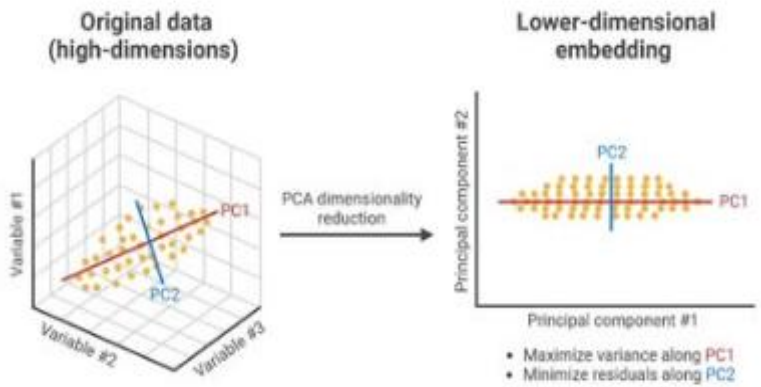
Principal Component Analysis (PCA)

(ALL-IN-ONE CHEAT SHEET)

Principal Component Analysis (PCA) is a technique used in statistics and data science to reduce the dimensionality of a dataset while retaining most of the variability present in the data. It is used to identify patterns in data, to compress it for storage or transmission, and to reduce noise.

PCA works by finding the linear combinations of variables in a dataset that explain the maximum amount of variance in the data. These linear combinations are called principal components, and they are ordered by the amount of variance they explain. The first principal component explains the most variance, the second explains the next most, and so on.

To perform PCA, the data is first standardized to have zero mean and unit variance. Then, the covariance matrix of the standardized data is computed. The principal components are then calculated by finding the eigenvectors and eigenvalues of the covariance matrix. The eigenvectors are the principal components, and the eigenvalues represent the amount of variance explained by each component.



Steps for PCA Algorithm:

1. **Getting the dataset**

   Firstly, we need to take the input dataset and divide it into two subparts X and Y, where X is the training set, and Y is the validation set.

2. **Representing data into a structure**

   Now we will represent our dataset into a structure. Such as we will represent the two-dimensional matrix of independent variable X. Here each row corresponds to the data items, and the column corresponds to the Features. The number of columns is the dimensions of the dataset.

3. **Standardizing the data**

   In this step, we will standardize our dataset. Such as in a particular column, the features with high variance are more important compared to the features with lower variance.

   If the importance of features is independent of the variance of the feature, then we will divide each data item in a column with the standard deviation of the column. Here we will name the matrix as Z.

4. **Calculating the Covariance of Z**

   To calculate the covariance of Z, we will take the matrix Z, and will transpose it. After transpose, we will multiply it by Z. The output matrix will be the Covariance matrix of Z.

5. **Calculating the Eigen Values and Eigen Vectors**

   Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix Z. Eigenvectors or the covariance matrix are the directions of the axes with high information. And the coefficients of these eigenvectors are defined as the eigenvalues.

6. **Sorting the Eigen Vectors**

   In this step, we will take all the eigenvalues and will sort them in decreasing order, which means from largest to smallest. And simultaneously sort the eigenvectors accordingly in matrix P of eigenvalues. The resultant matrix will be named as P*.

7. **Calculating the new features Or Principal Components**

   Here we will calculate the new features. To do this, we will multiply the P* matrix to the Z. In the resultant matrix Z*, each observation is the linear combination of original features. Each column of the Z* matrices are independent of each other.

8. **Remove less or unimportant features from the new dataset.**

   The new feature set has occurred, so we will decide here what to keep and what to remove. It means, we will only keep the relevant or important features in the new dataset, and unimportant features will be removed.

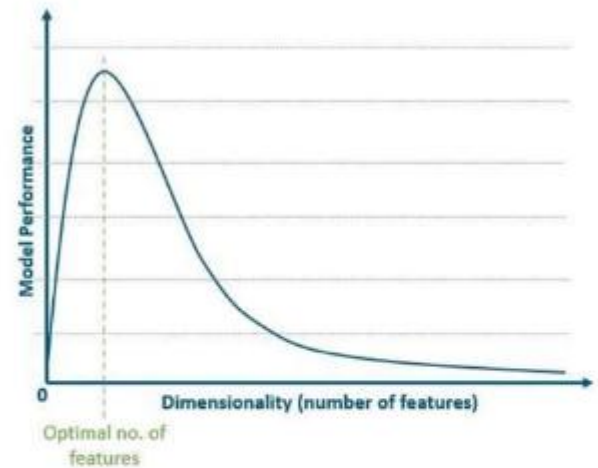
Common terms used in PCA Algorithm:

- **Dimensionality:** It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.
- **Correlation:** It signifies how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.
- **Orthogonal:** It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.
- **Eigenvectors:** If there is a square matrix M, and a non-zero vector v is given. Then v will be an eigenvector if Av is the scalar multiple of v.
- **Covariance Matrix:** A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

What is the Curse of Dimensionality?

It refers to the difficulties a machine learning algorithm faces when working with data in the higher dimensions(features), that did not exist in the lower dimensions.

This means, that you are required to have an `Optimal Set` of features to predict better accuracy of the model.



@data_science_school

With the increase in the data dimensions, your model –

- would also increase in `Computation Cost`
- would decrease the `Performance of the Model`
- would become increasingly dependent on the training data for production.

This leads to overfitting of the model, so even though the model performs really well on training data, it may fails on test data.

Dimensionality Reduction

It is a process of reducing the dimension of your data to a few principal features in the original dataset, to remove the curse of dimensionality.

Outliers in Machine Learning

- An outlier is an observation or data point that significantly differs from other observations in the dataset and also that deviates significantly from the normal pattern or behavior of the data. Various factors, such as measurement errors, unexpected events, data processing errors, etc., can cause these outliers. Measurement errors, data entry errors, or legitimate deviations in the data can cause outliers.
- Outlier analysis is important because it can help identify anomalous data points that can affect the overall analysis and interpretation of the data. By detecting and handling outliers appropriately, data scientists can improve the accuracy and reliability of their results.
- Outlier analysis in data mining involves identifying and analyzing data points significantly different or deviating from the rest of the dataset. Outliers can be caused by various factors, such as data entry errors, unexpected events, etc., and their detection can lead to valuable insights and improve the accuracy of models. A wide range of techniques can be used for outlier analysis in data mining, such as statistical methods, clustering algorithms, and machine learning models. Outlier analysis in data mining is the process of identifying and examining data points that significantly differ from the rest of the dataset.

Benefits of Outlier Analysis in Data Mining:

Outlier analysis in data mining can provide several benefits, as mentioned below:

- **Improved accuracy of data analysis** - Outliers can skew the results of statistical analyses or predictive models, leading to inaccurate or misleading conclusions. Detecting and removing outliers can improve the accuracy and reliability of data analysis.
- **Identification of data quality issues** - Outliers can be caused by data collection, processing, or measurement errors, which can indicate data quality issues. Outlier analysis in data mining can help identify and correct these issues to improve data quality.
- **Detection of unusual events or patterns** - Outliers can represent unusual events or patterns in the data that may be of interest to the businesses. Studying these outliers can provide valuable insights and lead to discoveries.
- **Better decision-making** - Outlier analysis in data mining can help decision-makers identify and understand the factors affecting their data, leading to better-informed decisions.
- **Improved model performance** - Outliers can negatively affect the performance of predictive models. Removing outliers or developing models that can handle them appropriately can improve model performance.

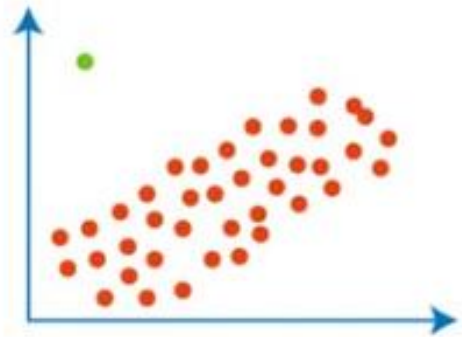
Types of Outliers:

Outliers are of three types, named as:

1. Global (or Point) Outliers
2. Collective Outliers
3. Contextual (or Conditional) Outliers

Global Outliers:

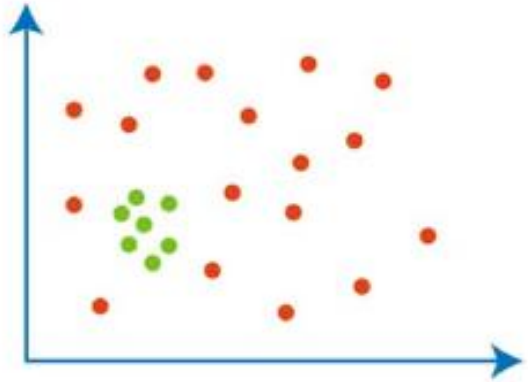
- When a single data object deviates from rest of the data points is known as Global outlier.
- A data point is considered a global outlier if its value is far outside the **entirety** of the data set in which it is found. In the below figure green data point is the global outlier.



Collective Outliers:

- In a given set of data, when a group of data points deviates from the rest of the data set is called collective outliers.
- To identify the types of different outliers, there is a need to go through background information about the relationship between the behavior of outliers shown by different data objects. For example, in an Intrusion Detection System, the DOS package from one system to another is taken as normal behavior. Therefore, if this happens with the various computer

simultaneously, it is considered abnormal behavior, and as a whole, they are called collective outliers. The green data points as a whole represent the collective outlier.



Contextual (Conditional) Outliers

- When a data object deviates from the other data points because of any specific condition in a given data set.
- For example, A temperature reading of 45 degrees Celsius may behave as an outlier in a rainy season but it behaves like a normal data point in the context of a summer season. In the given diagram, a green dot representing the low-temperature value in June is a contextual outlier since the same value in December is not an outlier

Outlier Detection:

The process of detecting outliers and removing them from the data is known as outlier detection. There are basically two types of outlier detection methods:

1. Statistical Distribution-Based Outlier Detection
2. Proximity based Outlier Detection
   - Distance-Based Outlier Detection
   - Density-Based Local Outlier Detection
   - Deviation-Based Outlier Detection

1. Statistical Distribution-Based Outlier Detection:

The statistical distribution-based approach to outlier detection assumes a distribution or probability model for the given data set (e.g., a normal or Poisson distribution) and then identifies outliers with respect to the model using a discordancy test. Application of the test requires knowledge of the data set parameters knowledge of distribution parameters such as the mean and variance and the expected number of outliers.

2. Proximity based Outlier Detection

It is based on location of data points.

i) Distance-Based Outlier Detection:

In comparison with statistical-based methods, distance based outlier detection generalizes the ideas behind discordancy testing for various standard distributions. Distance-based outlier detection avoids the excessive computation that can be associated with fitting the observed distribution into some standard distribution and in selecting discordancy tests.

ii) **Density-Based Local Outlier Detection:**

Statistical and distance-based outlier detection both depend on the overall or global distribution of the given set of data points, D. However, data are usually not uniformly distributed. These methods encounter difficulties when analyzing data with rather different density distributions.

iii) **Deviation-Based Outlier Detection:**

Deviation-based outlier detection does not use statistical tests or distance-based measures to identify exceptional objects. Instead, it identifies outliers by examining the main characteristics of objects in a group. Objects that deviate from this description are considered outliers.

Hence, in this approach the term deviations is typically used to refer to outliers. In this section, we study two techniques for deviation-based outlier detection. The first sequentially compares objects in a set, while the second employs an OLAP data cube approach.

Unit 2
Regression Techniques in Supervised Learning

Linear Regression

- Linear Regression is a statistical method for predicting the value of a continous dependent variable based on one or more independent variables. It estimates the relationship using a linear equation.
- How it works: Take input features
  • Calculate a weighted sum plus a bias term
  • Use the equation ($y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$)
  • Minimize the error (usually Mean Squared Error)
- The Linear Equation:
  This straight line models the relationship between the dependent and independent variables, showing how changes in features affect the target variable.
- When to use it:Predicting continuous outcomes
  • When relationships between features and outcome are linear
  • When you need interpretable results
  • When you have simple to moderately complex data

Continued…..

👍 Pros:

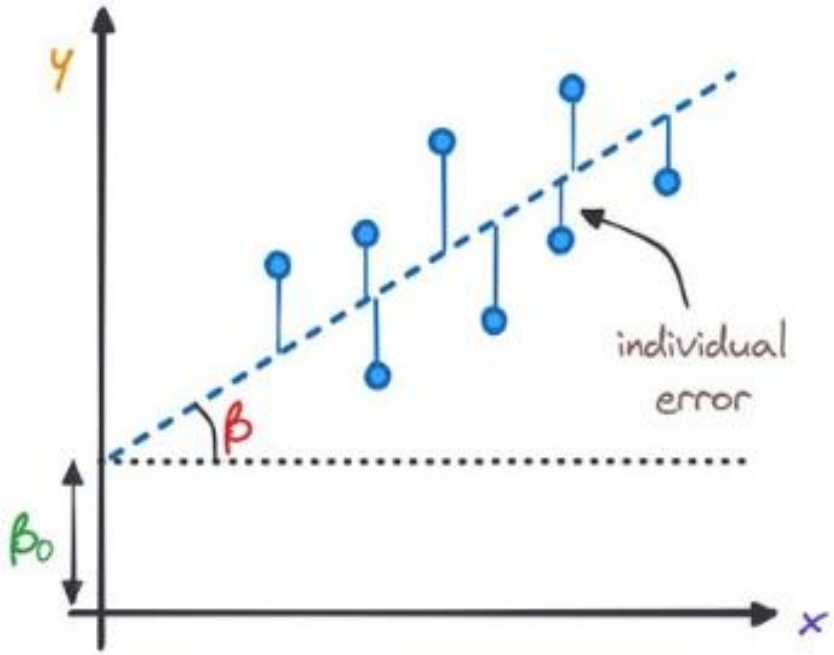
- Easy to understand & implement
- Fast to train
- Coefficients → feature importance
- Scalable to large datasets

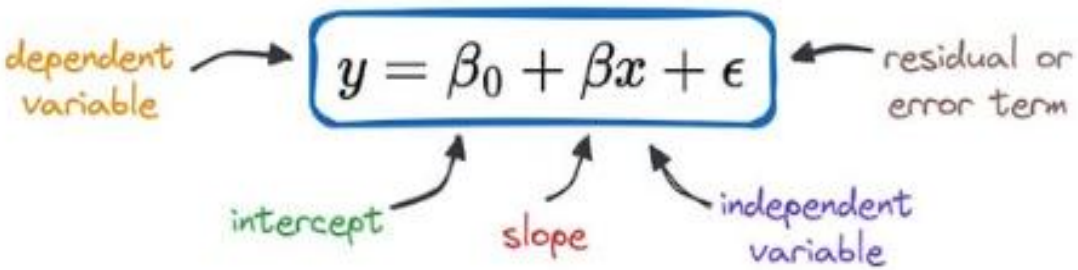
👎 Cons:

- Assumes linear relationships
- Can struggle with complex patterns
- Sensitive to outliers
- May have issues with multicollinearity

Linear Regression is a fundamental tool in machine learning.
It’s simple, fast, and interpretable, making it great for many regression tasks.

Linear Regression





Linear Regression:

Code Example:

```python
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_boston

X, y = load_boston(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
print(predictions)
```

Simple and Multiple Linear Regression

Linear regression is a statistical method used to model the relationship between a dependent variable and one independent variable. It aims to establish a linear relationship between these variables and can be used for both prediction and understanding the nature of the relationship.

Mathematical Equation

The mathematical representation of simple linear regression is:

$$Y = C_0 + C_1X + e$$

where,

- **Y:** Dependent Variable (target variable)
- **X:** Independent Variable (input variable)
- **C_0:** Intercept (value of Y when X=0)
- **C_1:** Slope of line
- **e:** Error term

Assumptions of Linear Regression

For the linear regression model to be valid.

- **Linearity**: The relationship between the independent and dependent variables should be linear.
- **Independence**: Observations should be independent of each other.
- **Homoscedasticity**: The variance of the errors should be the same across all levels of the independent variables.
- **Normality**: The dependent variable is normally distributed for a fixed value of the independent variable.
- **No Multicollinearity**: This is more pertinent for multiple regression, where all independent variables should be independent.

Limitations of Linear Regression

- **Outliers**: This can significantly impact the slope and intercept of the regression line.
- **Non-linearity**: Linear regression assumes a linear relationship, but this assumption may not hold in some cases.
- **Correlation ≠ Causation**: Just because two variables have a linear relationship doesn't mean changes in one cause changes in the other.

What is Multiple Regression?

Multiple regression is an extension of simple linear regression. It's used to model the relationship between one dependent variable and two or more independent variables. The primary purpose is to understand how the dependent variable changes as the independent variables change.

Mathematical Equation

The mathematical representation of multiple regression is:

$$Y = C_0 + C_1X_1 + C_2X_2 + C_3X_3 + ... + C_nX_n + e$$

where,

- **Y:** Dependent Variable (target variable)
- **$X_1, X_2, X_3,..., X_n$:** Independent Variable (input variable)
- **C_0:** Intercept (value of Y when X=0)
- **$C_1, C_2, C_3, C_4, C_5, ..., C_n$:** Slope of line
- **e:** Error term

Assumptions of Multiple Regression

- **Linearity:** A linear relationship exists between the dependent and independent variables.
- **Independence:** Observations are independent of each other.
- **No multicollinearity:** Independent variables aren't too highly correlated with each other.
- **Homoscedasticity:** Constant variance of the errors.
- **No Autocorrelation:** The residuals (errors) are independent.
- **Normality:** The dependent variable is normally distributed for any fixed value of the independent variables.

Limitations of Multiple Regression

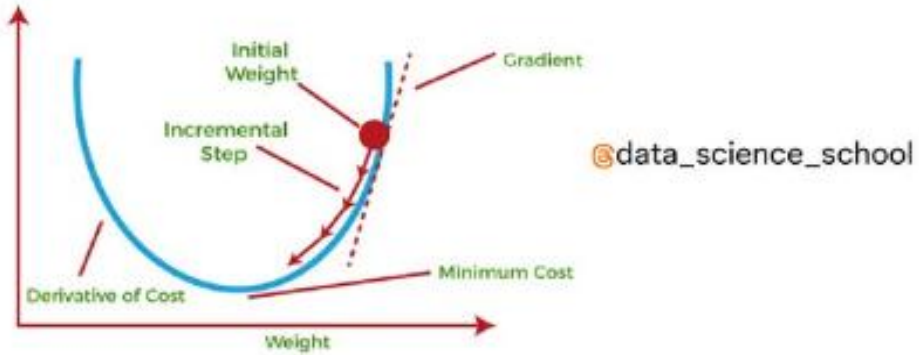
- **Overfitting:** Including too many independent variables can lead to a model that fits the training data too closely.
- **Omitted Variable Bias:** Leaving out a significant independent variable can bias the coefficients of other variables.
- **Endogeneity** occurs when an independent variable is correlated with the error term, leading to biased coefficient estimates.

Difference between Simple vs Multiple Linear Regression

Parameter	Linear (Simple) Regression	Multiple Regression
Definition	Models the relationship between one dependent and one independent variable.	Models the relationship between one dependent and two or more independent variables.
Equation	$Y = C_0 + C_1X + e$	$Y = C_0 + C_1X_1 + C_2X_2 + C_3X_3 + \ldots + C_nX_n + e$
Complexity	Simpler dealing with one relationship.	More complex due to multiple relationships.
Use Cases	Suitable when there is one clear predictor.	Suitable when multiple factors affect the outcome.
Assumptions	Linearity, Independence, Homoscedasticity, Normality	Same as linear regression, with the added concern of multicollinearity.

Gradient Descent in Machine Learning

It is the **iterative optimization algorithms** to train machine learning models by **minimizing errors between actual and expected results (cost function)** by iteratively adjusting model parameters (weights) to find the lowest cost. Further, it is also used to train Neural Networks of Deep Learning.



Note: *In ML, optimization is the task of minimizing the cost function parameterized by the model's parameters (weights and bias).*

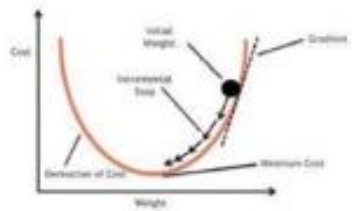
Gradient Descent for Linear Regression:

The main objective of gradient descent is to minimize the convex function using iteration of parameter updates, to find the local minimum of a function.

Ques: But what is Local Minima?

The best way to define the local minimum or local maximum of a function using gradient descent is as follows:

- If we move towards a negative gradient or away from the gradient of the function at the current point, we will reach at the **local minimum** of that function.



If we move towards a positive gradient or towards the gradient of the function at the current point, we will reach at the **local maximum** of that function.

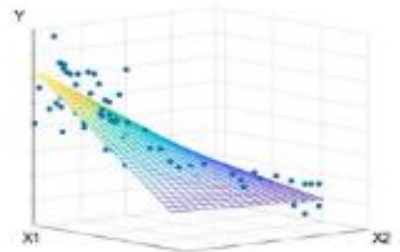
Linear Regression

Representation:

A LR model representation is a linear equation:

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_i x_i$$

β_0 is usually called intercept or **bias** coefficient. The dimension of the hyperplane of the regression is its **complexity.**



Learning:

Learning a LR means estimating the coefficients from the training data. Common methods include **Gradient Descent** or **Ordinary Least Squares.**

Variations:

There are extensions of LR training called **regularization** methods, that aim to **reduce the complexity** of the model:

- **Lasso Regression**: where OLS is modified to minimize the sum of the coefficients (L1 regularization)

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}|\beta_j| = RSS + \lambda \sum_{j=1}^{p}|\beta_j|$$

Linear Regression

- **Ridge Regression**: where OLS is modified to minimize the squared sum of the coefficients (L2 regularization)

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}\beta_j{}^2 = RSS + \lambda \sum_{j=1}^{p}\beta_j{}^2$$

where $\lambda \geq 0$ is a tuning parameter to be determined.

Data preparation:

- Transform data for linear relationship (ex: log transform for exponential relationship)
- Remove noise such as outliers
- Rescale inputs using standardization or normalization

Advantages:

+ Good regression baseline considering simplicity
+ Lasso/Ridge can be used to avoid overfitting
+ Lasso/Ridge permit feature selection in case of collinearity

Usecase examples:

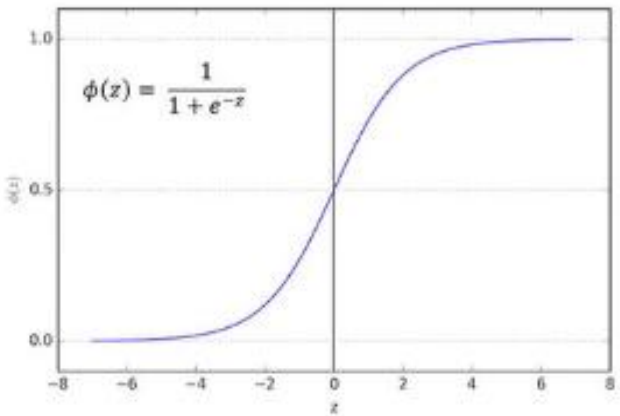
- Product sales prediction according to prices or promotions
- Call-center waiting-time prediction according to the number of complaints and the number of working agents

Logistic Regression

It is the go-to for **binary classification.**

Representation:

Logistic regression a linear method but predictions are transformed using the **logistic function** (or sigmoid):



ϕ is *S*-shaped and map real-valued number in (0,1).

The representation is an equation with binary output:

$$y = \frac{e^{\beta_0+\beta_1 x_1+\cdots+\beta_i x_i}}{1 + e^{\beta_0+\beta_1 x_1+\cdots+\beta_i x_i}}$$

Which actually models the probability of default class:

$$p(X) = \frac{e^{\beta_0+\beta_1 x_1+\cdots+\beta_i x_i}}{1 + e^{\beta_0+\beta_1 x_1+\cdots+\beta_i x_i}} = p(Y = 1|X)$$

Logistic Regression

Learning:

Learning the Logistic regression coefficients is done using **maximum-likelihood estimation**, to predict values close to 1 for default class and close to 0 for the other class.

Data preparation:

- Probability transformation to binary for classification
- Remove noise such as outliers

Advantages:

+ Good classification baseline considering simplicity
+ Possibility to change cutoff for precision/recall tradeoff
+ Robust to noise/overfitting with L1/L2 regularization
+ Probability output can be used for ranking

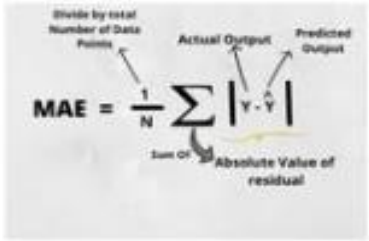
Usecase examples:

- Customer scoring with probability of purchase
- Classification of loan defaults according to profile

Regression Metrics - cheat sheet

Mean Absolute Error(MAE):

- MAE is a very simple metric which calculates the absolute difference between actual and predicted values.
- To better understand, let's take an example you have input data and output data and use Linear Regression, which draws a best-fit line.
- Now you have to find the MAE of your model which is basically a mistake made by the model known as an error. Now find the difference between the actual value and predicted value that is an absolute error but we have to find the mean absolute of the complete dataset.
- so, sum all the errors and divide them by a total number of observations and this is MAE. And we aim to get a minimum MAE because this is a loss.



MAE represents the average absolute difference between the actual and predicted values. It gives an idea of the **average magnitude of the errors** in the predictions. A lower MAE indicates better model performance.

Example:
Take 10 actual and predicted values of house prices, find Mean Absolute Error(MAE), and interpret the result

Actual Price	Predicted Price	Absolute Difference
$250,000	$245,000	$5,000
$310,000	$307,500	$2,500
$425,000	$430,000	$5,000
$189,000	$195,000	$6,000
$550,000	$542,000	$8,000
$295,000	$298,000	$3,000
$375,000	$368,000	$7,000
$480,000	$475,000	$5,000
$220,000	$223,000	$3,000
$615,000	$620,000	$5,000

- Sum the absolute differences: $49,500
- Divide the sum by the number of data points (10) to find the MAE: $49,500 / 10 = $4950
- The MAE of $4950 falls within the low MAE range (between $1,000 and $5,000). This indicates that the model's fairly accurate in predicting house prices, with an average error of around $4950.

Mean Squared Error(MSE):

- MSE is a most used and very simple metric with a little bit of change in mean absolute error. Mean squared error states that finding the squared difference between actual and predicted value.
- So, in the above we are finding the absolute difference and here we are finding the squared difference.
- What actually the MSE represents?
  - It represents the squared distance between actual and predicted values. we perform squared to avoid the cancellation of negative terms and it is the benefit of MSE.

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \hat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$

MAE is less sensitive to outliers compared to MSE because it doesn't square the errors. If your dataset contains outliers, MAE may provide a more accurate reflection of the model's performance.

MSE is similar to MAE but emphasizes larger errors due to the squaring. It gives the average of the squared differences between actual and predicted values. Like MAE, a lower MSE is desirable, and it is sensitive to outliers.

Example:
Assume 10 actual and predicted values of stock prices, find Mean Squared Error(MSE), and interpret the result

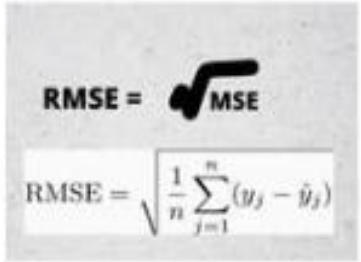
Actual Price	Predicted Price	Squared Error
$25.50	$25.25	0.0625
$31.75	$31.50	0.0625
$42.25	$42.50	0.0625
$18.75	$19.00	0.0625
$54.75	$54.50	0.0625
$29.25	$29.00	0.0625
$37.00	$36.75	0.0625
$47.75	$47.50	0.0625
$21.75	$22.00	0.0625
$61.25	$61.50	0.0625

- Sum the squared errors: 0.625
- Divide the sum by the number of data points (10) to find the MSE: 0.625 / 10 = 0.0625

- The MSE of 0.0625 is very low.
- This indicates that the model is highly accurate in predicting stock prices.
- The average squared difference between the actual and predicted prices is only 0.0625, which is a very small amount.

Root Mean Squared Error(RMSE):

- As RMSE is clear by the name itself, that it is a simple square root of mean squared error.



RMSE is the square root of MSE. It is in the same units as the dependent variable, providing a more interpretable measure. Like MSE, lower values indicate better model performance.

Example:
Assume 10 actual and predicted values of stock prices, find the Root Mean Squared Error(RMSE) step by step, and interpret the result

Actual Price	Predicted Price	Squared Error
$25.50	$25.25	0.0625
$31.75	$31.50	0.0625
$42.25	$42.50	0.0625
$18.75	$19.00	0.0625
$54.75	$54.50	0.0625
$29.25	$29.00	0.0625
$37.00	$36.75	0.0625
$47.75	$47.50	0.0625
$21.75	$22.00	0.0625
$61.25	$61.50	0.0625

- The MSE has already been calculated as 0.0625.
- Calculate Root Mean Squared Error (RMSE):
- Take the square root of the MSE to find the RMSE: $\sqrt{0.0625} = 0.25$

- The RMSE of 0.25 is low.
- This indicates that the model is fairly accurate in predicting stock prices.
- On average, the predicted prices are off by about $0.25 from the actual prices.

Root Mean Squared Log Error(RMSLE):

- Taking the log of the RMSE metric slows down the scale of error. The metric is very helpful when you are developing a model without calling the inputs. In that case, the output will vary on a large scale.
- To control this situation of RMSE we take the log of calculated RMSE error and resultant we get as RMSLE.

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(1+y_i) - \log(1+\hat{y}_i))^2}$$

Where:

- n is the number of observations.
- yi is the actual value for observation.
- y^i is the predicted value for observation.

RMSLE is often used when the target variable has a wide range. It penalizes underestimation more than overestimation. It is particularly useful for predicting values across multiple orders of magnitude.

Example:
Assume 10 actual and predicted values of weather temperature, find the Root Mean Squared Log Error(RMSLE) step by step, and interpret the result

Actual Temperature	Predicted Temperature	Log(Actual + 1)	Log(Predicted + 1)	Log squared Error
75	72	4.317	4.277	0.0016
82	80	4.407	4.382	0.000625
68	70	4.220	4.248	0.000784
91	89	4.511	4.488	0.000529
70	68	4.248	4.220	0.000784
85	83	4.443	4.419	0.000576
77	75	4.344	4.317	0.000729
65	63	4.174	4.143	0.000961
88	90	4.477	4.500	0.000529
73	75	4.290	4.317	0.000729

- Square each log error.
- Sum the squared log errors: 0.007846
- Divide the sum by the number of data points (10) to find the MSLE: 0.007846 / 10 = 0.00054
- Take the square root of the MSLE to find the RMSLE: $\sqrt{0.0007846} = 0.028$

- The RMSLE of 0.028 is very low.
- This indicates that the model is highly accurate in predicting weather temperatures.
- On average, the predicted temperatures are off by about 2.8% from the actual temperatures in terms of their logarit˙mic differences.

R Squared (R^2) and Adjusted R Squared:

- The R^2 (R-squared) and adjusted R^2 are metrics used to evaluate the goodness of fit of a regression model.
- They provide insights into how well the independent variables explain the variability in the dependent variable.

R^2 (R-squared):

- R^2 is a statistical measure that represents the proportion of the variance in the dependent variable (response variable) that is explained by the independent variables (predictors) in a regression model. It is expressed as a percentage and ranges from 0% to 100%.

The formula for R^2 is:

$$R2=1-RSS/TSS$$

Where:

- RSS is the Residual Sum of Squares, which measures the variance not explained by the model.
- TSS is the Total Sum of Squares, which measures the total variance in the dependent variable.

R^2 close to 1: Indicates that a large proportion of the variance in the dependent variable is explained by the model.
R^2 close to 0: Suggests that the model is not explaining much of the variability in the dependent variable.

However, R^2 has a limitation. As you add more predictors to the model, R^2 will always increase, even if the additional predictors don't contribute much to explaining the variance. This is where Adjusted R^2 comes into play.

Adjusted R^2:

- Adjusted R^2 is a modified version of R^2 that penalizes the inclusion of irrelevant predictors in the model. It accounts for the number of predictors in the model and adjusts R^2 to provide a more realistic measure of model fit, especially when dealing with multiple predictors.

The formula for Adjusted R^2 is:

$$R_a^2 = 1 - \left[\left(\frac{n-1}{n-k-1}\right) \times (1 - R^2)\right]$$

where:
n = number of observations
k = number of independent variables
R_a^2 = adjusted R^2

- Adjusted R^2 values are always less than or equal to R^2.

- R-squared measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1, with 1 indicating a perfect fit. Higher R-squared values suggest a better fit of the model to the data.
- Adjusted R-squared is a modification of R-squared that adjusts for the number of predictors in the model. It penalizes the addition of unnecessary predictors, providing a more reliable measure of the model's goodness of fit when comparing models with different numbers of predictors.

Example:
X = [1, 2, 3, 4, 5]
Y = [2, 4, 5, 4, 5]

Calculate the Mean of Y ($\hat{Y}$)
$\hat{Y} = (1/n) \sum Yi = (2 + 4 + 5 + 4 + 5) / 5 = 4$

Calculate the Total Sum of Squares (TSS)
$TSS = \sum (Yi - \hat{Y})^2$
$TSS = (2 - 4)^2 + (4 - 4)^2 + (5 - 4)^2 + (4 - 4)^2 + (5 - 4)^2$
$TSS = 2 + 0 + 1 + 0 + 1 = 4$

Fit a Linear Regression Model
Fit a linear regression model: $Y = \beta0 + \beta1X + \epsilon$
For simplicity, let's assume that the fitted model is $Y = 2 + 0.6X + \epsilon$.

Calculate the Residual Sum of Squares (RSS)
$RSS = \sum(Yi - (\beta0 + \beta1Xi))^2$
$RSS = (2 - (2 + 0.6 * 1))^2 + (4 - (2 + 0.6 * 2))^2 + (5 - (2 + 0.6 * 3))^2 + (4 - (2 + 0.6 * 4))^2 + (5 - (2 + 0.6 * 5))^2$

$RSS= (2-2.6)^2+(4-3.2)^2+(5-3.8)^2+(4-4.4)^2+(5-5)^2$
$RSS=0.36+0.64+1.44+0.64+0$
$RSS=3.08$

Calculate R^2
$R^2 = 1 - (RSS / TSS)$
$R^2 = 1 - (3.08 / 4)$
$R^2 = 0.23$

Calculate Adjusted R^2
k is the number of predictors (1), n is the number of observations (5)

Adjusted $R^2 = 1 - ((1 - R^2) * (n - 1) / (n - k - 1))$
Adjusted $R^2 = 1 - ((1 - 0.23) * (5 - 1) / (5 - 1 - 1))$
Adjusted $R^2 = 0.0267$

R^2 value of 0.23 indicates that approximately 23% of the variance in the dependent variable (Y) is explained by the linear regression model.
Adjusted R^2 takes into account the number of predictors. In this case, the adjusted R^2 is 0.0267.

- Lower values for MAE, MSE, RMSE, and RMSLE indicate better model accuracy.
- Higher values for R^2 and Adjusted R^2 indicate better model fit.
- Choose metrics based on the problem and model characteristics.
- Consider sensitivity to outliers, interpretability, and model complexity.
- Use multiple metrics to get a comprehensive evaluation of model performance.

Model Evaluation Techniques

(For Your Regression Model)

- Mean Error (ME)

$$ME = \frac{1}{T}\sum_{t=1}^{n} e_t$$

- Mean Absolute Error (MAE) or Mean Absolute Deviation (MAD)

$$MAD = \frac{1}{n}\sum_{t=1}^{n} |e_t|$$

- Mean Squared Error (MSE)

$$MSE = \frac{1}{n}\sum_{t=1}^{n} e_t^2$$

- Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$$

- Mean Percentage Error (MPE)

$$MPE = \frac{1}{n}\sum_{t=1} \frac{e_t}{Y_t}$$

- Mean Absolute Percentage Error (MAPE)

$$MAPE = \frac{1}{n}\sum_{t=1} \left|\frac{e_t}{Y_t}\right|$$

- Mean Absolute Scaled Error (MASE)

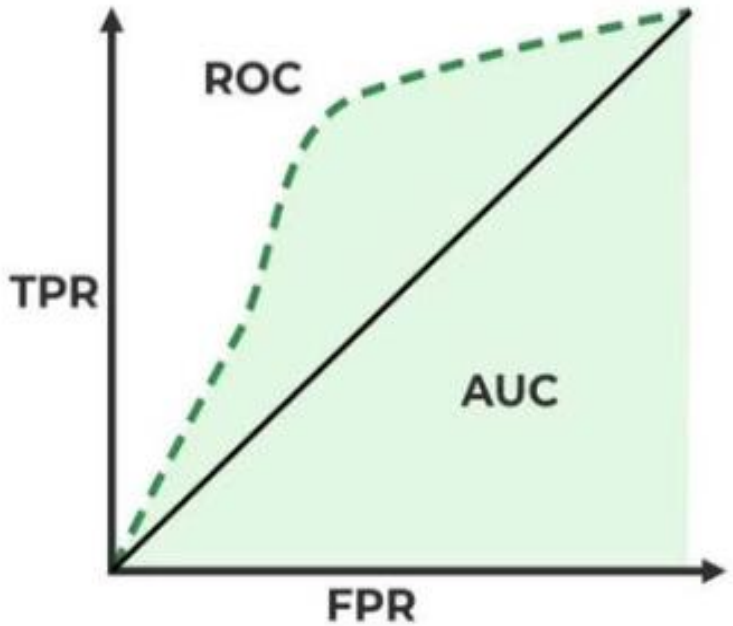
$$MASE = \frac{MAE}{MAE_{\text{in-sample, naive}}}$$

Top 5 Important Topics in Machine Learning

(Every Data Scientist should know)

ROC and AUC

- **Receiver Operator Characteristic (ROC)** curves are graphs showing classifiers' performance by plotting the true positive rate and false positive rate.
  - True Positive Rate = **Sensitivity**
  - False Positive Rate = **1 - Specificity**
- The area under the ROC curve (AUC) measures the performance of machine learnin algorithms.



Top 5 Important Topics in Machine Learning

(Every Data Scientist should know)

Sensitivity and Specificity

1. **Sensitivity** : Sensitivity is the metric that evaluates a model's ability to predict true positives of each available category.

$$\text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

2. **Specificity** : Specificity is the metric that evaluates a model's ability to predict true negatives of each available category

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}$$

Precision and Recall

1. **Precision** : Its the percentage of predicted positives correctly predicted.

$$\text{Precision} = \frac{TP}{TP + FP}$$

2. **Recall** : its the same as sensitivity.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Top 5 Important Topics in Machine Learning

(Every Data Scientist should know)

Cross Validation

Cross Validation allows us to compare different machine learning methods and get a sense of how well they will work in practice.

Steps

1. Estimate the parameters for machine learning methods. This is called training the algorithm.
2. Evaluate how well the machine learning methods work. This is called testing the algorithm.

Every block of data is used for testing and we can compare methods by seeing how well they performed.

Confusion Matrix

A confusion matrix is an intuitive table that lets us see the detailed classification results of an algorithm on a test set by analyzing its rows, columns, or entries.

- The rows in a Confusion Matrix correspond to what the machine learning algorithm predicted.
- The columns correspond to known truth.

Bias and Variance in Machine Learning - Cheatsheet

What is Bias?

In general, a machine learning model analyses the data, find patterns in it and make predictions. While training, the model learns these patterns in the dataset and applies them to test data for prediction. ***While making predictions, a difference occurs between prediction values made by the model and actual values/expected values, and this difference is known as bias errors or Errors due to bias.*** It can be defined as an inability of machine learning algorithms such as Linear Regression to capture the true relationship between the data points. Each algorithm begins with some amount of bias because bias occurs from assumptions in the model, which makes the target function simple to learn. A model has either:

- **Low Bias:** A low bias model will make fewer assumptions about the form of the target function.
- **High Bias:** A model with a high bias makes more assumptions, and the model becomes unable to capture the important features of our dataset. **A high bias model also cannot perform well on new data.**

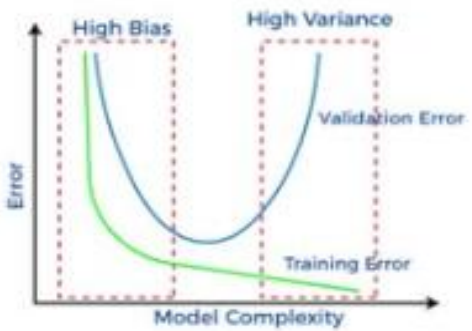
Generally, a linear algorithm has a high bias, as it makes them learn fast. The simpler the algorithm, the higher the bias it has likely to be introduced. Whereas a nonlinear algorithm often has low bias.

Some examples of machine learning algorithms with low bias **are Decision Trees, k-Nearest Neighbours and Support Vector Machines**. At the same time, an algorithm with high bias is **Linear Regression, Linear Discriminant Analysis and Logistic Regression.**

Ways to reduce High Bias:

High bias mainly occurs due to a much simple model. Below are some ways to reduce the high bias:

- Increase the input features as the model is underfitted.
- Decrease the regularization term.
- Use more complex models, such as including some polynomial features.



What is a Variance Error?

The variance would specify the amount of variation in the prediction if the different training data was used. In simple words, ***variance tells that how much a random variable is different from its expected value.*** Ideally, a model should not vary too much from one training dataset to another, which means the algorithm should be good in understanding the hidden mapping between inputs and output variables. Variance errors are either of **low variance or high variance.**

Low variance means there is a small variation in the prediction of the target function with changes in the training data set. At the same time, **High variance** shows a large variation in the prediction of the target function with changes in the training dataset.

A model that shows high variance learns a lot and perform well with the training dataset, and does not generalize well with the unseen dataset. As a result, such a model gives good results with the training dataset but shows high error rates on the test dataset.

Since, with high variance, the model learns too much from the dataset, it leads to overfitting of the model. A model with high variance has the below problems:

- A high variance model leads to overfitting.
- Increase model complexities.

Usually, nonlinear algorithms have a lot of flexibility to fit the model, have high variance.

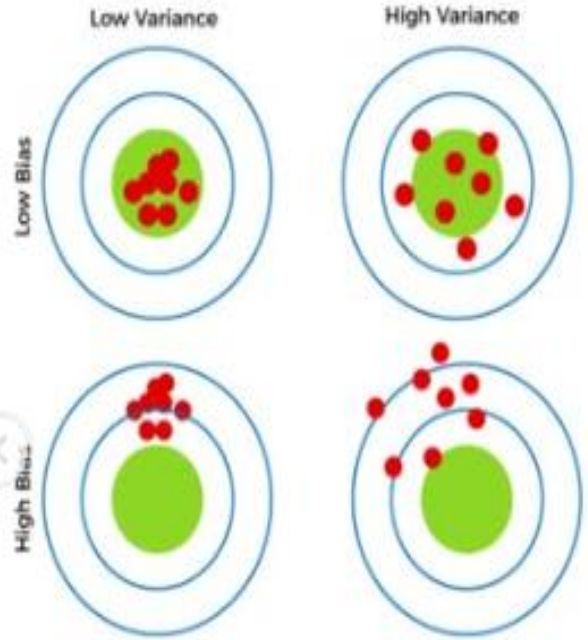
Some examples of machine learning algorithms with low variance are, **Linear Regression, Logistic Regression, and Linear discriminant analysis**. At the same time, algorithms with high variance are **decision tree, Support Vector Machine, and K-nearest neighbours.**

Ways to Reduce High Variance:

- Reduce the input features or number of parameters as a model is overfitted.
- Do not use a much complex model.
- Increase the training data.
- Increase the Regularization term.

Different Combinations of Bias-Variance

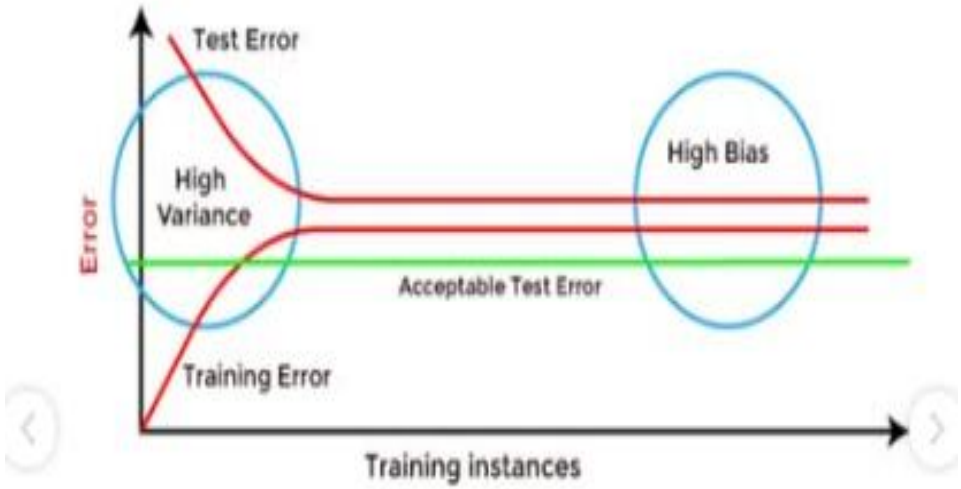
There are four possible combinations of bias and variances, which are represented by the below diagram:



1. **Low-Bias, Low-Variance:** The combination of low bias and low variance shows an ideal machine learning model. However, it is not possible practically.
2. **Low-Bias, High-Variance:** With low bias and high variance, model predictions are inconsistent and accurate on average. This case occurs when the model learns with a large number of parameters and hence leads to an **overfitting**
3. **High-Bias, Low-Variance:** With High bias and low variance, predictions are consistent but inaccurate on average. This case occurs when a model does not learn well with the training dataset or uses few numbers of the parameter. It leads to **underfitting** problems in the model.
4. **High-Bias, High-Variance:** With high bias and high variance, predictions are inconsistent and also inaccurate on average.

How to identify High variance or High Bias?

High variance can be identified if the model has:



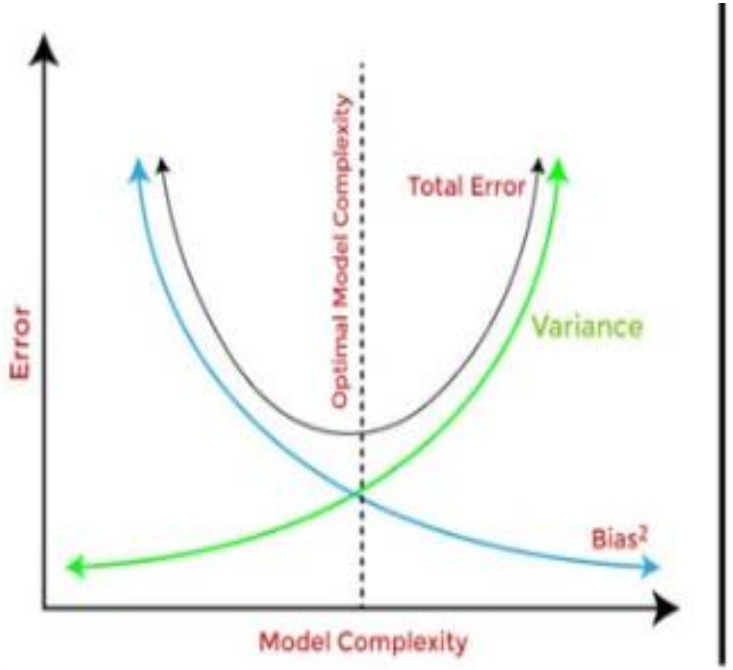
- Low training error and high test error.

High Bias can be identified if the model has:

- High training error and the test error is almost similar to training error.

Bias-Variance Trade-Off

While building the machine learning model, it is really important to take care of bias and variance in order to avoid overfitting and underfitting in the model. If the model is very simple with fewer parameters, it may have low variance and high bias. Whereas, if the model has a large number of parameters, it will have high variance and low bias. So, it is required to make a balance between bias and variance errors, and this balance between the bias error and variance error is known as **the Bias-Variance trade-off.**



For an accurate prediction of the model, algorithms need a low variance and low bias. But this is not possible because bias and variance are related to each other:

- If we decrease the variance, it will increase the bias.
- If we decrease the bias, it will increase the variance.

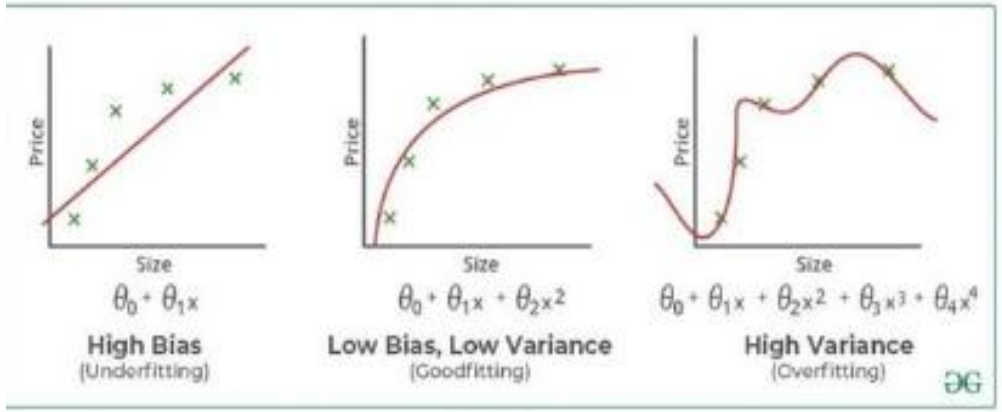
Bias-Variance trade-off is a central issue in supervised learning. Ideally, we need a model that accurately captures the regularities in training data and simultaneously generalizes well with the unseen dataset. Unfortunately, doing this is not possible simultaneously. Because a high variance algorithm may perform well with training data, but it may lead to overfitting to noisy data. Whereas, high bias algorithm generates a much simple model that may not even capture important regularities in the data. So, we need to find a sweet spot between bias and variance to make an optimal model.

Hence, the ***Bias-Variance trade-off is about finding the sweet spot to make a balance between bias and variance errors.***

Overfitting and Underfitting in Machine Learning

Bias: Bias refers to the error due to overly simplistic assumptions in the learning algorithm. These assumptions make the model easier to comprehend and learn but might not capture the underlying complexities of the data. It is the error due to the model's inability to represent the true relationship between input and output accurately. When a model has poor performance both on the training and testing data means high bias because of the simple model, indicating underfitting.

Variance: Variance, on the other hand, is the error due to the model's sensitivity to fluctuations in the training data. It's the variability of the model's predictions for different instances of training data. High variance occurs when a model learns the training data's noise and random fluctuations rather than the underlying pattern. As a result, the model performs well on the training data but poorly on the testing data, indicating overfitting.



Bias and Variance

Underfitting in Machine Learning

A statistical model or a machine learning algorithm is said to have underfitting when a model is too simple to capture data complexities. It represents the inability of the model to learn the training data effectively result in poor performance both on the training and testing data. In simple terms, an underfit model's are inaccurate, especially when applied to new, unseen examples.

> High bias and low variance.

Reasons for Under fitting

1. The model is too simple, So it may be not capable to represent the complexities in the data.
2. The input features which is used to train the model is not the adequate representations of underlying factors influencing the target variable.
3. The size of the training dataset used is not enough.
4. Excessive regularization are used to prevent the overfitting, which constraint the model to capture the data well.
5. Features are not scaled.

Techniques to Reduce Under-fitting

1. Increase model complexity.
2. Increase the number of features, performing feature engineering.
3. Remove noise from the data.
4. Increase the number of epochs or increase the duration of training to get better result

Overfitting in Machine Learning

A statistical model is said to be overfitted when the model does not make accurate predictions on testing data. When a model gets trained with so much data, it starts learning from the noise and inaccurate data entries in our data set. And when testing with test data results in High variance. Then the model does not categorize the data correctly, because of too many details and noise.

> Low Bias and High Variance

Reasons for Overfitting:

1. High variance and low bias.
2. The model is too complex.
3. The size of the training data.

Techniques to Reduce Overfitting

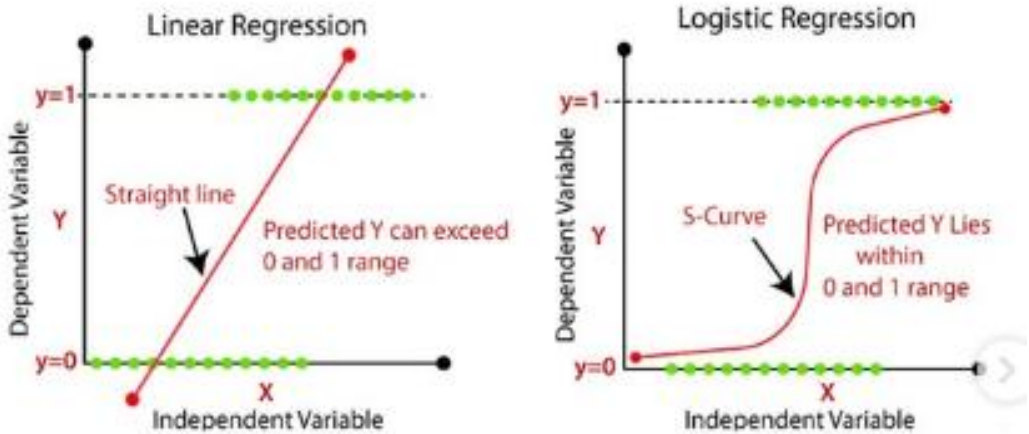
1. Improving the quality of training data reduces overfitting by focusing on meaningful patterns, mitigate the risk of fitting the noise or irrelevant features.
2. Increase the training data can improve the model's ability to generalize to unseen data and reduce the likelihood of overfitting.
3. Reduce model complexity.
4. Early stopping during the training phase (have an eye over the loss over the training period as soon as loss begins to increase stop training).
5. Ridge Regularization and Lasso Regularization.
6. Use dropout for neural networks to tackle overfitting.

Classification Metrics in Machine Learning

Accuracy metric	Also known as	Computation
Sensitivity (TPR)	True positive rate; Hit rate; Recall; Probability of detection	$\frac{TP}{TP+FN}$
Specificity (TNR)	True negative rate	$\frac{TN}{TN+FP}$
Positive predictivity (PPV)	Positive predictive value; Precision	$\frac{TP}{TP+FP}$
Negative predictivity (NPV)	Negative predictive value	$\frac{TN}{TN+FN}$
False negative rate (FNR)	Miss rate	$\frac{FN}{FN+TP}$
False positive rate (FPR)	Fall-out	$\frac{FP}{FP+TN}$
False discovery rate (FDR)	n/a	$\frac{FP}{FP+TP}$
False ommision rate (FOR)	n/a	$\frac{FN}{FN+TN}$
Percent correctly classified (PCC)	Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$

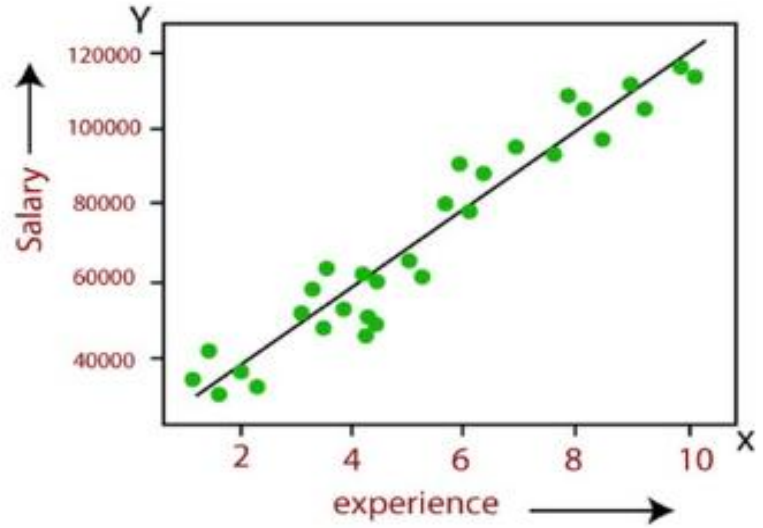
Linear Regression vs Logistic Regression

Linear Regression and Logistic Regression are the two famous Machine Learning Algorithms which come under supervised learning technique. Since both the algorithms are of supervised in nature hence these algorithms use labeled dataset to make the predictions. But the main difference between them is how they are being used. The Linear Regression is used for solving Regression problems whereas Logistic Regression is used for solving the Classification problems. The description of both the algorithms is given below along with difference table.



Linear Regression:

- Linear Regression is one of the most simple Machine learning algorithm that comes under Supervised Learning technique and used for solving regression problems.
- It is used for predicting the continuous dependent variable with the help of independent variables.
- The goal of the Linear regression is to find the best fit line that can accurately predict the output for the continuous dependent variable.
- If single independent variable is used for prediction then it is called Simple Linear Regression and if there are more than two independent variables then such regression is called as Multiple Linear Regression.
- By finding the best fit line, algorithm establish the relationship between dependent variable and independent variable. And the relationship should be of linear nature.
- The output for Linear regression should only be the continuous values such as price, age, salary, etc. The relationship between the dependent variable and independent variable can be shown in below image:



In above image the dependent variable is on Y-axis (salary) and independent variable is on x-axis(experience). The regression line can be written as:

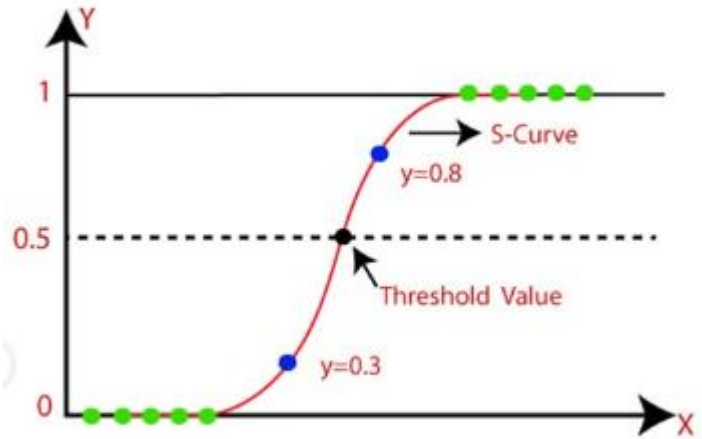
$y= a_0+a_1x+ \varepsilon$

Where, a_0 and a_1 are the coefficients and ε is the error term.

Logistic Regression:

- Logistic regression is one of the most popular Machine learning algorithm that comes under Supervised Learning techniques.
- It can be used for Classification as well as for Regression problems, but mainly used for Classification problems.
- Logistic regression is used to predict the categorical dependent variable with the help of independent variables.
- The output of Logistic Regression problem can be only between the 0 and 1.
- Logistic regression can be used where the probabilities between two classes is required. Such as whether it will rain today or not, either 0 or 1, true or false etc.
- Logistic regression is based on the concept of Maximum Likelihood estimation. According to this estimation, the observed data should be most probable.
- In logistic regression, we pass the weighted sum of inputs through an activation function that can map values in between 0 and 1. Such activation function is known as **sigmoid function** and the curve obtained is called as sigmoid curve or S-curve. Consider the below image:

Logistic Regression:



- The equation for logistic regression is:

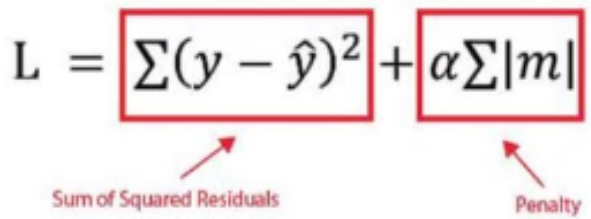
$$\log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

Linear Regression vs Logistic Regression

Linear Regression	Logistic Regression
Linear Regression is a supervised regression model.	Logistic Regression is a supervised classification model.
In Linear Regression, we predict the value by an integer number.	In Logistic Regression, we predict the value by 1 or 0.
Here no activation function is used.	Here activation function is used to convert a linear regression equation to the logistic regression equation
Here no threshold value is needed.	Here a threshold value is added.
Here we calculate Root Mean Square Error(RMSE) to predict the next weight value.	Here we use precision to predict the next weight value.
Here dependent variable should be numeric and the response variable is continuous to value.	Here the dependent variable consists of only two categories. Logistic regression estimates the odds outcome of the dependent variable given a set of quantitative or categorical independent variables.
It is based on the least square estimation.	It is based on maximum likelihood estimation.
Here when we plot the training datasets, a straight line can be drawn that touches maximum plots.	Any change in the coefficient leads to a change in both the direction and the steepness of the logistic function. It means positive slopes result in an S-shaped curve and negative slopes result in a Z-shaped curve.
Linear regression is used to estimate the dependent variable in case of a change in independent variables. For example, predict the price of houses.	Whereas logistic regression is used to calculate the probability of an event. For example, classify if tissue is benign or malignant.

LASSO Regression (L1 Regularization)

LASSO stands for Least Absolute Shrinkage and Selection Operator, is a popular technique used in statistical modeling and machine learning to estimate the relationships between variables and make predictions.



Why do we need LASSO Regression?

The primary goal of LASSO regression is to reduce overfitting by shrinking and selecting features with less importance.

L1 regularization adds a penalty that is equal to the absolute value of the magnitude of the coefficient to the traditional linear regression model, which encourages sparse solutions where some coefficients are forced to be exactly zero.

Note: This feature makes LASSO particularly useful for **Feature Selection**, as it can automatically identify and discard irrelevant or redundant variables.

$$L_1 = \alpha * (|\beta_1| + |\beta_2| + ... + |\beta_p|)$$

Where:

```
    α is the regularization parameter that controls the amount of
regularization applied.
    β₁, β₂, ..., βₚ are the coefficients.
```

Note: A larger α value increases the amount of regularization, leading to more coefficients being pushed towards zero.

Conversely, a smaller α value reduces the regularization effect, allowing more variables to have non-zero coefficients.

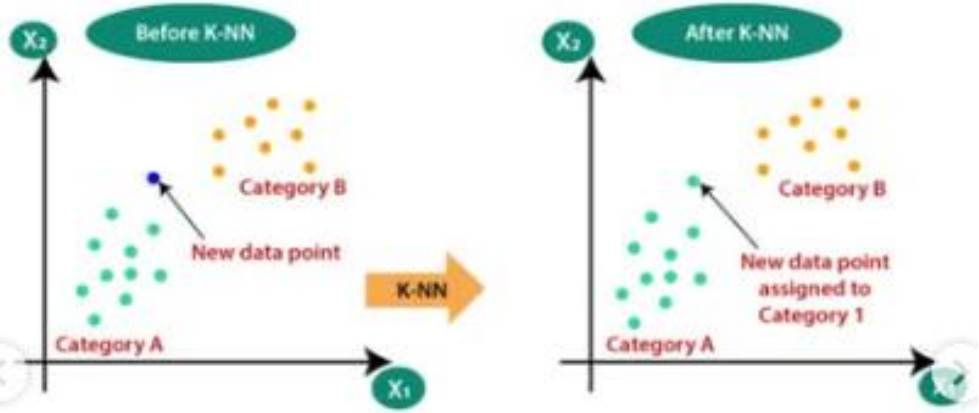
Polynomial Regression

K-Nearest Neighbor(KNN) Algorithm in Machine Learning

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is **a non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

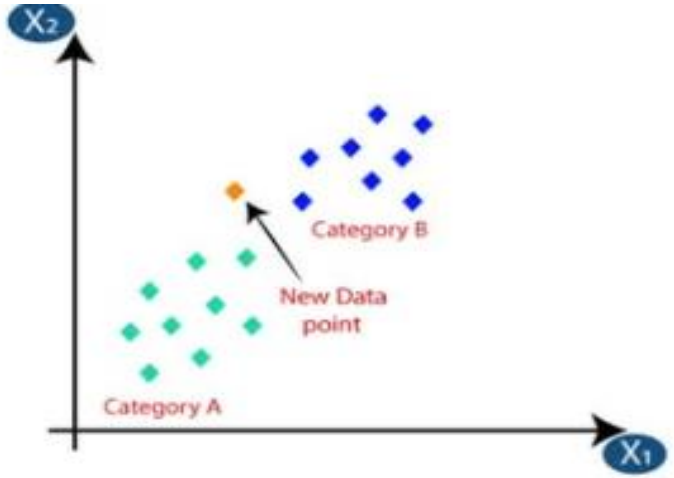


How does K-NN work?

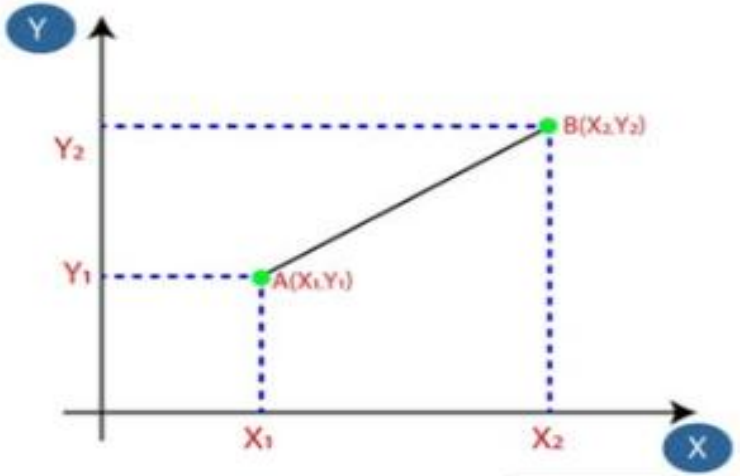
The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:



- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



$$\text{Euclidean Distance between } A_1 \text{ and } B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$$

- By calculating the Euclidean distance we get the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:

- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

Decision Tree Algorithm

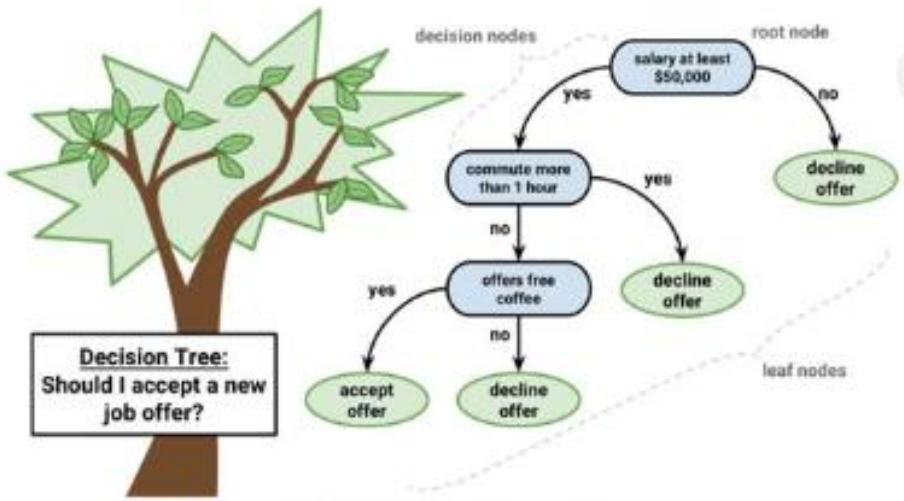
Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, decision tree algorithm can be used for solving **regression and classification problems** too.

The general motive of using Decision Tree is to create a training model which can use to predict class or value of target variables by **learning decision rules** inferred from prior data (training data).

The understanding level of Decision Trees algorithm is so easy compared with other classification algorithms. The decision tree algorithm tries to solve the problem, by using tree representation. Each **internal node** of the tree corresponds to an attribute, and each **leaf node** corresponds to a class label.

Decision Tree Algorithm Pseudocode

1. Place the best attribute of the dataset at the **root** of the tree.
2. Split the training set into **subsets**. Subsets should be made in such a way that each subset contains data with the same value for an attribute.
3. Repeat step 1 and step 2 on each subset until you find **leaf nodes** in all the branches of the tree.



Decision Tree classifier. **Image credit:** www.packtpub.com

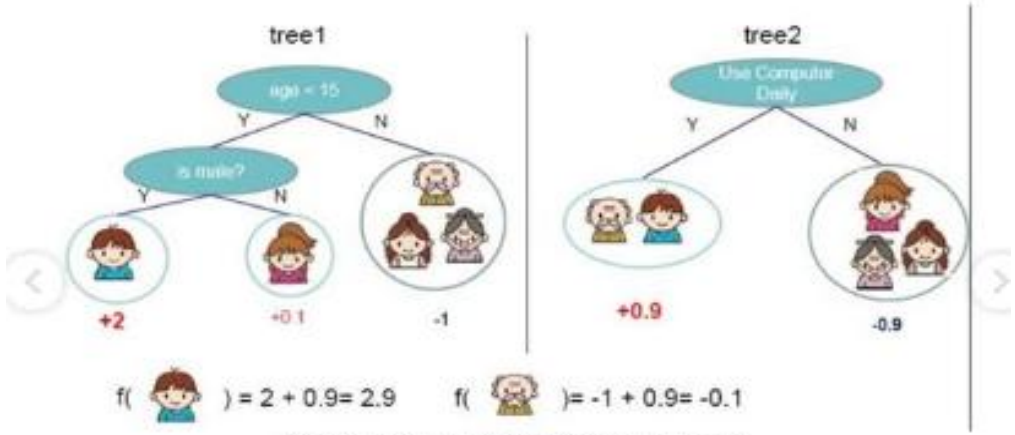
In decision trees, for predicting a class label for a record we start from the **root** of the tree. We compare the values of the root attribute with record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

We continue comparing our record's attribute values with other **internal nodes** of the tree until we reach **a leaf node** with predicted class value. As we know how the modeled decision tree can be used to predict the target class or the value. Now let's understanding how we can create the decision tree model.

Assumptions while creating Decision Tree

The below are the some of the assumptions we make while using Decision tree:

- At the beginning, the whole training set is considered as the **root.**
- Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
- Records are **distributed recursively** on the basis of attribute values.
- Order to placing attributes as root or internal node of the tree is done by using some statistical approach.



Decision tree model example Image Credit: **http://champeryfang.github.io/**

Decision Trees follow **Sum of Product (SOP)** representation. For the above images, you can see how we can predict **can we accept the new job offer? and Use computer daily?** from traversing for the root node to the leaf node.

It's a sum of product representation. The Sum of product(SOP) is also known as **Disjunctive Normal Form**. For a class, every branch from the root of the tree to a leaf node having the same class is a conjunction(product) of values, different branches ending in that class form a disjunction(sum).

The primary challenge in the decision tree implementation is to identify which attributes do we need to consider as the root node and each level. Handling this is know the attributes selection. We have different attributes selection measure to identify the attribute which can be considered as the root note at each level.

The popular attribute selection measures:

- Information gain
- Gini index

Attributes Selection

If dataset consists of **"n"** attributes then deciding which attribute to place at the root or at different levels of the tree as internal nodes is a complicated step. By just randomly selecting any node to be the root can't solve the issue. If we follow a random approach, it may give us bad results with low accuracy.

For solving this attribute selection problem, researchers worked and devised some solutions. They suggested using some *criterion* like **information gain, gini index,** etc. These criterions will calculate values for every attribute. The values are sorted, and attributes are placed in the tree by following the order i.e, the attribute with a high value(in case of information gain) is placed at the root.

While using information Gain as a criterion, we assume attributes to be categorical, and for gini index, attributes are assumed to be continuous.

Information Gain

By using information gain as a criterion, we try to estimate the information contained by each attribute.

We are going to use some points deducted from **information theory.**
To measure the randomness or uncertainty of a random variable X is defined by **Entropy**.

For a binary classification problem with only two classes, positive and negative class.

- If all examples are positive or all are negative then entropy will be zero i.e, low.
- If half of the records are of positive class and half are of negative class then entropy is one i.e, high.

By calculating **entropy measure** of each attribute we can calculate their **information gain**. Information Gain calculates the expected reduction in entropy due to sorting on the attribute. Information gain can be calculated.

To get a clear understanding of calculating **information gain & entropy**, we will try to implement it on a sample data.

Example: Construct a Decision Tree by using "information gain" as a criterion

	A	B	C	D	E
1	4.8	3.4	1.9	0.2	positive
2	5	3	1.6	0.2	positive
3	5	3.4	1.6	0.4	positive
4	5.2	3.5	1.5	0.2	positive
5	5.2	3.4	1.4	0.2	positive
6	4.7	3.2	1.6	0.2	positive
7	4.8	3.1	1.6	0.2	positive
8	5.4	3.4	1.5	0.4	positive
9	7	3.2	4.7	1.4	negative
10	6.4	3.2	4.5	1.5	negative
11	6.9	3.1	4.9	1.5	negative
12	5.5	2.3	4	1.3	negative
13	6.5	2.8	4.6	1.5	negative
14	5.7	2.8	4.5	1.3	negative
15	6.3	3.3	4.7	1.6	negative
16	4.9	2.4	3.3	1	negative

We are going to use this data sample. Let's try to use information gain as a criterion. Here, we have 5 columns out of which 4 columns have continuous data and 5th column consists of class labels.

A, B, C, D attributes can be considered as predictors and E column class labels can be considered as a target variable. For constructing a decision tree from this data, we have to convert continuous data into categorical data.

We have chosen some random values to categorize each attribute:

A	B	C	D
>= 5	>= 3.0	>= 4.2	>= 1.4
< 5	< 3.0	< 4.2	< 1.4

There are **2 steps for calculating information gain** for each attribute:

1. Calculate entropy of Target.
2. Entropy for every attribute A, B, C, D needs to be calculated. Using information gain formula we will subtract this entropy from the entropy of target. The result is Information Gain.

The entropy of Target: We have 8 records with negative class and 8 records with positive class. So, we can directly estimate the entropy of target as 1.

Variable E	
Positive	Negative
8	8

Calculating entropy using formula:

E(8,8) = -1*((p(+ve)*log(p(+ve)) + (p(-ve)*log(p(-ve)))
= -1*((8/16)*log2(8/16)) + (8/16) * log2(8/16))
= 1

Information gain for Var A

Var A has value >=5 for 12 records out of 16 and 4 records with value <5 value.

- For Var A >= 5 & class == positive: 5/12
- For Var A >= 5 & class == negative: 7/12
  - Entropy(5,7) = -1 * ((5/12)*log2(5/12) + (7/12)*log2(7/12)) = 0.9799
- For Var A <5 & class == positive: 3/4

- For Var A <5 & class == negative: 1/4
  - Entropy(3,1) = -1 * ((3/4)*log2(3/4) + (1/4)*log2(1/4)) = 0.81128

Entropy(Target, A) = P(>=5) * E(5,7) + P(<5) * E(3,1)
= (12/16) * 0.9799 + (4/16) * 0.81128 = 0.937745

Information Gain(IG) = E(Target) - E(Target,A) = 1- 0.9337745 = 0.062255

Information gain for Var B

Var B has value >=3 for 12 records out of 16 and 4 records with value <5 value.

- For Var B >= 3 & class == positive: 8/12
- For Var B >= 3 & class == negative: 4/12
  - Entropy(8,4) = -1 * ((8/12)*log2(8/12) + (4/12)*log2(4/12)) = 0.39054
- For VarB <3 & class == positive: 0/4
- For Var B <3 & class == negative: 4/4
  - Entropy(0,4) = -1 * ((0/4)*log2(0/4) + (4/4)*log2(4/4)) = 0

Entropy(Target, B) = P(>=3) * E(8,4) + P(<3) * E(0,4)
= (12/16) * 0.39054 + (4/16) * 0 = 0.292905

Information Gain(IG) = E(Target) - E(Target,B) = 1- 0.292905= 0.707095

Information gain for Var C

Var C has value >=4.2 for 6 records out of 16 and 10 records with value <4.2 value.

- For Var C >= 4.2 & class == positive: 0/6
- For Var C >= 4.2 & class == negative: 6/6
  - Entropy(0,6) = 0
- For VarC < 4.2 & class == positive: 8/10
- For Var C < 4.2 & class == negative: 2/10
  - Entropy(8,2) = 0.72193

Entropy(Target, C) = P(>=4.2) * E(0,6) + P(< 4.2) * E(8,2)
= (6/16) * 0 + (10/16) * 0.72193 = 0.4512

Information Gain(IG) = E(Target) - E(Target,C) = 1- 0.4512= 0.5488

Information gain for Var D

Var D has value >=1.4 for 5 records out of 16 and 11 records with value <5 value.

- For Var D >= 1.4 & class == positive: 0/5
- For Var D >= 1.4 & class == negative: 5/5
  - Entropy(0,5) = 0
- For Var D < 1.4 & class == positive: 8/11
- For Var D < 14 & class == negative: 3/11
  - Entropy(8,3) = -1 * ((8/11)*log2(8/11) + (3/11)*log2(3/11)) = 0.84532

Entropy(Target, D) = P(>=1.4) * E(0,5) + P(< 1.4) * E(8,3)
= 5/16 * 0 + (11/16) * 0.84532 = 0.5811575

Information Gain(IG) = E(Target) - E(Target,D) = 1- 0.5811575 = 0.41189

		Target	
		Positive	Negative
A	>= 5.0	5	7
	<5	3	1
Information Gain of A = 0.062255			

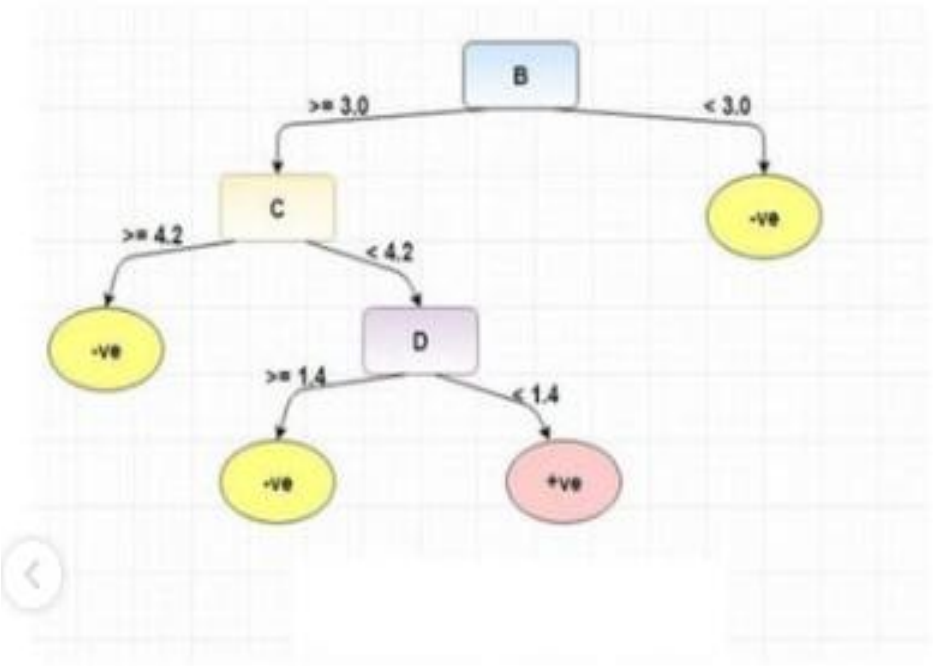
		Target	
		Positive	Negative
B	>= 3.0	8	4
	< 3.0	0	4
Information Gain of B= 0.7070795			

		Target	
		Positive	Negative
C	>= 4.2	0	6
	< 4.2	8	2
Information Gain of C= 0.5488			

		Target	
		Positive	Negative
D	>= 1.4	0	5
	< 1.4	8	3
Information Gain of D= 0.41189			

From the above Information Gain calculations, we can build a decision tree. We should place the attributes on the tree according to their values.

An Attribute with better value than other should position as root and A branch with entropy 0 should be converted to a leaf node. A branch with entropy more than 0 needs further splitting.



	A	B	C	D	E
1	4.8	3.4	1.9	0.2	positive
2	5	3	1.5	0.2	positive
3	5	3.4	1.5	0.4	positive
4	5.2	3.5	1.5	0.2	positive
5	5.2	3.4	1.4	0.2	positive
6	4.7	3.2	1.5	0.2	positive
7	4.8	3.1	1.5	0.2	positive
8	5.4	3.4	1.5	0.4	positive
9	7	3.2	4.7	1.4	negative
10	6.4	3.2	4.5	1.5	negative
11	6.9	3.1	4.9	1.5	negative
12	5.5	2.3	4	1.3	negative
13	6.5	2.8	4.6	1.5	negative
14	5.7	2.8	4.5	1.3	negative
15	6.3	3.3	4.7	1.6	negative
16	4.9	2.4	3.3	1	negative

We are going to use same data sample that we used for information gain example. Let's try to use gini index as a criterion. Here, we have 5 columns out of which 4 columns have continuous data and 5th column consists of class labels.

A, B, C, D attributes can be considered as predictors and E column class labels can be considered as a target variable. For constructing a decision tree from this data, we have to convert continuous data into categorical data.

We have chosen some random values to categorize each attribute:

A	B	C	D
>= 5	>= 3.0	>=4.2	>= 1.4
< 5	< 3.0	< 4.2	< 1.4

Gini Index

Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified. It means an attribute with lower gini index should be preferred.

Example: Construct a Decision Tree by using "gini index" as a criterion

Gini Index for Var A

Var A has value >=5 for 12 records out of 16 and 4 records with value <5 value.

- For Var A >= 5 & class == positive: 5/12
- For Var A >= 5 & class == negative: 7/12
  - gini(5,7) = 1- ((5/12)2 + (7/12)2) = 0.4860
- For Var A <5 & class == positive: 3/4
- For Var A <5 & class == negative: 1/4
  - gini(3,1) = 1- ((3/4)2 + (1/4)2) = 0.375

By adding weight and sum each of the gini indices:

gini(Target, A) = (12/16) * (0.486) + (4/16) * (0.375) = 0.45825

Gini Index for Var B

Var B has value >=3 for 12 records out of 16 and 4 records with value <5 value.

- For Var B >= 3 & class == positive: 8/12
- For Var B >= 3 & class == negative: 4/12
  - gini(8,4) = 1- ((8/12)2 + (4/12)2) = 0.446
- For Var B <3 & class == positive: 0/4
- For Var B <3 & class == negative: 4/4
  - gini(0,4) = 1- ((0/4)2 + (4/4)2) = 0

gini(Target, B) = (12/16) * 0.446 + (4/16) * 0= 0.3345

Gini Index for Var C

Var C has value >=4.2 for 6 records out of 16 and 10 records with value <4.2 value.

- For Var C >= 4.2 & class == positive: 0/6
- For Var C >= 4.2 & class == negative: 6/6
  - gini(0,6) = 1- ((0/8)2 + (6/8)2) = 0
- For Var C < 4.2& class == positive: 8/10
- For Var C < 4.2 & class == negative: 2/10
  - gini(8,2) = 1- ((8/10)2 + (2/10)2) = 0.32

gini(Target, C) = (6/16) * 0+ (10/16) * 0.32 = 0.2

Gini Index for Var D

Var D has value >=1.4 for 5 records out of 16 and 11 records with value <1.4 value.

- For Var D >= 1.4 & class == positive: 0/5
- For Var D >= 1.4 & class == negative: 5/5
  - gini(0,5) = 1- ((0/5)2 + (5/5)2) = 0
- For Var D < 1.4 & class == positive: 8/11
- For Var D < 1.4 & class == negative: 3/11
  - gini(8,3) = 1- ((8/11)2 + (3/11)2) = 0.397

gini(Target, D) = (5/16) * 0+ (11/16) * 0.397 = 0.273

		wTarget	
		Positive	Negative
A	>= 5.0	5	7

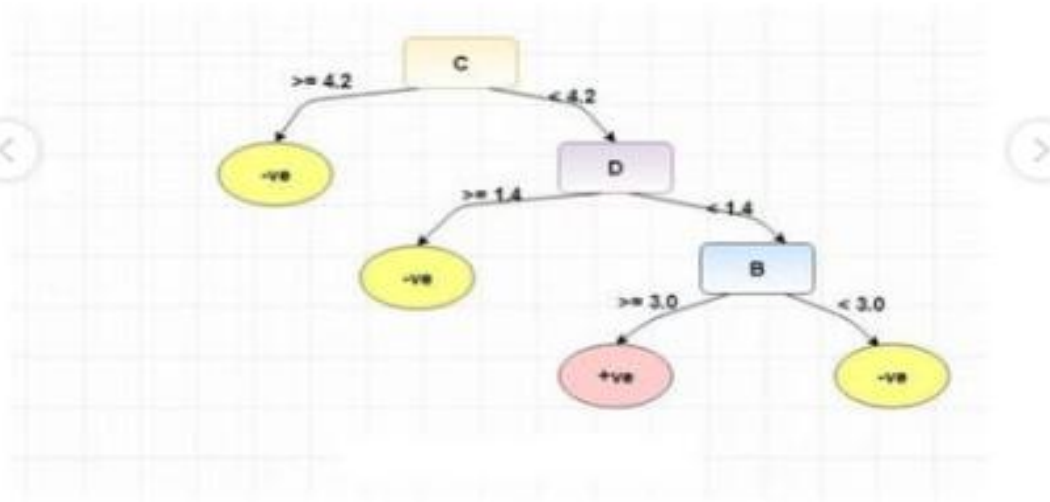
		Target	
		Positive	Negative
B	>= 3.0	8	4
	< 3.0	0	4

	<5	3	1
Ginin Index of A = 0.45825			

Gini Index of B= 0.3345

		Target	
		Positive	Negative
C	>= 4.2	0	6
	< 4.2	8	2
Gini Index of C= 0.2			

		Target	
		Positive	Negative
D	>= 1.4	0	5
	< 1.4	8	3
Gini Index of D= 0.273			



Overfitting

Overfitting is a practical problem while building a decision tree model. The model is having an issue of overfitting is considered when the algorithm continues to go deeper and deeper in the to reduce the training set error but results with an increased test set error i.e, Accuracy of prediction for our model goes down. It generally happens when it builds many branches due to outliers and irregularities in data.

Two approaches which we can use to avoid overfitting are:

- Pre-Pruning
- Post-Pruning

Pre-Pruning

In pre-pruning, it stops the tree construction bit early. It is preferred not to split a node if its goodness measure is below a threshold value. But it's difficult to choose an appropriate stopping point.

Post-Pruning

In post-pruning first, it goes deeper and deeper in the tree to build a complete tree. If the tree shows the overfitting problem then pruning is done as a post-pruning step. We use a cross-validation data to check the effect of our pruning. Using cross-validation data, it tests whether expanding a node will make an improvement or not.

If it shows an improvement, then we can continue by expanding that node. But if it shows a reduction in accuracy then it should not be expanded i.e, the node should be converted to a leaf node.

Decision Tree Algorithm Advantages and Disadvantages

Advantages:

1. Decision Trees are easy to explain. It results in a set of rules.
2. It follows the same approach as humans generally follow while making decisions.
3. Interpretation of a complex Decision Tree model can be simplified by its visualizations. Even a naive person can understand logic.
4. The Number of hyper-parameters to be tuned is almost null.

Disadvantages:

1. There is a high probability of overfitting in Decision Tree.
2. Generally, it gives low prediction accuracy for a dataset as compared to other machine learning algorithms.
3. Information gain in a decision tree with categorical variables gives a biased response for attributes with greater no. of categories.
4. Calculations can become complex when there are many class labels.

Naive Bayes Algorithm

- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

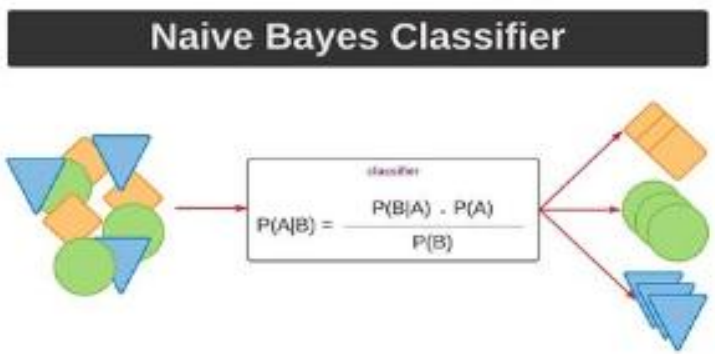
Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.



Naïve Bayes Classifier Algorithm

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.
- Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.

Working of Naïve Bayes' Classifier:

Working of Naïve Bayes' Classifier can be understood with the help of the below example:

Suppose we have a dataset of **weather conditions** and corresponding target variable "**Play**". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So to solve this problem, we need to follow the below steps:

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability.

Problem: If the weather is sunny, then the Player should play or not?

Solution: To solve this, first consider the below dataset:

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Frequency table for the Weather Conditions:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	4

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14=0.35
Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

Applying Bayes'theorem:

P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)/P(Sunny)

P(Sunny|Yes)= 3/10= 0.3

P(Sunny)= 0.35

P(Yes)=0.71

So P(Yes|Sunny) = 0.3*0.71/0.35= **0.60**

P(No|Sunny)= P(Sunny|No)*P(No)/P(Sunny)

P(Sunny|NO)= 2/4=0.5

P(No)= 0.29

P(Sunny)= 0.35

So P(No|Sunny)= 0.5*0.29/0.35 = **0.41**

So as we can see from the above calculation that **P(Yes|Sunny)>P(No|Sunny)**

Hence on a Sunny day, Player can play the game.

Advantages of Naïve Bayes Classifier:

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- It can be used for Binary as well as Multi-class Classifications.
- It performs well in Multi-class predictions as compared to the other Algorithms.
- It is the most popular choice for **text classification problems**.

Disadvantages of Naïve Bayes Classifier:

- Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features.

Applications of Naïve Bayes Classifier:

- It is used for **Credit Scoring**.
- It is used in **medical data classification**.
- It can be used in **real-time predictions** because Naïve Bayes Classifier is an eager learner.
- It is used in Text classification such as **Spam filtering** and **Sentiment analysis**.

Types of Naïve Bayes Model:

There are three types of Naive Bayes Model, which are given below:

- **Gaussian**: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
- **Multinomial**: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education, etc.
  The classifier uses the frequency of words for the predictors.
- **Bernoulli**: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.

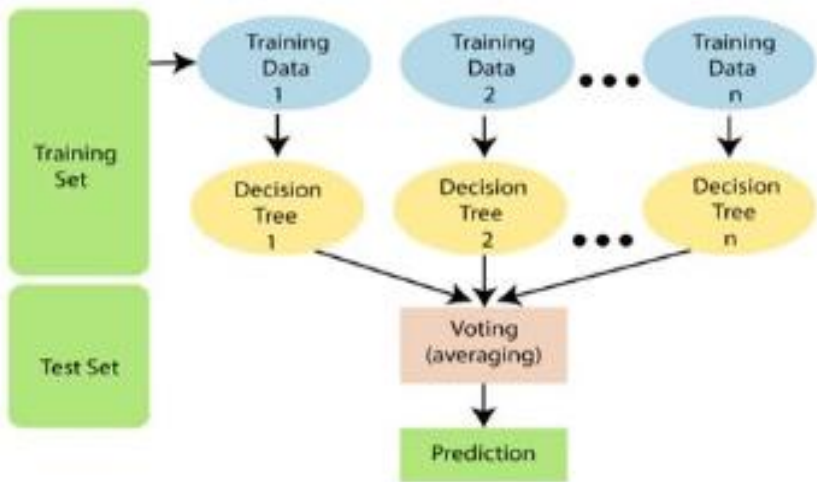
Random Forest Algorithm in Machine Learning

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, ***"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."*** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:



Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Random Forest algorithm

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

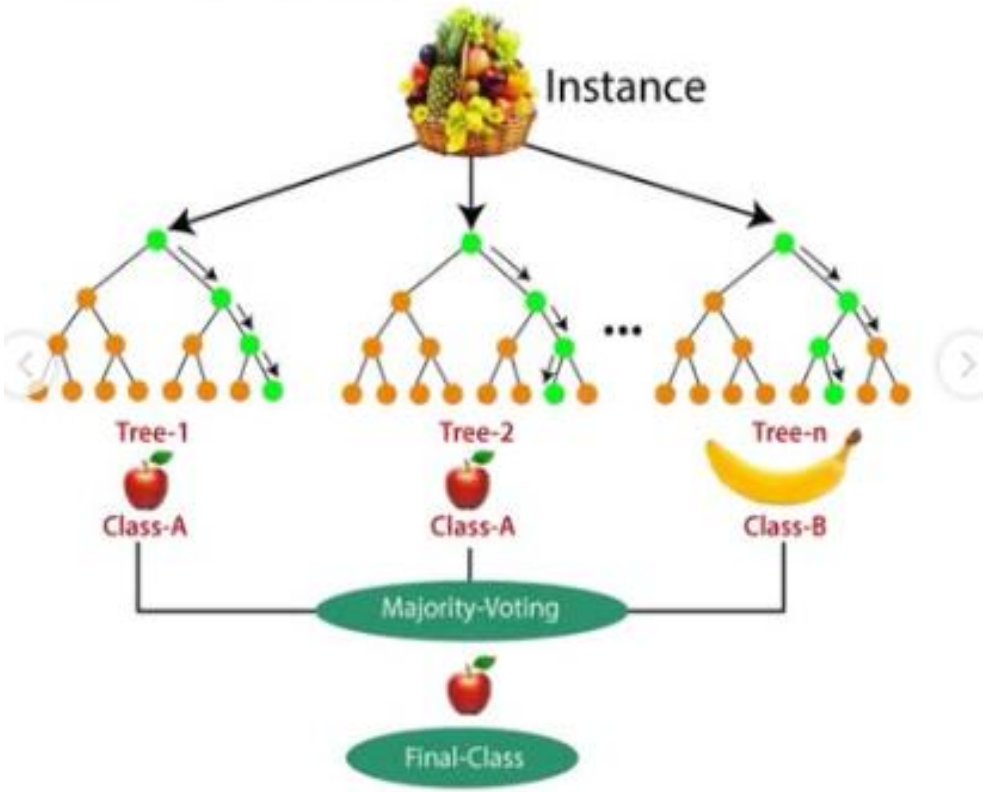
Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

The working of the algorithm can be better understood by the below example:

Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:



Applications of Random Forest

There are mainly four sectors where Random forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

Advantages & Disadvantages

Advantages of Random Forest

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages of Random Forest

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.