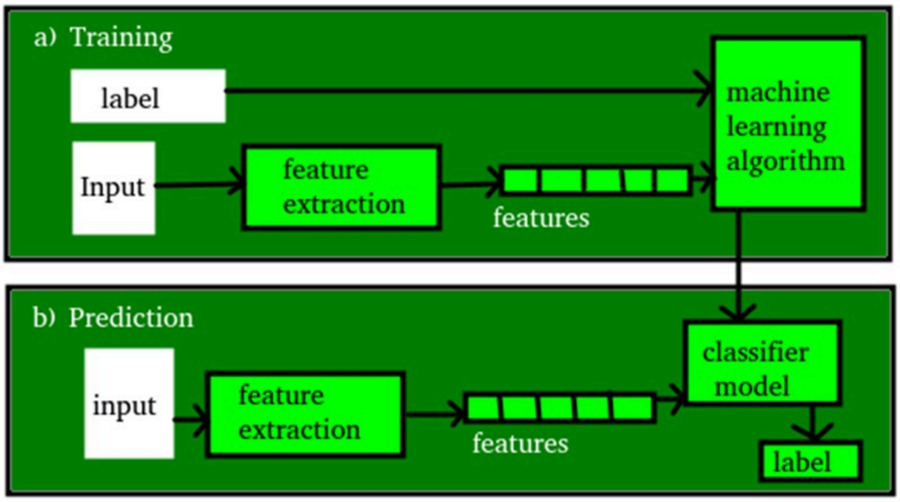
Introduction to Machine Learning

Chapter 1: Introduction

Basic Terms

- `Machine Learning` is a subset of artificial intelligence (AI) that focuses on the development of algorithms and statistical models that enable computers to perform tasks without explicit programming instructions. It's concerned with the creation of systems that can automatically learn and improve from experience or data, rather than relying solely on human intervention or predefined rules.
- `Machine Learning Program` a computer program which learns from experience.
- `Model (Hypothesis)` : a specific representation learned from data by applying some machine learning algorithim.
- `Feature` :
    - An individual measurable property of our data.
    - A set of numeric feature that can be fed as input to a model cann be convinently described by a `feature vector` .
- `Target (Label)` : a value to be predicted.
- `Training` : giving a set of inputs(features) and its expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- `Prediction` : the process providing predicted output(label) based on the input provided.



Features of Machine Learning

- Ability to Learn from Data,
- Adaptability and Flexibility,

- Scalability,
- Pattern Recognition,
- Continuous Improvement,
- Handling Complex and Nonlinear Relationships,
- Data-Driven Decisions,
- Multidomain Applicability,
- Handling Multidimensional and Multivariable Data,
- Real-Time Processing and Prediction.

Importance of Machine Learning

- Automation of Tasks
- Data-driven insights
- Handling Large Volumes of Data
- Predictive Analysis
- Enhanced Customer Experience
- Healthcare advancements
- Enhanced Accuracy and Efficiency
- Fraud detection and Cybersecurity
- Autonomous Systems
- Adaptability
- Solving Complex Problems
- Scientific Discoveries
- Continous Improvement
- Improved Decision Making
- Cost-Effectiveness, Enabling Innovation

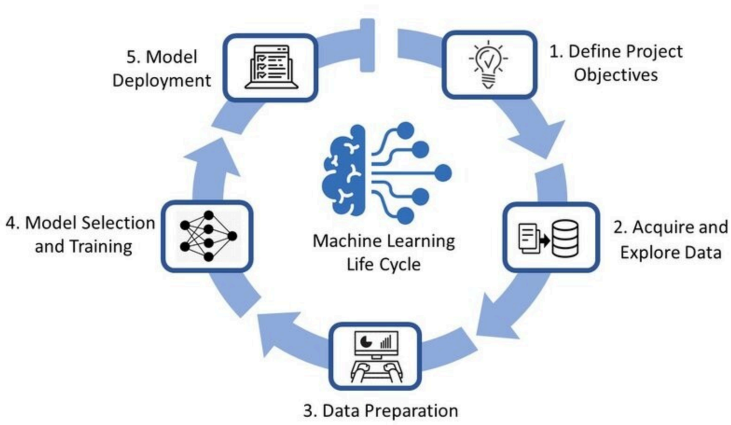
Applications of Machine Learning

- Predictive Maintenance
- Quality Control and Defect Detection
- Supply Chain Optimization
- Energy Management
- Structural Health Monitoring
- Smart Gruds and Power Systems
- Health Monitoring and Biomedical Engineering
- Environmental Monitoring
- Process Optimization

Machine Learning Lifecycle

- **Define the Problem**: identifying the problem by understanding the business problem and defining the objectives of the model.
- **Collect Data**: gather and clean the data.
- **Explore the Data**: using data visualization and statstical methods to understand the strucutre and the relationships within the data.

- **Pre-process the Data**: prepare the data for modeling by normalizing, transforming, and cleaning it as neccessary.
- **Split the Data**:
  - Divide the data into training and test datasets to validate the model.
  - Steps:
    - `Import the dataset` : load the dataset into machine learning environment.
    - `Divide the dataset` : split the dataset into `training` and `test` set.
    - `Randomization (Optional)` : before splitting, it's often a good practice to shuffle your dataset to ensure that the data points are not ordered in any particular way. This helps to prevent any bias that might arise if the data has a specific order.
    - `Split the Dataset` : Decide on a ratio for splitting your dataset into training and test sets. A common ratio is 70-30 or 80-20, where 70% or 80% of the data is used for training and the remaining 30% or 20% is used for testing. You can also use cross-validation techniques for more robust evaluation.
- **Choose a Model**: select a machine learning model that is appropriate for the problem and the data that's collected. This step requies knowledge of the strengths and weaknesses of different algorithms. Sometimes we use multiple models and compare their results and select the best model as per our requirements.
- **Train the Model**: using training data to train the model, adjusting it's parameters to fit the data as accurately as possible.
- **Evaluate the Model**: using the test data to evaluate the performance the model and determine its accuracy by using different techniques like `classification report` , `F1 score` , `precision` , `recall` , `ROC Curve` , `Mean Square error` , and `absolute error` .
- **Fine-tune the Model**: based on the results of the evaluation, fine-tune the model by adjusting its parameters and repeating the training process until the desired level of accuracy is achieved.
- **Deploy the Model**: integrating the model into application, system and making it avaliable for users.
- **Monitor the Model**: continously monitoring the performance of the model to ensure that it continues to provide accurate results over time.



Machine Learning Problems

- Machine learning encompasses a wide range of problems, datasets, and tools that are used for various applications:

- **Classification**: Assigning categories or labels to input data based on past observations. `Example` : Spam email detection.
    - **Regression**: Predicting continuous values based on input features. `Example` : Predicting house prices.
    - **Clustering**: Grouping similar data points together based on their characteristics. `Example` : Customer segmentation.
    - **Anomaly Detection**: Identifying outliers or unusual patterns in data. `Example` : Fraud detection.
    - **Dimensionality Reduction**: Reducing the number of input features while preserving important information. `Example` : Principal Component Analysis (PCA).
    - **Recommendation Systems**: Predicting user preferences or recommendations based on historical data. `Example` : Movie or product recommendations.
    - **Natural Language Processing (NLP)**: Analyzing and understanding human language data. `Example` : Sentiment analysis.
    - **Time Series Forecasting**: Predicting future values based on historical time stamped data. `Example` : Stock price prediction.
    - **Reinforcement Learning**: Teaching agents to take actions in an environment to maximize rewards. `Example` : Game playing agents.

Data in Machine Learning

- Types of data based on the presence or absence of target output (labels) associated with each data point:
    - `Labeled data` includes both input (features) and output (target labels), making it suitable for supervised learning tasks.
    - `Unlabeled data` includes only input (features) and is used in unsupervised or semi-supervised learning tasks.
- Types of data based on its role in the model development and evaluation process:
    - `Training data` : part of data that's used to train the model.
    - `Validation data` : part of data that is used to a frequent evaluation of the model, fit on the training dataset along with improving involved hyperparameters (initially set parameters before the model begins learning). This data plays its part when the model is actually training.
    - `Test data` : part of data that's used to compare with the output (labels) of model in order to evaluate its accuracy.

Dataset Examples

- `MNIST` , `CIFAR-10/100` , `UCI Machine Learning Repository` , `Titanic Dataset` , `Enron Email Dataset` , `Boston Housing Dataset` .
- `Kaggle` : widely used open source dataset platform.

Maching Learning Tools & Frameworks

Tool	Description
Scikit-learn	A popular Python library for machine learning tasks such as classification, regression, clustering, and dimensionality reduction.
TensorFlow	An open-source machine learning framework developed by Google for building and training neural networks.
PyTorch	A deep learning library developed by Facebook's AI Research lab, known for its dynamic computation graph.
Keras	A high-level neural networks API written in Python, compatible with TensorFlow, Theano, and Microsoft Cognitive Toolkit.
Pandas	A Python library for data manipulation and analysis, widely used for data preprocessing and exploration.
NumPy	A fundamental package for scientific computing in Python, providing support for multidimensional arrays and matrices.
Matplotlib and Seaborn	Python libraries for data visualization, used for creating plots, charts, and graphs.
Jupyter Notebooks	An interactive computing environment that allows users to create and share documents containing live code, visualizations, and narrative text.
XGBoost and LightGBM	Libraries for gradient boosting, commonly used for classification and regression tasks.
NLTK (Natural Language Toolkit)	A Python library for NLP tasks such as tokenization, stemming, and parsing.

Types of Machine Learning

- **Supervised Learning**:
  - Algorithm is on labeled dataset.
  - It learns to map input features to targets based on labeled training data.
  - It's provided with input and corresponding output labels to generalize from this data to make predictions on new unseen data.
  - The training process continues until the model achieves the desired level of accuracy on the training data.
  - Data is usually split in the ratio of 80:20 i.e. 80% as training data and the rest as testing data.
  - **Examples**:
    - `Image Classification` : You train with images/labels. Then in the future, you give a new image expecting that the computer will recognize the new object.

        - `Market Prediction/Regression` : You train the computer with historical market data and ask the computer to predict the new price in the future.
    - Use cases:
        - `Classification` :
            - Inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes and predicts whether or not something belongs to a particular class.
            - Categories:
                - Binary Classification
                - Multiclass Classification
            - **Example**: Spam filtering
        - `Regression` : predicts a numeric value and outputs are continuous rather than discrete. Example can be a model that predicts a stock prices using historical data.
    - Algorithms:
        - `Linear Regression` : type of regression algorithm that tries to find a linear realtionship between the input features and the output value. Output is predicted based on the weighted sum of the input features.
        - `Logistic Regression` : type of classification algorithm that is used to predict a binary output variable.
        - `Decision Trees` :
            - Algorithm that is used for both classification and regression tasks.
            - It is a tree-like structure that is used to model decisions and their possible consequences.
            - Each internal node in the tree represents a decision, while each leaf node represents a possible outcome.
        - `Random Forests` :
            - An ensemble learning technique that is used for both classification and regression tasks.
            - They are made up of multiple decision trees that work together to make predictions. Each tree in the forest is trained on a different subset of the input features and data.
            - The final prediction is made by aggregating the predictions of all the trees in the forest.
        - `SVM`
- **Semi-Supervised Learning**:
    - In a large amount of input data only some of the data is labeled.
    - **Examples**:
        - Image classification with limited labeled data
        - Sentiment analysis on social media
        - Spam filtering in email
        - Anomaly detection in industrial processes
    - Algorithms and Techniques:
        - `Self-training (Self-labeling)` :

    - A simple yet effective method where a model is initially trained with a small amount of labeled data. Then, the model is used to predict labels for the unlabeled data.
    - Some of the these predictions with high confidence are added to the training set as if they were true labels and the model is retrained.
  - `Co-training`:
    - Two models are trained separately on the same labeled data but with different views (set of features).
    - Each model then labels unlabeled data for the other model to use in further training.
    - This method assumes that the two views are conditionally independent given the class label and that each view is sufficient for classification.
  - `Semi-supervised Support Vector Machines (S3VMs)`:
    - Traditional SVMs are extended for semi-supervised learning by finding a decision boundary that not only separates the labeled data points but also correctly classifies the unlabeled points.
    - One common approach is to treat the labels of the unlabeled data as variables and iteratively adjust them along with the decision boundary.
  - `Graph-based methods`:
    - Construct a graph where nodes represent both labeled and unlabeled data points, and edges represent the similarity between points.
    - The goal is to propagate labels from the labeled points to the unlabeled points through the graph, based on similarity.
    - `Label propagation` and `label spreading` are examples of graph-based semi-supervised learning algorithms.
  - `Deep learning approaches`:
    - `Pseudo labeling`
    - `Consistency regularization` encourages the model to produce the same output for an unlabeled example even after it has been perturbed in some way (e.g., adding noise).
    - `Variational Autoencoders (VAEs)`
    - `Generative Adversarial Networks (GANs)`
- **Unsupervised Learning**:
  - Type of machine learning technique where algorithms learn from data without any pre-defined labels or categories.
  - Features:
    - `Identify hidden structures`: they can group similar data points together, uncovering underlying relationships and patterns in the data. This is commonly used for tasks like `customer segmentation` or `image clustering`.
    - `Extract features`: they can identify important characteristics within the data, which can then be used for further analysis or building other machine learning models.
    - `Reduce complexity`: they can simplify complex datasets by reducing the number of dimensions, making them easier to analyze and visualize.
  - Algorithms:

        - `Clustering` :
            - Algorithms aims to group similar data points together based on their characteristics.
            - **Examples**:
                - `K-means clustering` : partitions data into a predefined number of clusters (k).
                - `Hierarchical clustering` : builds a hierarchy of clusters, allowing for a flexible number of clusters.
        - `Dimensionality reduction` :
            - Algorithms aim to reduce the number of features in a dataset while retaining the most important information.
            - **Example**:
                - `Principal Component Analysis (PCA)` : identifies the most significant features that explain most of the data's variance.
        - `Anomaly detection` :
            - Algorithms identify data points that deviate significantly from the expected patterns, potentially indicating anomalies or outliers.
            - **Example**:
                - `Local Outlier Factor (LOF)` : identifies data points that have a significantly lower density of neighbors compared to surrounding points.
        - `Association rule learning` :
            - Discovers relationships between items or events in a dataset.
            - **Example**:
                - `Apriori algorithm` : identifies frequent itemsets and generates association rules based on their co-occurrence.
        - `Latent variable models` :
            - Aim to uncover hidden variables that explain the observed data.
            - **Example**:
                - `Autoencoders` : learn compressed representations of the data by encoding the data into a lower-dimensional space and then reconstructing it from that space.
    - Use cases:
        - `Clustering` : computer separates similar data into clusters which is essential in research and science.
        - `Density estimation` : finding the distribution of inputs in some space.
        - `High-Dimension Visualization` : visualizing high-dimension data.
        - `Generative Models` : After a model captures the probability distribution of your input data, it will be able to generate more data. This can be very useful to make your classifier more robust.
    - Examples:
        - Anomaly detection in network traffic
        - Fraud detection in financial transactions
        - Content filtering and moderation
        - Personalization in real-time applications

Commonly used Algorithms, Libraries, Languages and Tools in Machine Learning

- Algorithms:
  - Decision Trees
  - Bayesian Classifiers
  - Neural Networks
  - K-Nearest Neighbour
  - Support Vector Machines
  - Linear Regression
  - Logistic Regression
  - Dimensionality reduction algorithms
  - Gradient boosting algorithm and AdaBoosting algorithm
- Associated Tools and Languages:
  - Used to mine/ extract useful information from raw data.
  - Libraries used: `Jupyter` , `NumPy` , `Matplotlib` , `Pandas` , `ScikitLearn` , `NLTK` , `TensorFlow` , `Seaborn` , `Basemap`
  - Main Languages used: `R` , `SAS` , `Python` , `SQL`
  - Major Tools used: `RapidMiner` , `Orange` , `KNIME` , `Spark` , `Weka`

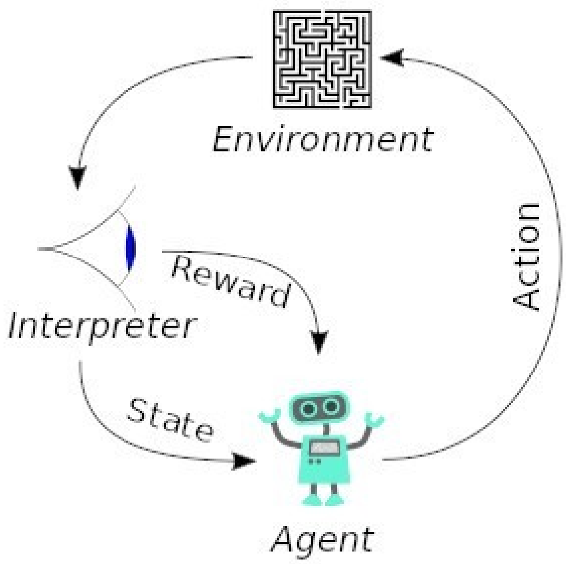
Supervised Learning vs Unsupervised Learning

Parameters	Supervised machine learning	Unsupervised machine learning
Input Data	Algorithms are trained using labeled data.	Algorithms are used against data that is not labeled
Computational Complexity	Simpler method	Computationally complex
Accuracy	Highly accurate	Less accurate
No. of classes	No. of classes is known	No. of classes is not known
Data Analysis	Uses offline analysis	Uses real-time analysis of data
Algorithms used	Linear and Logistics regression, Random forest, Support Vector Machine, Neural Network, etc.	K-Means clustering, Hierarchical clustering, Apriori algorithm, etc.
Output	Desired output is given.	Desired output is not given.
Training data	Use training data to infer model.	No training data is used.
Complex model	It is not possible to learn larger and more complex models than with supervised learning	It is possible to learn larger and more complex models with unsupervised learning

Reinforcement Learning

- A type of machine learning technique where an agent learns to make decisions by interacting with an environment in order to achieve a specific goal.
- The agent learns through trial and error, receiving feedback in the form of rewards or punishments for its actions.
- After each action the algorithm receives feedback that helps it determine whether the choice it made is `correct` , `neutral` , or `incorrect` .
- The goal of the agent is to maximize the total cumulative reward it receives over time.

- An autonomous and self teaching system.
- **Terms**:
  - `Input` : an initial state from which the model will start.
  - `Output` : possible outputs as there are a variety of solutions to a particular problem.
  - `Training` : based upon the input, The model will return a state and the user will decide to reward or punish the model based on its output.
- **How it works**:
  - `Agent` : The entity that learns to make decisions based on its interactions with the environment.
  - `Environment` : The external system with which the agent interacts. It provides feedback to the agent based on its actions.
  - `State` : The current situation or configuration of the environment.
  - `Action` : The decision or choice made by the agent in a given state.
  - `Reward` : Feedback from the environment to the agent indicating the desirability of the action taken in a particular state.



- **Example**:
  - Consider a robot places in a maze , and your goal is to teach the robot to find its way out of the maze using reinforcement learning:
    - `Agent` : The robot.
    - `Environment` : The maze.
    - `State` : The robot's current position in the maze.
    - `Action` : Moving in one of the available directions (up, down, left, right).
    - `Reward` : +1 for reaching the exit, -1 for hitting a wall, 0 otherwise.
    - Process:
      - `Initialization` : The robot starts in a random position within the maze.
      - `Action Selection` : The robot selects an action (e.g., move up).
      - `Environment Interaction` : The robot moves according to the chosen action, and its position in the maze changes.
      - `Reward Assignment` : The robot receives a reward based on its new position in the maze.

            - `Learning` : The robot updates its policy (strategy for selecting actions) based on the received reward and its past experiences.
            - `Repeat` : Steps 2-5 are repeated until the robot consistently finds the exit.
- **Types**:
    - `Positive Reinforcement` is defined as when an event, occurs due to a particular behavior, increases the strength and the frequency of the behavior. In other words, it has a positive effect on behavior.
        - Advantages:
            - Maximizes Performance.
            - Sustain Change for a long period of time
        - Disadvantage: too much Reinforcement can lead to an overload of states which can diminish the results
    - `Negative Reinforcement` is defined as strengthening of behavior because a negative condition is stopped or avoided.
        - Advantages:
            - Increases Behavior
            - Provide defiance to a minimum standard of performance
        - Disadvantage: only provides enough to meet up the minimum behavior
- **Applications**:
    - Robotics
    - Game Playing
    - Adaptive control devices
    - Autonomous Driving
    - Finance
    - Healthcare
    - Resource Management
    - NLP
- **Advantages**:
    - Can be used to solve very complex problems.
    - The model can correct the errors that occurred during the training process.
    - Training data is obtained via the direct interaction of the agent with the environment.
    - Can handle environments that are non-deterministic.
    - Can be used to solve a wide range of problems, including those that involve decision making, control, and optimization.
    - Flexible approach that can be combined with other machine learning techniques, such as deep learning, to improve performance.
- **Disadvantages**:
    - Not preferable to use for solving simple problems.
    - Needs a lot of data and a lot of computation.
    - Highly dependent on the quality of the reward function. If the reward function is poorly designed, the agent may not learn the desired behavior.

- Difficult to debug and interpret.

Reinforcement Learning vs Supervised Learning

Reinforcement Learning	Supervised Learning
The output depends on the state of the current input and the next input depends on the output of the previous input (sequential)	Decision is made on the initial input or the input given at the start
Decision is dependent, So we give labels to sequences of dependent decisions	Decisions are independent of each other so labels are given to each decision
`Example` : Chess game, text summarization	`Example` : Object recognition,spam detetction

Deep Learning

- A subset of machine learning that utilizes artificial neural networks with multiple layers.
- These neural networks are designed to automatically learn and extract hierarchical representations of data, leading to increasingly abstract and complex features as you move deeper into the network.
- **Key Components**:
  - Artificial Neural Networks (ANNs):
    - Computational models inspired by the structure and function of biological neurons in the human brain.
    - Consist of interconnected nodes organized into layers, including an input layer, one or more hidden layers, and an output layer.
  - Deep Architectures:
    - Multiple layers allowing them to learn intricate patterns and representations from data.
    - These architectures can range from a few layers to dozens or even hundreds of layers, depending on the complexity of the task and the amount of data available.
  - Feature Learning:
    - Algorithms automatically learn to extract relevant features from raw data, eliminating the need for manual feature engineering.
    - This ability to learn hierarchical representations of data is one of the key strengths of deep learning, enabling it to effectively handle complex and high dimensional data.
  - Training with Backpropagation:
    - Deep learning models are trained using optimization algorithms such as `gradient descent` and `backpropagation` .
    - During the training process, the model adjusts its parameters (weights and biases) based on the discrepancy between its predictions and the ground truth labels in the training data, gradually minimizing the loss function.

    - Big Data and Computational Power:
        - Deep learning algorithms often require large amounts of data to train effectively, and they also benefit from significant computational power for training complex models with millions or even billions of parameters.
        - Advances in data collection, storage, and processing, as well as the availability of powerful hardware (e.g., GPUs, TPUs), have played a crucial role in the success of deep learning.
- Popular deep learning architectures include `Convolutional Neural Networks (CNNs)` for `image recognition`, `Recurrent Neural Networks (RNNs)` for `sequence modeling`, and `Transformers` for `natural language processing`.

- **How does it work**:

    - Deep learning models work through a combination of `architecture`, `training`, and `prediction`:
    - Architecture:
        - ANNs consist of interconnected layers of artificial neurons (nodes).
        - Each layer performs specific transformations on the data, and information is passed and refined through these layers.
        - The depth (number of layers) is what distinguishes deep learning from traditional machine learning models, allowing them to learn increasingly complex features from the data.
    - Training:
        - Deep learning models are trained on large datasets of labeled data. Each data point has an associated label or category (e.g., an image labeled as "cat").
        - During training, the model adjusts the weights and biases of its connections between neurons based on the differences between the model's predictions and the actual labels. This process is called backpropagation.
        - Through repeated exposure to training data and adjustments, the model learns to identify patterns and relationships within the data, gradually improving its accuracy in predicting labels for new, unseen data.
    - Prediction:
        - New, unseen data is fed into the model, and it propagates through the layers, undergoing transformations at each layer.
        - Finally, the output layer produces a prediction, which can be a classification (e.g., classifying an image as a dog), a value (e.g., predicting the price of a house), or even a new creation (e.g., generating realistic images).
- **Key Points**:

    - Deep learning models learn from data by adjusting their internal parameters (weights and biases) through training.
    - The architecture of the neural network plays a crucial role in determining the model's ability to learn complex patterns.
    - Deep learning models can be used for various tasks, including classification, regression, and generation.

- **Examples**:
  - Image recognition
  - Natural Language Processing
  - Speech recognition
  - Recommendation systems
  - Self-driving cars

Most Popular Deep Learning Algorithms

- Convolutional Neural Networks (CNNs):
  - Excel at tasks involving grid like data, particularly images.
  - Their architecture is specifically designed to extract features from images, making them highly effective in applications like image recognition, object detection, and image classification.
- Recurrent Neural Networks (RNNs):
  - Designed to handle sequential data, where the order of elements matters.
  - They are able to process information from previous steps, making them suitable for tasks like language translation, speech recognition, and sentiment analysis.
- Long Short-Term Memory Networks (LSTMs):
  - A specific type of RNN that addresses the vanishing gradient problem, a limitation of traditional RNNs in handling long sequences.
  - LSTMs are particularly powerful for tasks involving longer sequences, such as machine translation, handwriting recognition, and video captioning.
- Generative Adversarial Networks (GANs):
  - These involve two competing neural networks: a generator that creates new data, and a discriminator that tries to distinguish the generated data from real data.
  - This competition drives both networks to improve, allowing GANs to generate highly realistic images, text, and even music.
- Autoencoders:
  - These are neural networks trained to reconstruct their input data.
  - They learn by compressing the input data into a lower-dimensional representation (bottleneck) and then attempting to reconstruct the original data from this compressed version.
  - This process helps identify important features in the data and can be used for tasks like dimensionality reduction, anomaly detection, and data compression.

Challenges and Limitations of Machine Learning

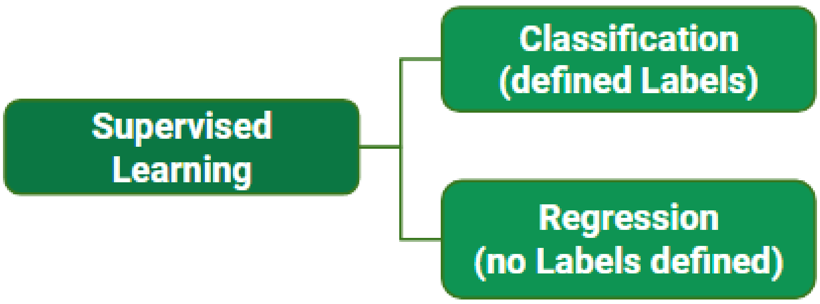
- Lack of data or diversity in the dataset.

Chapter 2: Supervised Learning

Major Objectives of Supervised Learning

- Prediction: To accurately predict the output labels for new, unseen input data based on patterns learned from labeled training examples.
- Generalization: To generalize well on unseen data by learning underlying patterns and relationships from the training data, thus making accurate predictions on new data beyond the training set.
- Optimization: To optimize the model parameters or weights based on a specified objective function (e.g., minimizing prediction errors) using optimization algorithms such as gradient descent.
- Evaluation: To evaluate the performance of the trained model on unseen data using appropriate evaluation metrics to assess its accuracy, reliability, and generalization capabilities.

Types of Supervised Learning



Linear Classification

- Assign an input feature vector to one of two or more classes using a linear decision boundary.
- It is widely used for tasks like spam detection, image classification, and medical diagnosis.
- A linear classifier uses a linear combination of input features to make decisions.
- It assumes that the data can be separated by a `linear boundary` (hyperplane in higher dimensions):
    - $f(x) = w_1x_1 + w_2x_2 + ... + w_px_p = W^TX + b$ where:
        - $x = (x_1, x_2, ..., x_p)$ is the `feature vector`
        - $w = (w_1, w_2, ..., w_p)$ are the `weights`
        - b is `the bias(intercept)`
    - $W^TX + b = 0$:
        - $W^TX + b > 0$, classify as `Class 1`.
        - $W^TX + b \leq 0$, classify as `Class 2`.

Types of Linear Classifiers

- `Perceptron:` one of the simplest types of linear classifiers, originally developed in the 1950s. It uses a step function for `binary classification`.

- `Logistic Regression` : although often considered a regression technique, Logistic Regression is commonly used for `binary classification` .
- `Support Vector Machines (SVM)` : SVM is another popular linear classifier that finds the maximum-margin hyperplane to separate classes.

Strengths and Weaknesses of Linear Classification

- Strengths:
  - Simple and easy to implement.
  - Works well for linearly separable data.
  - Computationally efficient for large datasets.
  - Extensions like kernel methods in SVM can handle non-linear problems.
- Weaknesses:
  - Assumes data is linearly separable. For complex patterns, linear classifiers may not perform well.
  - Sensitive to the scaling of features. Preprocessing such as standardization is often required.

Linear Regression

- Type of regression model where the output variable is a real or continuous value.
- It tries to fit data with the best hyperplane which goes through the points.
- Used for `predictive analysis` .
- A linear approach for modeling the relationship between the criterion or the scalar response and the multiple predictors or explanatory variables.
- Focuses on the conditional probability distribution of the response given the values of the predictors.
- $y = \theta x + b$ where:
  - θ It is the model weights or parameters(is the slope of the regression line)
  - b It is known as the bias.(b is the y-intercept. This is the value of y when x is zero.)
  - x is the independent variable. This is the variable you believe influences the dependent variable.
  - y is the dependent variable. This is the variable you are trying to predict or explain.

What's Regression Analysis?

- A statistical process for estimating the relationships between the dependent variables or criterion variables and one or more independent variables or predictors.
- It is generally used when we deal with a dataset that has the target variable in the form of continous data.
- Heavily based on statistics and hence gives quite reliable results due to this reason only regression models are used to find the linear as well as non-linear relation between the independent and the dependent or target variables.

Types of Regression Techniques

- Linear Regression

- Polynomial Regression
- Stepwise Regression
- Decision Tree Regression
- Random Forest Regression
- Support Vector Regression
- Ridge Regression
- Lasso Regression
- ElasticNet Regression
- Bayesian Linear Regression

Multivariate Regression: Polynomial Regression

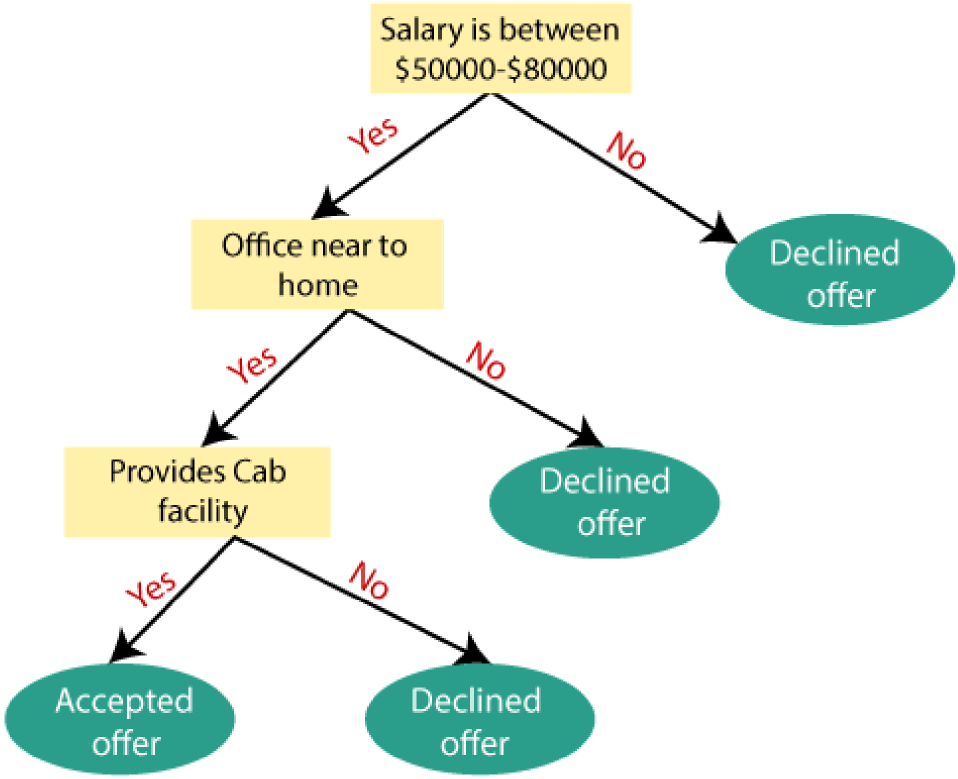
- An extension of linear regression and is used to model a non-linear relationship between the dependent variable and independent variables.
- We can easily fit some non-linear relationship between the target as well as input features.
- $y = \beta_0 + \beta_1 x + \beta_2 x^2 + ... + \beta_d x^d + \epsilon$ where:
  - y - Dependent variable (what you're trying to predict)
  - β_0 (beta-nought) - Intercept (y-value when x = 0)
  - β_i (beta-i) - Coefficients for each term (i = 1, 2, 3, …, d)
  - x - Independent variable
  - x^i - Independent variable raised to the power of i (i = 1, 2, 3, …, d) - This is where the complexity comes in compared to linear regression
  - d - Degree of the polynomial (highest power of x)
  - ϵ (epsilon) - Error term (accounts for random noise or unexplained variance)
- Key Points:
  - The coefficients (β_i) represent the weight or influence of each term on the final prediction of y.
  - The degree (d) determines the complexity of the relationship being modeled. Higher degrees can capture more intricate curves, but also risk overfitting the data.
  - Just like linear regression, the goal is to find the values for β that minimize the error term (ϵ) and provide the best fit for the data. The goal is to estimate the regression coefficients (β) that best fit the data.
- Types of Polynomial Regression:
  - The most common types are named based on their degree:
    - `Linear (d = 1)` - This is essentially standard linear regression.
    - `Quadratic (d = 2)` - Models U-shaped or inverted U-shaped relationships.
    - `Cubic (d = 3)` - Can capture more complex S-shaped curves.
    - `Higher-order polynomials (d > 3)` - Can be used for very intricate relationships, but are prone to overfitting.

Decision Tree Regression

- Most powerful and popular tool for classification and prediction.
- A flowchart like tree structure where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class

label.

- There is a non-parametric method used to model a decision tree to predict a continuous outcome.
- Handles non-linear data well.



- Key Concepts of Decision Tree Regression:
  - `Root Node` : The starting point of the tree, representing the entire dataset.
  - `Decision Nodes` : Points where the data is split into branches based on specific conditions.
  - `Leaf Nodes` : End points of the tree that provide the predicted values (output of the model).
  - `Splitting` : The process of dividing the data based on features to minimize prediction error.
  - `Recursive Partitioning` : Continuously splitting the data into subsets until a stopping condition is met (like reaching a minimum number of samples in a leaf node).

Random Forest Regression

- An Ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called `Bootstrap` and `Aggregation` , commonly known as `bagging` .
- `Ensemble` means it relies on combining the predictions of multiple models to get a more accurate final prediction.
- The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.
- We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This part is called `Bootstrap` .

Support Vector Regression (SVR)

- A type of support vector machine (SVM) that is used for regression tasks.

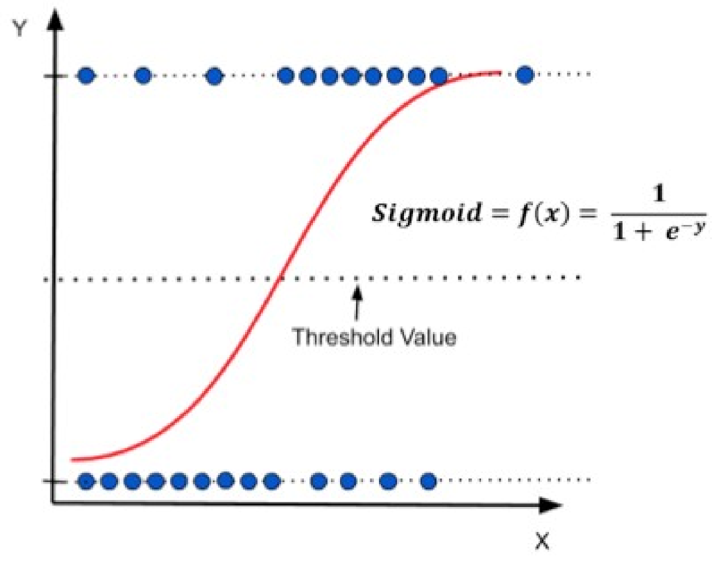
- It tries to find a function that best predicts the continuous output value for a given input value.
- SVR can use both linear and non-linear kernels.
- A linear kernel is a simple dot product between two input vectors, while a non-linear kernel is a more complex function that can capture more intricate patterns in the data. The choice of kernel depends on the data's characteristics and the task's complexity.

Gradient Boosting and Neural Network Regressions

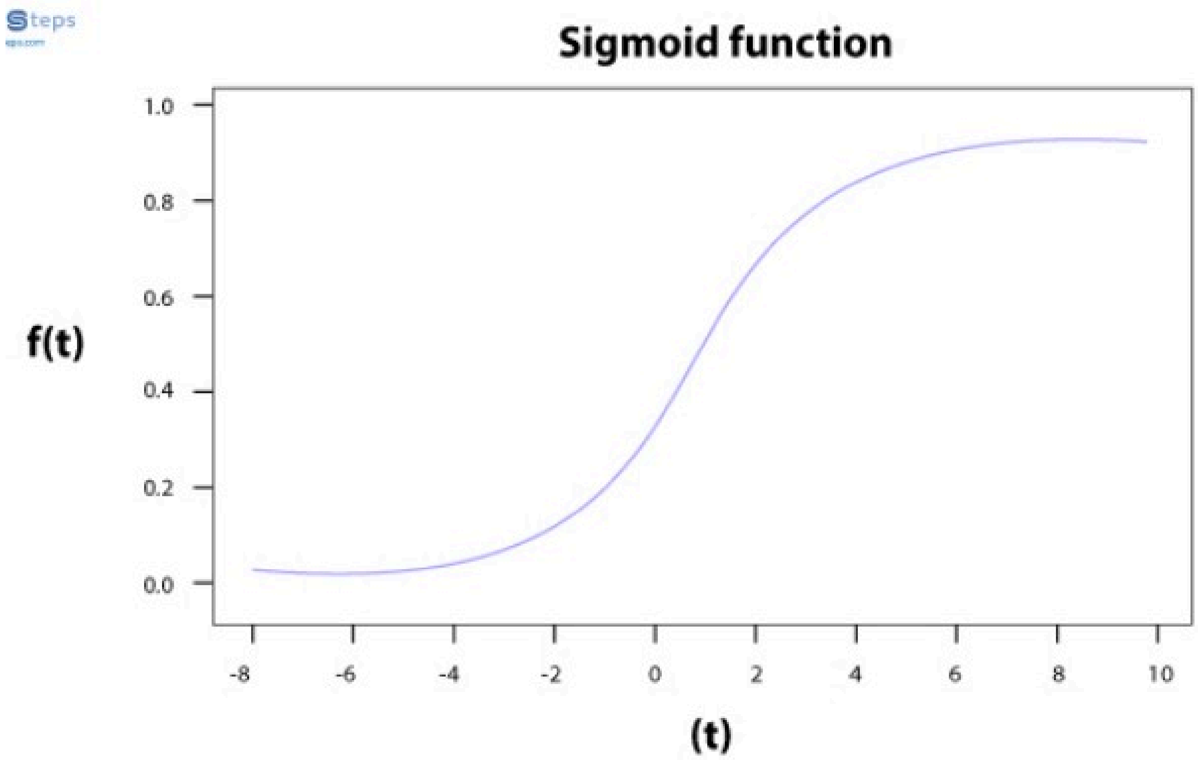
- Gradient Boosting Regression
    - `Purpose` : Another ensemble method that builds trees sequentially, with each new tree correcting the errors of the previous ones.
    - `Model` : Uses gradient descent to minimize a loss function.
    - `Use Case` : Provides highly accurate predictions for a variety of problems, though it is prone to overfitting if not tuned properly.
- Neural Network Regression
    - `Purpose` : A non-linear model inspired by the human brain, capable of learning complex patterns in data.
    - `Model` : Consists of layers of interconnected nodes (neurons), where each node performs a weighted sum of inputs and passes the result through an activation function.
    - `Use Case` : Best for large, complex datasets with non-linear relationships (e.g., image or speech data).

Logistic Regression

- A type of classification algorithm that is used to predict a binary output variable.
- It is commonly used in machine learning applications where the output variable is either true or false, such as in fraud detection or spam filtering.
- The algorithm tries to find a linear relationship between the input features and the output variable.
- The output variable is then transformed using a logistic function to produce a probability value between 0 and 1.



- Predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value.
- A significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Linear Regression and Logistic Regression are similar, but Linear Regression is used for solving `Regression problems`, whereas Logistic regression is used for solving the `classification problems`.
- Logistic Function (Sigmoid Function):
  - A mathematical function used to map the predicted values to probabilities.
  - It maps any real value into another value within a range of 0 and 1.
  - The S-form curve is called the `Sigmoid function` or the `logistic function`.



We typically denote the sigmoid function by the greek letter σ (sigma) and define as

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Where

- x is the input to the sigmoid function
- e is [Euler's number](https://en.wikipedia.org/wiki/E_(mathematical_constant) ($e = 2.781\ldots$)

- Types:
  - `Binomial`: there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.

- `Multinomial` : there can be 3 or more possible unordered types of the dependent variable, such as “cat”, “dogs”, or “sheep”.
- `Ordinal` : there can be 3 or more possible ordered types of dependent variables, such as “low”, “Medium”, or “High”.

- Steps:

    - `Define the problem` : Identify the dependent variable and independent variables and determine if the problem is a binary classification problem.
    - `Data preparation` : Clean and preprocess the data, and make sure the data is suitable for logistic regression modeling.
    - `Exploratory Data Analysis (EDA)` : Visualize the relationships between the dependent and independent variables, and identify any outliers or anomalies in the data.
    - `Feature Selection` : Choose the independent variables that have a significant relationship with the dependent variable, and remove any redundant or irrelevant features.
    - `Model Building` : Train the logistic regression model on the selected independent variables and estimate the coefficients of the model.
    - `Model Evaluation` : Evaluate the performance of the logistic regression model using appropriate metrics such as accuracy, precision, recall, F1-score, or AUC-ROC.
    - `Model improvement` : Based on the results of the evaluation, fine-tune the model by adjusting the independent variables, adding new features, or using regularization techniques to reduce overfitting.
    - `Model Deployment` : Deploy the logistic regression model in a real-world scenario and make predictions on new data.

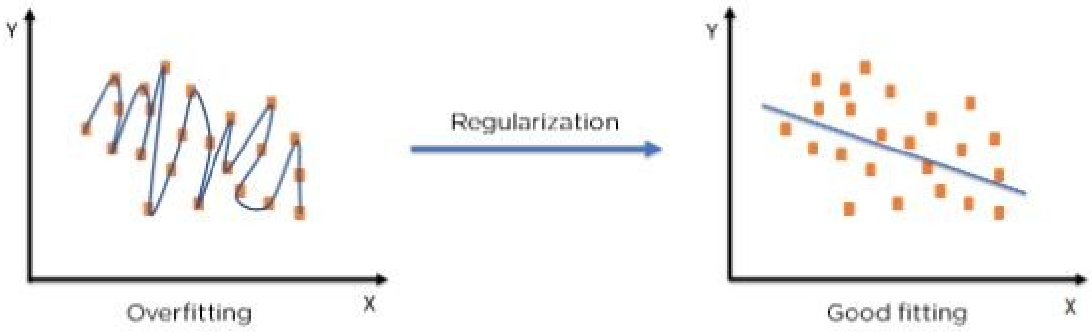
Overfitting

- When a model performs exceptionally well on the training data but fails to generalize to new, unseen data.
- This can happen when a model is too complex and has memorized the training data, rather than learning the underlying patterns.
- Indicators:
    - High Training Accuracy,
    - Low Test Accuracy,
    - High Variance,
    - Model Complexity
- Causes:
    - Limited Data,
    - Noise in Data,
    - Lack of Regularization and
    - High Model Complexity
- How to avoid Overfitting:
    - `Simplify the Model` : Use models with fewer parameters, prune decision trees, or reduce the number of features.
    - `Regularization` : Apply techniques like L1 or L2 regularization to discourage overly complex models.

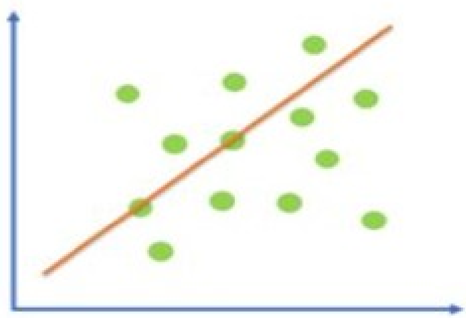
- `Cross-Validation` : Use cross-validation to test the model's performance on different subsets of data, helping to prevent it from learning specific patterns in just one subset.
- `Use Ensemble Methods` : Techniques like Random Forests and Gradient Boosting combine multiple models to balance complexity and prevent overfitting.
- `Increase Training Data` : More data can help the model generalize better by exposing it to diverse patterns.

Regularization Techniques

- Used to prevent overfitting.
- Regularization refers to techniques that are used to calibrate machine learning models in order to minimize the adjusted loss function and prevent overfitting or underfitting.



- **Types**:
  - Ridge Regularization(L2):
    - Modifies the overfitted or underfitted models by adding the penalty equivalent to the sum of the squares of the magnitude of coefficients.
    - The mathematical function representing our machine learning model is minimized and coefficients are calculated. The magnitude of coefficients is squared and added.
    - Performs regularization by shrinking the coefficients present. The function depicted below shows the cost function of ridge regression:



    - Shrinks coefficients uniformly, but it never reduces any coefficient to exactly zero. All features are retained, but their influence is reduced.
  - Lasso Regularization (L1):
    - A technique used to prevent overfitting in machine learning models, particularly linear regression models.
    - It works by adding a penalty term to the cost function that penalizes the sum of the absolute values of the model's coefficients (weights).

- This can drive some of these coefficients to zero. This effectively removes features with little improve interpretability and handle importance from the model, making it simpler and less prone to overfitting.
- The penality is the absolute value of the coefficients.
- **Benefits**:
    - `Reduces Overfitting` : The main advantage is preventing overfitting by promoting sparsity (having many coefficients equal to zero).
    - `Feature Selection` : By driving coefficients to zero, L1 can implicitly perform feature selection, highlighting the most important features for the model.
    - `Interpretability` : A model with fewer non-zero coefficients is generally easier to interpret, as you can focus on the features with the most significant impact.

Elastic Net (Hybrid) Model

- A hybrid method that combines both `Ridge` and `Lasso` penalties.
- It uses a linear combination of both L1 (lasso) and L2 (ridge) penalties.
- This method is useful when you have many correlated predictors.
- It can shrink coefficients to zero (like Lasso) and reduce the impact of multicollinearity (like Ridge).

Comparision of Methods

Method	Shrinkage Type	Feature Selection	When to Use
Ridge Regression	L2 Regularization	No	When all predictors are important but you want to prevent overfitting.
Lasso Regression	L1 Regularization	Yes	When you suspect that some predictors are irrelevant.
Elastic Net	L1 and L2	Yes	When you have many predictors and some may be correlated or irrelevant.

- If you are dealing with a `large dataset` with `many features` , `Ridge` might be appropriate if you believe all features carry some signal.
- However, if you think only a `few features` are important, `Lasso` would shrink the irrelevant features to zero.
- Elastic Net is useful when you believe some `features` are `redundant` or correlated but still want to perform feature selection.
- Each method can be fine-tuned by adjusting the regularization parameter λ (also called alpha in some implementations), which determines how much shrinkage to apply. You can use `cross-validation` to choose the best value for λ.

Factors for choosing the right technique

- The type of data (numerical, categorical, text)
- The machine learning algorithm being used
- The specific problem you're trying to solve

Application to Classification and Regression

- **Classification**: encoding categorical variables and creating interaction features can be particularly useful for models that do not inherently model non- linear relationships or interactions (e.g., logistic regression).
- **Regression**: polynomial features and binning can help capture complex, non-linear relationships between the features and the target variable.

Chapter 3: Support Vector Machines

- A powerful supervised machine learning algorithmcwidely used for both linear and nonlinear classification, as well as regression and outlier detection tasks.
- Highly adaptable, making them suitable for various applications
- Particularly effective because they focus on finding the maximum separating hyperplane between the different classes in the target feature, making them robust for both binary and multiclass classification.
- Best suited for classification tasks.
- The primary objective of the SVM algorithm is to identify the optimal hyperplane in an N-dimensional space that can effectively separate data points into different classes in the feature space.
- The algorithm ensures that the margin between the closest points of different classes, known as `support vectors` , is maximized.
- The dimension of the hyperplane depends on the number of features. For instance, if there are two input features, the hyperplane is simply a line, and if there are three input features, the hyperplane becomes a 2-D plane. As the number of features increases beyond three, the complexity of visualizing the hyperplane also increases.

Main Objective

- To find the optimal hyperplane that maximally separates data points from different classes in a high dimensional space.
- To create a decision boundary (hyperplane) that best differentiates between two categories of data, while maximizing the margin, which is the distance between the closest data points (called support vectors) of each class to the hyperplane.
- Maximize the margin between classes to improve generalization.
- Minimize classification error by ensuring correct placement of support vectors.
- Handle both linear and non-linear classification by using kernel functions to map data into higher-dimensional spaces when necessary.

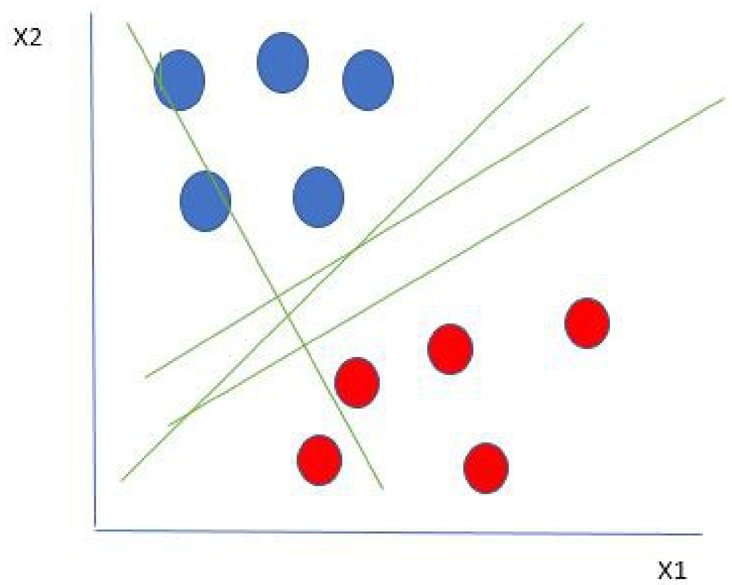
Terminologies

- `Hyperplane` : the decision boundary used to separate data points of different classes in a feature space. For linear classification, this is a linear equation represented as $wx + b = 0$.
- `Support Vectors` : the closest data points to the hyperplane. These points are critical in determining the hyperplane and the margin in Support Vector Machine (SVM).

- `Margin` : to the distance between the support vector and the hyperplane. The primary goal of the SVM algorithm is to maximize this margin, as a wider margin typically results in better classification performance.
- `Kernel` : a mathematical function used in SVM to map input data into a higher-dimensional feature space. This allows the SVM to find a hyperplane in cases where data points are not linearly separable in the original space. Common kernel functions include `linear` , `polynomial` , `radial basisfunction (RBF)` , and `sigmoid` .
- `Hard Margin` : the maximum-margin hyperplane that perfectly separates the data points of different classes without any misclassifications.
- `Soft Margin` : method introduces a slack variable for each data point to allow some misclassifications while balancing between maximizing the margin and minimizing violations when data contains outliers or is not perfectly separable.
- `C` : a regularization term that balances margin maximization and the penalty for misclassifications. A higher C value imposes a stricter penalty for margin violations, leading to a smaller margin but fewer misclassifications.
- `Hinge Loss` : a common loss function in SVMs. It penalizes misclassified points or margin violations and is often combined with a regularization term in the objective function.
- `Dual Problem` : involves solving for the Lagrange multipliers associated with the support vectors. This formulation allows for the use of the kernel trick and facilitates more efficient computation.

SVM Hyperplane Explanation

- Consider two independent variables, `x1` and `x2` , and one dependent variable represented as either a blue circle or a red circle.



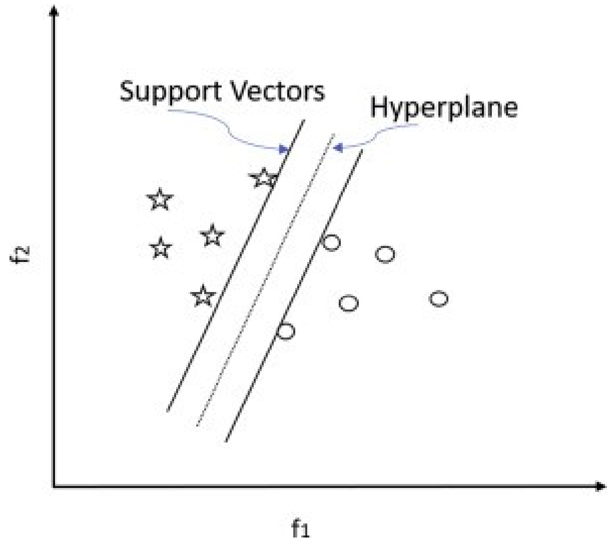
- In this scenario, the hyperplane is a line because we are working with two features (x1 and x2).
- There are multiple lines (or hyperplanes) that can separate the data points.
- The challenge is to determine the best hyperplane that maximizes the separation margin between the red and blue circles.

Common Applications of SVMs Include

- Image classification
- Text Classification & Natural Language Processing (NLP)
- Bioinformatics analysis
- Financial prediction
- Anomaly detection
- E-Commerce
- Robotics & Control Systems
- Remote Sensing
- Time-Series Forecasting
- Recommendation Systems
- Handwriting Recognition
- Speech Recognition
- Medical Diagnosis

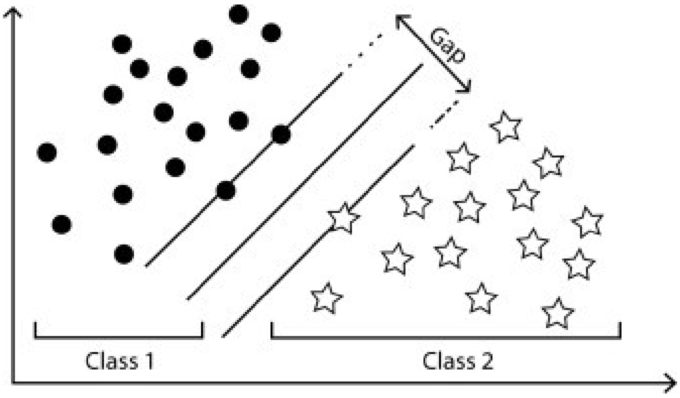
Types of SVM

- Based on the nature of the decision boundary, Support Vector Machines (SVM) can be divided into two main parts:
    - Linear SVM:
        - Use a linear decision boundary to separate the data points of different classes.
        - When the data can be precisely linearly separated, linear SVMs are very suitable.
        - This means that a single straight line (in 2D) or a hyperplane (in higher dimensions) can entirely divide the data points into their respective classes.
        - A hyperplane that maximizes the margin between the classes is the decision boundary.



    - Non-Linear SVM:
        - Used to classify data when it cannot be separated into two classes by a straight line (in the case of 2D).
        - By using kernel functions, nonlinear SVMs can handle nonlinearly separable data.
        - The original input data is transformed by these kernel functions into a higher-dimensional feature space, where the data points can be linearly separated.

- A linear SVM is used to locate a nonlinear decision boundary in this modified space.



Advantages of Support Vector Machine (SVM)

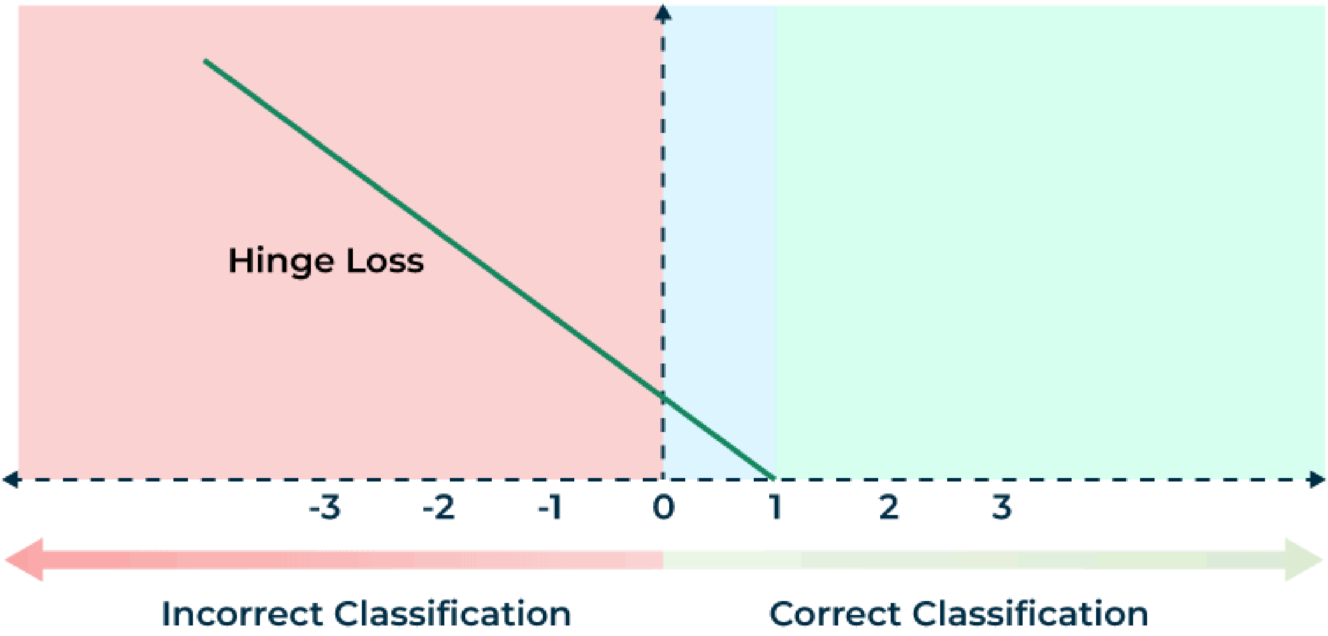
- `High-Dimensional Performance` : SVM excels in high-dimensional spaces, making it suitable for image classification and gene expression analysis.
- `Nonlinear Capability` : Utilizing kernel functions like RBF and polynomial, SVM effectively handles nonlinear relationships.
- `Outlier Resilience` : The soft margin feature allows SVM to ignore outliers, enhancing robustness in spam detection and anomaly detection.
- `Binary and Multiclass Support` : SVM is effective for both binary classification and multiclass classification, suitable for applications in text classification.
- `Memory Efficiency` : SVM focuses on support vectors, making it memory efficient compared to other algorithms.

Disadvantages of Support Vector Machine (SVM)

- `Slow Training` : SVM can be slow for large datasets, affecting performance in SVM in data mining tasks.
- `Parameter Tuning Difficulty` : Selecting the right kernel and adjusting parameters like C requires careful tuning, impacting SVM algorithms.
- `Noise Sensitivity` : SVM struggles with noisy datasets and overlapping classes, limiting effectiveness in real-world scenarios.
- `Limited Interpretability` : The complexity of the hyperplane in higher dimensions makes SVM less interpretable than other models.
- `Feature Scaling Sensitivity` : Proper feature scaling is essential; otherwise, SVM models may perform poorly.

Hinge loss in SVM

- A loss function used for training classifiers, most notably the SVM models.
- It is used for the de-facto maximum-margin classification for SVMs.
- Hinge loss is used in binary classification problems where the objective is to separate the data points in two classes typically labeled as +1 and -1.



- When a datapoint's distance from the boundary is greater than or equal to 1, the loss size is 0.
- If the distance from the boundary is less than 1, the we incur a loss. At 0 distance (the data point is on the boundary), then the loss size is 1.
- Correctly classified points will have a small loss size, while incorrectly classified instances will have a high loss size.
- High hinge loss indicates datapoints being on the wrong side of the boundary, and hence are misclassified; while a positive distance calls for low (or zero) hinge loss and correct classification:
  - $L(y, f(x)) = max(0, 1 – y * f(x))$ where:
    - y- the actual class (-1 or 1)
    - $f(x)$ – the output of the classifier for the datapoint
- A simple and efficient loss function to optimize.
- Robust to noise in the data.
- Encourages SVMs to find hyperplanes with a large margin.

Perceptron Learning

- `Perceptron learning` is a fundamental algorithm in machine learning that can be seen as a precursor to Support Vector Machines (SVMs).

Perceptron learning Vs. SVM

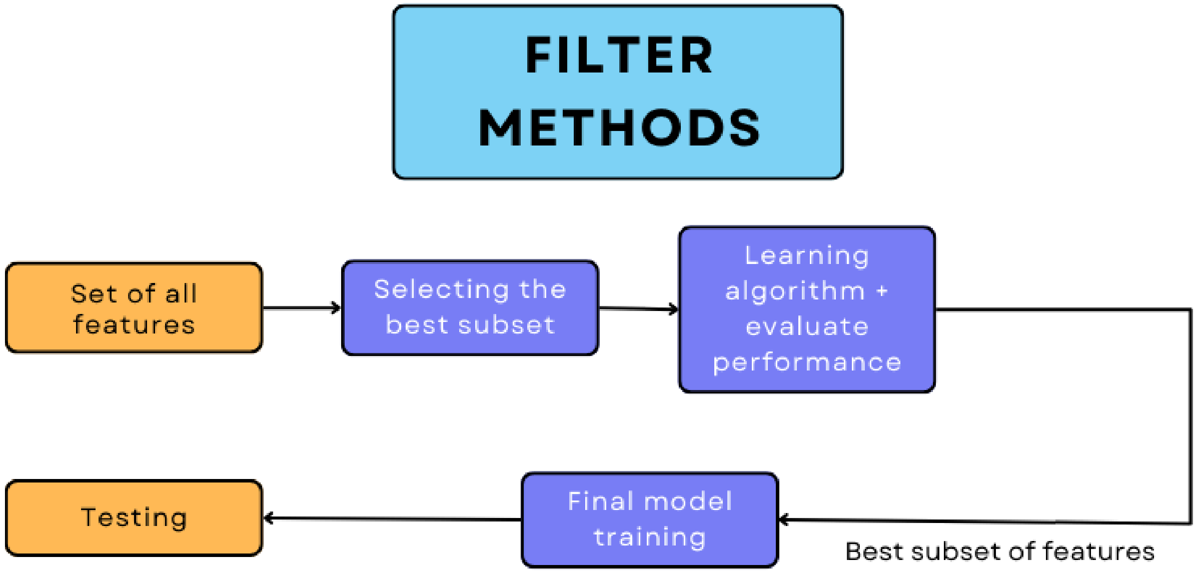
Aspect	Perceptron	SVM (Support Vector Machine)
Objective	Finds a hyperplane that separates the data, without considering margin.	Maximizes the margin (distance) between the hyperplane and the nearest data points, leading to better generalization.
Data Type	Works with linearly separable data.	Can handle non-linearly separable data using the kernel trick to map data into a higher-dimensional space.
Algorithm	Uses a simple iterative algorithm that updates weights based on misclassified points.	Formulates the problem as a quadratic programming optimization problem, providing a more globally optimal solution.

- Both algorithms are used for binary classification, aiming to find a hyperplane that separates data points into two classes.

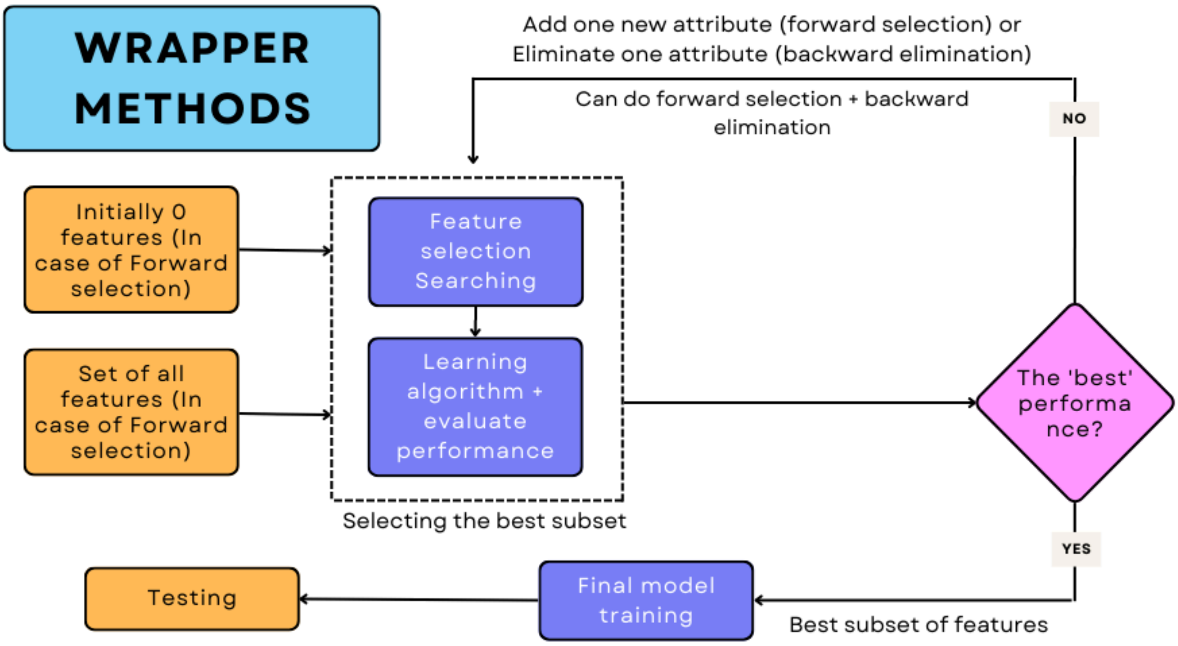
Feature Engineering and Selection Techniques in SVM

- Critical steps in optimizing the performance of SVM and other machine learning models.
- **Feature Engineering Techniques**:
    - Involves creating new features or modifying existing ones to improve the model's performance.
    - Types:
        - `Polynomial Features` : SVM with a linear kernel may not perform well if the data is not linearly separable. Polynomial features transform the original features into higher dimensions, allowing SVM to model more complex relationships.
        - `Kernel Trick` : The kernel trick is a powerful technique for implicit feature engineering in SVMs. Instead of manually transforming data into higher dimensions, a kernel function computes the inner product in a higher-dimensional space.
    - Importance:
        - `Enhanced Interpretability` : By choosing the most relevant features, you gain a clearer understanding of which factors significantly affect the model's predictions.
        - `Improved Efficiency` : Reducing the feature set lowers training and prediction times, making the model more computationally efficient.
        - `Reduced Overfitting` : Feature selection helps prevent the model from memorizing irrelevant details in the training data, leading to better generalization on unseen data.
        - `Better Generalization Performance` : A model trained on a well-chosen feature subset is likely to perform better on new data compared to a model using all features.
        - `Addressing Curse of Dimensionality` : In high-dimensional settings, SVMs can suffer from the curse of dimensionality, where performance degrades. Feature selection mitigates this effect.
    - Key considerations:
        - `Data Dimensionality` : SVM performs well in high dimensional spaces, but only when the features are informative. Irrelevant or redundant features can hurt performance.
        - `Non-linearity` : If data is not linearly separable, feature engineering techniques like polynomial features or kernel methods can help SVM create more complex decision boundaries.
        - `Curse of Dimensionality` : While SVM handles high dimensional spaces, an excessive number of irrelevant features can introduce noise and reduce the generalization ability of the model. Hence, feature selection is crucial.
        - `Computational Efficiency` : Feature selection can significantly reduce computational costs, especially with large datasets. Techniques like PCA, RFE, and filter methods are widely used to improve efficiency.
    - Methods;
        - `Filtering Method` :
            - Using some information, distance, or correlation measures.

- Here, the features' sub-setting is generally done using one of the statistical measures like the `Chi-square test`, `ANOVA test`, or `correlation coefficient`.
- These help in selecting the attributes that are highly correlated with the target variable.

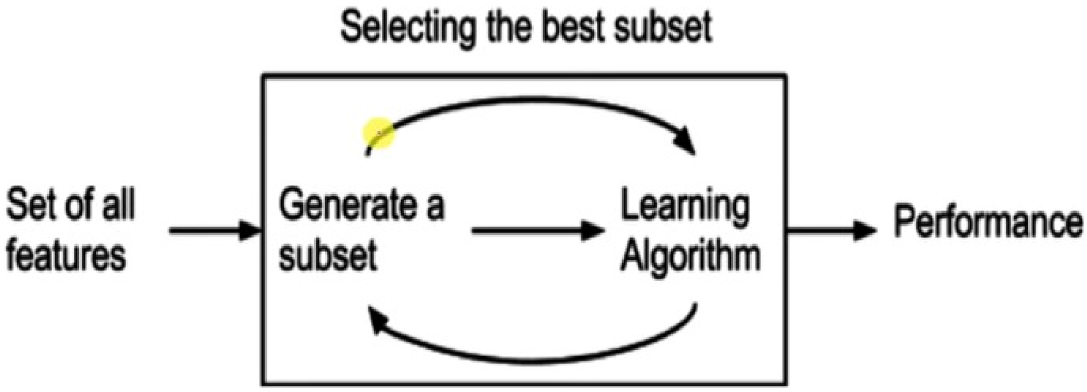


- `Wrapper Methods`:
  - Generate a new model for each feature subset that is generated.
  - The performance of each of these is recorded and the features which produce the best performance model are used for training and testing the final algorithm.
  - Unlike filter methods that use distance or information-based measures for feature selection, wrapper methods use many simple techniques for choosing the most significant attributes.

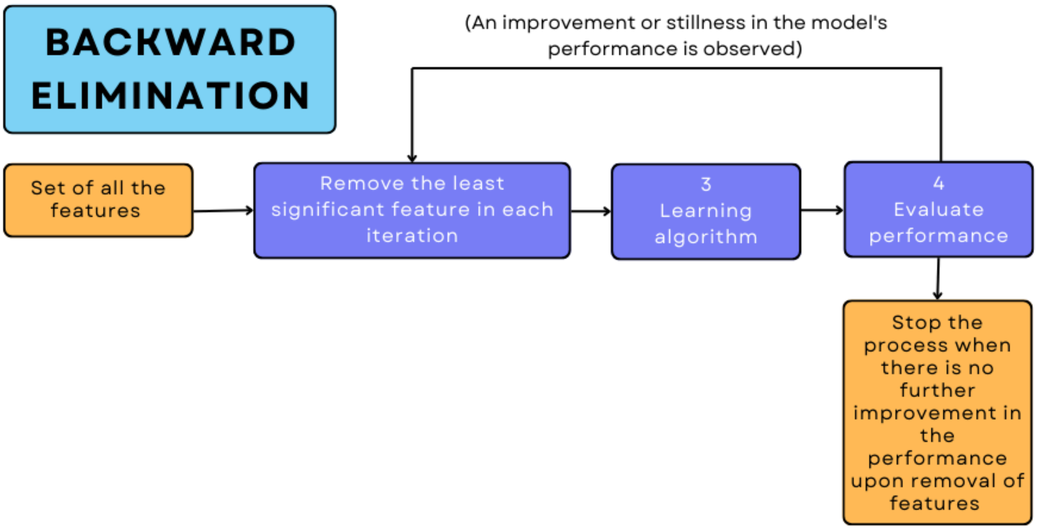


  - **Methods**:
    - `Forward feature selection`:

- This method adds features to the feature set iteratively, one at a time, starting with a blank set of features.
- The feature that enhances the model's performance the most at each step is chosen.
- The procedure is carried out repeatedly until the model's performance shows no further improvement.
- When there is a large number of features in the dataset and limited computational resources, forward feature selection is advantageous.



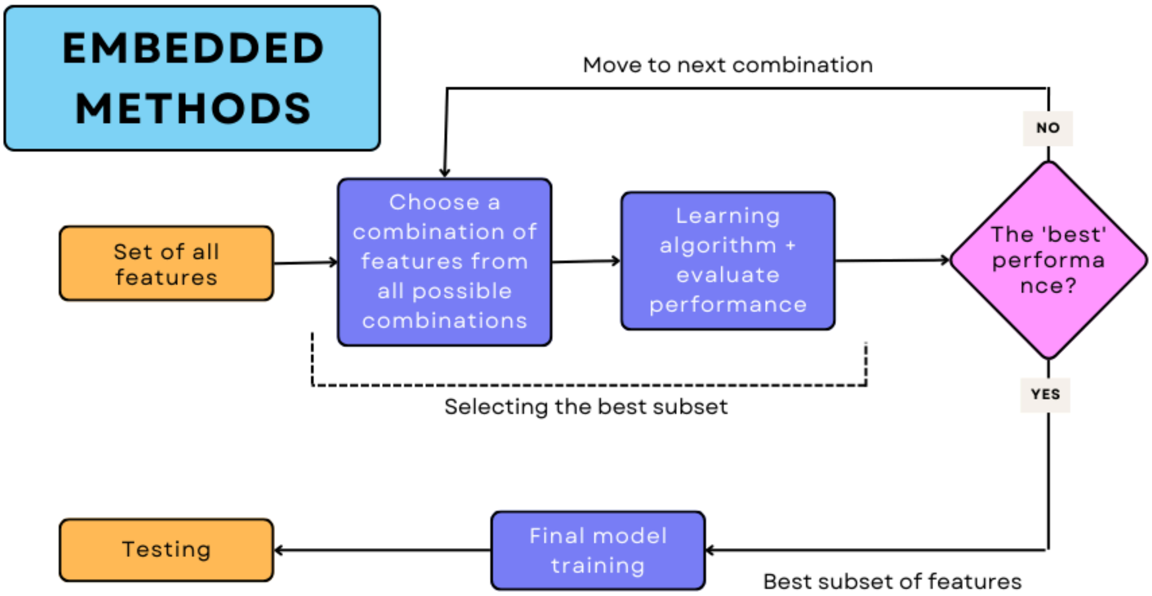
- `Backward feature selection` :
  - Backward feature selection, in contrast to forward feature selection, begins with every feature included and eliminates each feature one at a time.
  - The least amount of model performance loss determines which feature should be eliminated.
  - This method keeps going until the model's performance does not improve with additional feature removal.
  - With limited computational resources, backward feature selection might be helpful because it starts with the entire feature set and gradually eliminates less significant characteristics.



- `Recursive feature elimination` :
  - Recursively eliminating features from the entire feature set is the process of recursive feature selection.

- Either the coefficient magnitude or feature relevance might serve as the selection criterion.
- The model is retrained using the remaining features after the least significant feature is eliminated at each phase.
- This method keeps going until the target feature count is reached or until removing more features noticeably impairs the performance of the model.
- Recursive feature selection helps you find the most important characteristics while making the model's computations less difficult.

- `Embedded methods` :
  - All the combinations of the features are generated.
  - Then each of these combinations of attributes is used to train the model, and as usual, its performance is observed.
  - The combination which gives the best performance is chosen for the final training.
  - Optimal feature selection for Support Vector Machines (SVMs) involves finding a subset of features that maximizes classification performance.



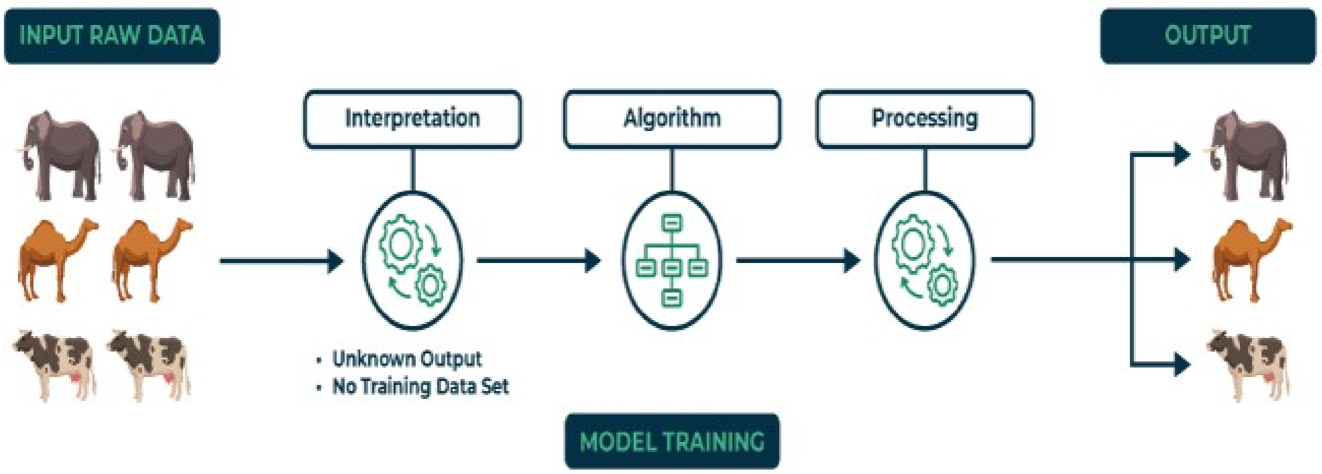
Popular kernels

- **Polynomial Kernel**:
  - Maps features into a higher polynomial dimension.
  - The polynomial kernel is a kernel function commonly used in Support Vector Machines (SVMs) and other kernelized models.
  - It maps the original data into a higher-dimensional feature space, allowing for the learning of non-linear relationships.
- **Radial Basis Function (RBF) Kernel**: Captures complex, non-linear relationships by using a distance-based similarity measure.
- **Sigmoid Kernel**: Similar to neural network activation functions and works well in certain non-linear problems.

Chapter 4: Unsupervised Learning

Objective of Unsupervised Learning

- To analyze and uncover hidden `patterns`, `structures`, or `relationships` within data that is unlabeled.



How does Unsupervised Learning Work?

- By analyzing unlabeled data to identify patterns and relationships.
- It's very rewarding, as it can reveal insights into the data that would not be apparent from a labeled dataset.

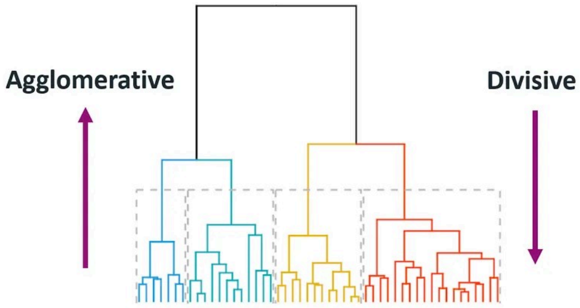
Inputs of Unsupervised Learning

- `Unstructured data` : May contain noisy(meaningless) data, missing values, or unknown data.
- `Unlabeled data` : Data only contains a value for input parameters, there is no targeted value(output). It is easy to collect as compared to the labeled one in the Supervised approach.

Unsupervised Learning Algorithms

- Clustering:
    - The process of grouping unlabeled data into clusters based on their similarities.
    - The goal is to identify patterns and relationships in the data without any prior knowledge of the data's meaning.
    - Uses patterns such as `similarities` or `differences`.
    - `Graphing`, `the shortest distance`, and `the density of the data points` are a few of the elements that influence cluster formation.
    - Types:
        - `Hard Clustering` : Type of clustering where each data point belongs to a cluster completely or not.
        - `Soft clustering` : Type of clustering where instead of assigning each data point into a separate cluster, a probability or likelihood of that point being that cluster is evaluated.
    - Clustering can be used for `Market Segmentation`, `Market Basket Analysis`, `Social Network Analysis`, `Medical Imaging`, and `Anomaly Detection`.

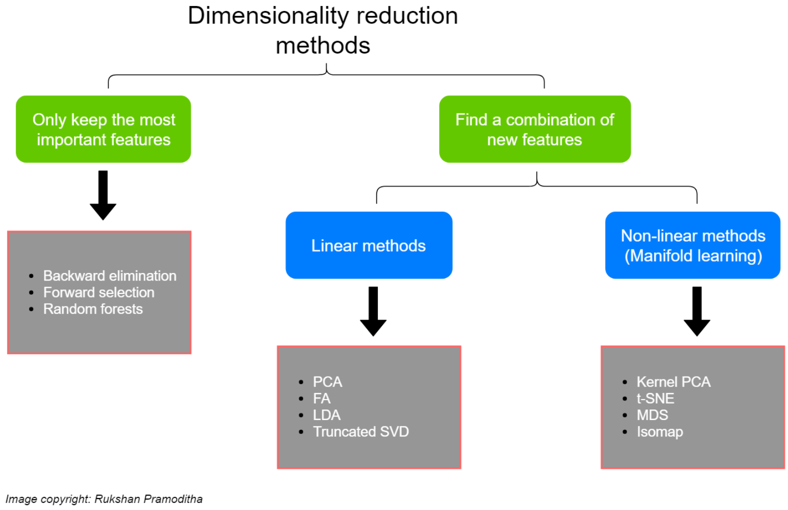
- Types of Clustering Algorithms:
  - `Centroid based clustering (Partitioning methods)` :
    - The most easiest clustering algorithm.
    - Group data points based on the basis of their closeness.
    - `Euclidian distance` , `Manhattan Distance` or `Minkowski Distance` are chosen as similarity measure.
    - The primary drawback for these algorithms is the requirement that we establish the number of clusters, “k,” either intuitively or scientifically (using the `Elbow Method`) before any clustering machine learning system starts allocating the data points.
    - `Examples` : `K-means` and `K-medoids` clustering.
  - `Density based clustering (Model based methods)` :
    - Finds groups based on the density of data points.
    - Great at handling clusters of different sizes and forms, making them ideally suited for datasets with irregularly shaped or overlapping clusters.
    - Overcomes the drawbacks of centroid-based techniques by autonomously choosing cluster sizes, being resilient to initialization, and successfully capturing clusters of various sizes and forms.
  - `Connectivity based clustering (Hierarchical clustering)` :
    - Technique of grouping data based on the their similarity or distance.
    - It builds a hierarchy of clusters, where clusters are formed in a step-by-step manner, starting with individual data points as single clusters and progressively merging them into larger clusters based on their connections.
    - `Process` :
      - Starts with each data point in its own cluster.
      - Progressively merges the most similar clusters based on a predefined distance or similarity measure.
      - Continues merging until all data points belong to a single cluster.
    - `Output` :
      - A tree-like structure called a dendrogram that visually depicts the merging process and the hierarchical relationships between clusters.
      - You can decide on the desired number of clusters by cutting the dendrogram at a specific level.
    - `Approaches` :
      - `Divisive Clustering` : It follows a top-down approach, here we consider all data points to be part one big cluster and then this cluster is divide into smaller groups.
      - `Agglomerative Clustering` : It follows a bottom-up approach, here we consider all data points to be part of individual clusters and then these clusters are clubbed together to make one big cluster with all data points.



- `Distribution based clustering` :
  - Groups data points based on their statistical distributions.
  - Assumes that the data points are generated from underlying probability distributions.
  - It aims to find clusters by modeling these distributions. (e.g., `Gaussian distribution` , `binomial distribution`).
  - `Gaussian Mixture Models (GMM)` : It represents the data as a mixture of several Gaussian distributions, each representing a cluster.
  - `Expectation-Maximization (EM) Algorithm` : GMM clustering typically employs the EM algorithm to iteratively estimate the parameters of the Gaussian distributions and assign data points to clusters based on these estimates.
  - Used in `image segmentation` , `anomaly detection` , and `customer segmentation` , where understanding underlying distributions of data is crucial for analysis.

Dimensionality Reduction Techniques

- `Dimension reduction` is a process in data analysis and machine learning that reduces the number of variables (features) in a dataset while retaining as much meaningful information as possible.



- Techniques:
  - PCA (Principal Component Analysis):
    - Widely used dimensional reduction technique.
    - Simplifies complex datasets by transforming them into a lower-dimensional space while capturing the significant information.
    - Key Ideas:

        - `Dimensionality Reduction` : PCA aims to find a new set of variables, called principal components (PCs), that explain most of the data's variance in a lower-dimensional space (fewer variables).
        - `Focus on Variance` : represent the directions of greatest variation in the data. The first PC captures the most variance, the second captures the remaining most significant variance, and so on.
        - `Data Compression` : By focusing on the most informative directions (PCs), PCA reduces data complexity while retaining essential patterns.
    - Benefits:
        - Improved Visualization
        - Noise Reduction
        - Computational Efficienc
    - While convenient, PCA does discard some information during dimensionality reduction. The choice of how many PCs to retain depends on the acceptable information loss.
    - Depending on the data, PCs may not directly correspond to the original variables, making their interpretation less intuitive.
- t-distributed Stochastic Neighbor Embedding (t-SNE):
    - A visualization technique used for exploring high-dimensional data by embedding it into a lower-dimensional space, typically 2 or 3 dimensions, suitable for visualization.
    - Unlike PCA it excels at preserving the `local similarities` between data points in the high-dimensional space.
    - Key Points:
        - `Non-linear Dimensionality Reduction` : t-SNE can handle non-linear relationships within the data.
        - `Preserving Local Structure` : t-SNE aims to maintain the similarities between nearby points in the high-dimensional space even after embedding them in a lower-dimensional space allowing for visualization of complex, potentially curved, structures in the data.
        - `Stochastic Process` : The algorithm employs a probabilistic approach to optimize the embedding, making it more robust to noise compared to deterministic methods.
    - Advantages of t-SNE:
        - `Visualizing Complex Structures` : Effective for revealing non-linear relationships and clusters in high-dimensional data.
        - `Intuitive Interpretation` : Easier to interpret the relative positions of data points in the lower-dimensional space compared to PCA.
    - Considerations:
        - `Computational Cost` : t-SNE can be computationally expensive compared to PCA.
        - `Parameter Tuning` : The algorithm's performance can be sensitive to certain parameters that require some experimentation for optimal results.
        - `Visualization, Not Perfect Clustering` : While it can reveal clusters, t-SNE is primarily a visualization tool, and the resulting clusters may not be perfect for all

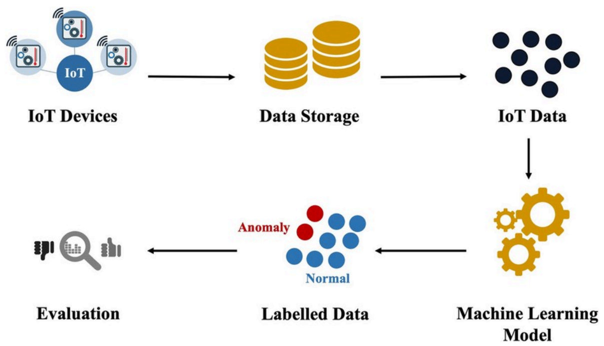
applications.

PCA vs t-SNE

	PCA	t-SNE
Purpose	Dimensionality reduction	Data visualisation in low dimensions (typically 2D)
Type of algorithm	Deterministic	Stochastic
Unique solution?	Yes	No
Projection type	Linear	Non-linear
Type of approach	Global	Local/Global*
Interpretation	PCA is just a rotation of axes	Subjective interpretation

Anomaly Detection

- Focuses on finding events or data points that deviate significantly from the majority of data, often indicating rare or unusual events.
- These anomalies can be caused by various factors, such as errors, faults, fraud, or rare events.
- The goal is to distinguish these anomalies from “normal” data points to take appropriate actions or gain valuable insights.



- Examples:
    - `Finance` : Credit card fraud detection, and Stock market analysis.
    - `Cybersecurity` : Intrusion detection systems (IDS), Server anomaly detection, Bot Detection and Malware Detection.
    - `Healthcare` : Disease outbreak detection, Patient health monitoring, and Medical Image Analysis.
    - `Manufacturing` : Quality control, Predictive maintenance, Equipment Failure Prediction, and Supply Chain Anomaly Detection.
    - `E-commerce` : Clickstream analysis, and Fraudulent review detection.
    - `Internet of Things (IoT)` : Sensor Data Anomaly Detection, Network Traffic Anomaly Detection, and Device Behavior Anomaly Detection.

Outliner Analysis

- Deals with identifying data points that fall far outside the majority of the data.
- These outliers can be legitimate or illegitimate.
- Helps us understand the distribution of the data and identify potential issues with data collection or measurement.

Anomaly Detection vs Outliner Analysis

- `Anomaly detection` focuses on identifying unusual events or patterns, while `outlier analysis` focuses on identifying unusual data points.
- Example: Imagine a class taking a math exam:
  - `Anomaly detection` would identify a sudden spike in very high scores, which could indicate cheating.
  - `Outlier analysis` would identify the individual student(s) with the very high scores to investigate further (legitimate reason or not).

Chapter 5: Model Optimization and Evaluation

Gradient Descent

- `Gradient descent` is a fundamental optimization algorithm used to train machine learning models that iteratively adjusts the model's parameters in the direction of the negative gradient of the objective function, allowing the model to converge to a (local) minimum.
- It is an optimization algorithm, which is an algorithm that can be used for optimizing our cost function, in order to make our data model more accurate and less error prone.
- Works very well for more supervised learning like Linear regression, Logistic regression ,etc.
- In `linear regression`, gradient descent is used to find the best-fitting line by `minimizing the Mean Squared Error (MSE) loss function`. The model parameters, slope (m) and intercept (b), are updated iteratively. At each step, the gradients (partial derivatives of the loss function with respect to m and b) are calculated, and the parameters are adjusted in the direction that reduces the loss, ensuring the line fits the data better over time.
- In `neural network training`, gradient descent updates the network's weights to minimize a loss function (e.g., categorical cross-entropy for classification). `Forward propagation` computes the `network's output`, and `backpropagation calculates` the `gradients of the loss with respect to the weights`. The weights are then adjusted using gradient descent, iteratively improving the network's performance.

Optimization Variants

- Advanced techniques designed to improve the performance of the basic gradient descent algorithm by addressing its limitations, such as `slow convergence`, `difficulty handling noisy gradients`, and `suboptimal exploration of the parameter space`.
- **Stochastic Gradient Descent (SGD)**:
  - Commonly used in training large-scale machine learning models, particularly in `deep learning`.
  - SGD computes the gradient using only a subset of the data at each iteration (mini-batch) instead of computing the gradient of the loss function using the entire dataset like Gradient Descent.
  - Significantly reduces the `computational cost` and `memory requirements` of training the model.
  - Example: Training a Convolutional Neural Network (CNN) for image classification.

- **Momentum Optimization**:
    - A variant of gradient descent that helps `accelerate convergence` and `improve the stability of the optimization process`.
    - In addition to considering the current gradient, momentum optimization also takes into account the accumulated gradient from previous iterations.
    - Example: Training a Recurrent Neural Network (RNN).
- **Adaptive Learning Rate Methods**:
    - Adjust the learning rate during training based on the history of gradients observed in previous iterations.
    - These methods aim to improve the `convergence speed` and `stability of gradient optimization` algorithms by adapting the learning rate for each parameter individually.
    - Example: Training a deep neural network (DNN) for image recognition.

Adaptive Learning Rate Methods

- **AdaGrad (Adaptive Gradient Algorithm)**:
    - Adapts the learning rate to the parameters, performing `smaller updates` for `parameters associated with frequently occurring features`, and `larger updates` for `parameters associated with infrequent features`.
    - It achieves this by accumulating the `square of the gradients for each parameter in a vector`, which it then uses to adjust the learning rates.
    - Example: Training model on text data where words occur with different frequencies.
- **RMSprop (Root Mean Square Propagation)**:
    - Modifies AdaGrad to perform well in `non-convex optimization problems`, like `training neural networks`.
    - It does so by using a moving average of squared gradients to scale the learning rate, addressing AdaGrad's aggressive, monotonically decreasing learning rate.
    - Example: In deep learning, especially in tasks like training Recurrent Neural Networks (RNNs) for natural language processing(NLP)
- **Adam (Adaptive Moment Estimation)**:
    - Combines ideas from `RMSprop` and `momentum`.
    - Besides keeping an exponentially decaying average of past squared gradients (like RMSprop), Adam also keeps an exponentially decaying average of past gradients, similar to momentum.
    - It calculates an adaptive learning rate for each parameter.
    - Suited for large datasets and complex neural network architectures.
    - Example: In a Computer Vision task, such as image classification.
- **Hyperparameter Tunning**:
    - `Learning rate` is a hyperparameter that controls the step size at which a model's parameters are updated during training. It is a crucial parameter in optimization algorithms, such as gradient descent, that are used to minimize the loss function and improve the model's performance.
    - `Hyperparameter tunning` refers to the process of adjusting the settings (hyperparameters) of a machine learning model to improve its performance.

- By tunning the `learning rate` and other hyperparameters we can find the settings that allow the model to learn effectively from the training data and generalize well to unseen data.
- Example: Training machine learning model to classify images of cats and dogs.
- Models:
  - Grid Search:
    - A brute-force method that systematically iterates through a specified subset of hyperparameters, training a model with each combination, and recording the performance of each.
    - The combination that yields the best performance is then selected.
    - Preferable when you have relatively `few hyperparameters` and `computational resources` are not a limiting factor.
  - Random Search:
    - Selects random combinations of hyperparameters to train the model.
    - The process is repeated for a specified number of iterations and the combination that yields the best performance during validation is then selected.
    - Efficient in scenarios with `high-dimensional hyperparameter spaces` or when certain hyperparameters are believed to be important than others.
  - Bayesian optimization:
    - An advanced strategy for hyperparameter optimization in machine learning that seeks to minimize the number of evaluations needed to find the best hyperparameters.
    - Builds a probabilistic model to predict the performance of various hyperparameters and intelligently choose the next set of hyperparameters to evaluate based on this model.
    - Useful when evaluations of the objective function (`model training and validation`) are `expensive` , `time-consuming` or `resource intensive` .
    - `Benefits` :
      - `Efficient Hyperparameter Tuning` : It focuses evaluations on promising areas, reducing the number needed compared to grid search.
      - `Data-Driven Exploration` : It balances exploration and exploitation, leading to potentially better solutions.
      - `Uncertainty Management` : It considers the uncertainties in performance estimates, making the search more robust.

Overfitting

- Occurs when a model performs exceptionally well on the training data but fails to generalize to new, unseen data.
- This can happen when a model is `too complex` and `has memorized the training data` , rather than `learning the underlying patterns` .
- To build a machine learning model that performs well on unseen data we need to implement:
  - **Optimization**: Finding the best possible configuration for your model to achieve accurate predictions.

    - **Evaluation**: Assessing how well your optimized model performs on unseen data to ensure it generalizes well.

Model Evaluation and Validation

- Refers to the process of assessing the performance and generalization ability of a machine learning model on unseen data.

- Involves splitting the dataset into training and testing subsets.

- **Cross Validation Techniques**:

    - A resampling technique used to assess the performance of a machine learning model and to mitigate issues like overfitting.
    - It involves dividing the dataset into multiple subsets, called `folds`, and training the model multiple times, each time using a different fold as the testing set while the rest are used for training.
- **Performance Metrics**:

    - Measures used to evaluate the effectiveness and accuracy of machine learning model.
    - Different metrics are used depending on the type of problem and the specific requirements of the task.
    - Basic Terms:
        - True Positive (TP):
            - Cases where the model correctly predicts the positive class.
            - Example: If the person has the disease (actual positive) and the test correctly detects it (predicted positive), it's a true positive.
        - True Negative (TN):
            - Cases where the model correctly predicts the negative class.
            - Example: If the person does not have the disease (actual negative) and the test correctly predicts no disease (predicted negative), it's a true negative.
        - False Positive (FP):
            - Cases where the model incorrectly predicts the positive class for a negative instance.
            - Example: If the person does not have the disease (actual negative) but the test predicts they do (predicted positive), it's a false positive.
        - False Negative (FN):
            - Cases where the model incorrectly predicts the negative class for a positive instance.
            - Example: If the person has the disease (actual positive) but the test fails to detect it (predicted negative), it's a false negative.

1. **Accuracy:** The proportion of correctly classified instances out of all instances.
   $$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$
2. **Precision:** The proportion of true positive predictions out of all positive predictions.
   $$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives+False Positives}}$$
3. **Recall (Sensitivity):** The proportion of true positive predictions out of all actual positive instances.
   $$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives+False Negatives}}$$
4. **F1-score:** The harmonic mean of precision and recall, providing a balance between the two metrics.
   $$F1 = 2 \times \frac{\text{Precision}\times\text{Recall}}{\text{Precision+Recall}}$$

- ROC Curve (Receiver Operating Characteristic Curve):
  - A graphical plot that illustrates the diagnostic ability of a binary classifier across different threshold settings.
  - It plots the `True Positive Rate (TPR)` against the `False Positive Rate (FPR)` at various threshold settings.
- AUC-ROC (Area Under the ROC Curve):
  - A single scalar value representing the area under the `ROC curve`.
  - It provides an aggregate measure of the classifier's performance across all possible threshold settings.
  - AUC-ROC = 1 indicates a `perfect classifier`.
  - AUC-ROC = 0.5 indicates a `random classifier`.

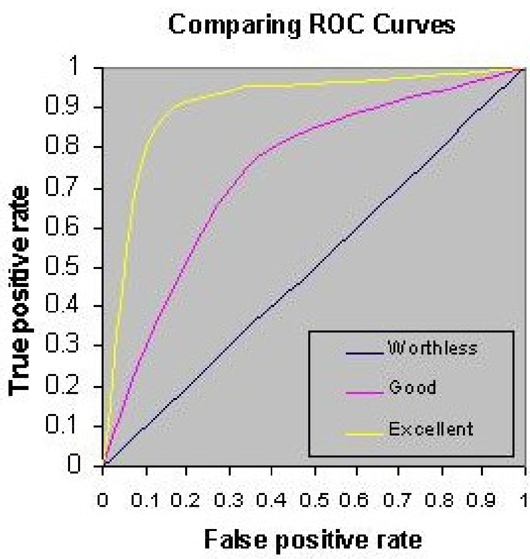
Performance Metrics for Classification and Regression

- For classification:
  - Accuracy:
    - It's a good initial indicator of `performance` but can be misleading in `imbalanced datasets`.
    $$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$
  - Precision (Positive Predictive Value):
    - It is crucial when the `cost of false positives is high`.
    $$Precision = \frac{TP}{TP+FP}$$
  - Recall (Sensitivity or True Positive Rate):
    - It is crucial when the `cost of false negatives is high`.
    $$Recall = \frac{TP}{TP+FN}$$
  - F1 Score:
    - It is useful when you need a balance between `precision` and `recall` and there's an uneven class distribution.
    $$F_1 = 2 * \frac{precision*recall}{precision+recall}$$

- ROC-AUC (Receiver Operating Characteristic - Area Under Curve):
  - The AUC represents the likelihood of the model distinguishing between positive and negative classes.
  - It is plotted between FPR (X-axis) and TPR (Y-axis).

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

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



- Confusion Matrix:
  - A table showing the actual vs. predicted classifications.
  - It gives a detailed breakdown of correct and incorrect classifications for each class.

	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

- **Specificity**:
  - The ratio of True negatives and True negatives + False positives.
  - We want the value of specificity to be high. Its value lies between [0,1].

$$specificity = \frac{TN}{TN+FP}$$

- **Log Loss**:
  - Measures the performance of a classification model where the prediction is a probability value between 0 and 1.
  - Increases as the predicted probability diverge from the actual label.
  - Used metric for Kaggle competitions.

$$\text{log-loss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log p_i + (1-y_i)\log(1-p_i).$$

- **For regression**:
    - Mean Absolute Error (MAE):
        - A measure of the average size of the mistakes in a collection of predictions, without taking their direction into account.
        - It is measured as the average absolute difference between the predicted values and the actual values and is used to assess the effectiveness of a regression model.

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|$$

where:

n is the number of observations in the dataset.

y_i is the true value.

ŷ_i is the predicted value.

Example Say, y_i = [5,10,15,20] and $\hat{y}_i$ = [4.8,10.6,14.3,20.1]

Thus, MAE = 1/4 * (|5-4.8|+|10-10.6|+|15-14.3|+|20-20.1|) = 0.4

- Mean Squared Error (MSE):
    - The average squared difference between the value observed in a statistical study and the values predicted from a model.
    - Squaring these differences eliminates this situation.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

Here, the error term is squared and thus more sensitive to outliers as compared to Mean Absolute Error (MAE).

Thus, MSE = 1/4 * (|5-4.8|^2+|10-10.6|^2+|15-14.3|^2+|20-20.1|^2) = 0.225

- Root Mean Squared Error (RMSE):
    - The average difference between a statistical model's predicted values and the actual values.
    - Mathematically, it is the standard deviation of the `residuals`.
    - Residuals represent the distance between the regression line and the data points.
    - In other words, it tells you how concentrated the data is around the line of best fit
    - RMSE quantifies how dispersed these residuals are, revealing how tightly the observed data clusters around the predicted values.

$$RSME = \sqrt{\frac{\Sigma(y_i - \hat{y}_i)^2}{N - P}}$$

Where:

- **y_i is the actual value for the i^{th} observation.**
- **$\hat{y}_i$ is the predicted value for the i^{th} observation.**
- **N is the number of observations.**
- **P is the number of parameter estimates, including the constant.**

- R-squared (Coefficient of Determination):
    - Measures how well a statistical model predicts an outcome.
    - It explains the degree to which the input variables explain the variation of the output / predicted variable.
    - The higher the R squared, the more variation is explained by the input variables and better is the model.

  - Although, there exists a limitation in this metric, which is solved by the Adjusted R-squared.
  - The outcome is represented by the model's dependent variable.
  - The lowest possible value of R^2 is 0 and the highest possible value is 1.
  - Put simply, the better a model is at making predictions, the closer its R^2 will be to 1.
  - The limitation of R-squared is that it will either stay the same or increases with the addition of more variables, even if they do not have any relationship with the output variables.

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

- Adjusted R-squared:
  - It penalizes you for adding the variables which do not improve your existing model.
  - Hence, if you are building Linear regression on multiple variables, it is always suggested that you use Adjusted R-squared to judge the goodness of the model.
  - If there exists only one input variable, R-square and Adjusted R squared are same.

$$Adjusted\ R^2 = 1 - \frac{(1 - R^2)(N-1)}{N-p-1}$$

Feature Engineering

- Involves creating new features or modifying existing ones to improve the model's performance.
- Involves transforming raw data into features that a machine learning algorithm can understand and use for predictions.
- Aims to improve the quality and relevance of features, leading to better model performance and deeper understanding of the data.

Feature Engineering Common Techniques

- **Handling missing values**: Techniques like imputation fill in missing data points.
- **Categorical encoding**: Converts categorical variables into numerical values for algorithms to process.
- **Feature scaling**: Standardizes features to a common range, improving model convergence.
- **Feature creation**: Derives new features from existing ones to capture hidden patterns.
- **Dimensionality reduction**: Techniques like PCA reduce the number of features while preserving important information, useful for datasets with many features.
- **Variable transformations**: Applying transformations like log or square root can improve linearity.

Feature Selection

- Focuses on choosing a subset of the most relevant features from the available ones.
- Reduces model `complexity`, `training time`, and the `risk of overfitting`.

- **Common techniques**:
  - Filter methods: Use statistical tests (like information gain or chi-square) to score features based on their correlation with the target variable.
  - Wrapper methods: Evaluate feature subsets based on their performance in a machine learning model.
  - Regularization techniques: L1 regularization (LASSO) inherently performs feature selection by assigning weights to features, shrinking unimportant ones to zero.

Chapter 6: Probabilistic Graphical Models

What are PGMs?

- Graphical representations of probability distributions that encode dependencies and independencies between random variables.

Why are they useful?

- Capture complex relationships efficiently.
- Facilitate reasoning and inference about these relationships.
- Provide a visual and intuitive framework for understanding uncertainty.

Core concepts

- **Graph**:
  - `Vertices` represent random variables.
  - `Edges` represent dependencies between variables:
    - `Directed edges` (arrows) imply a causal relationship (`Bayesian Networks`).
    - `Undirected edges` imply a mutual influence (`Markov Networks`).
  - `Model` :
    - Mathematical description associated with the graph.
    - It specifies the joint probability distribution over all variables:
      - `Conditional Probability Tables (CPTs)` for `discrete` variables.
      - `Probability Density Functions (PDFs)` for `continuous` variables.
  - `Data` : Observations of the random variables used to learn or fit the model.

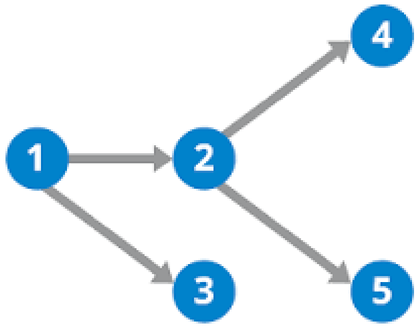
Key Questions in PGMs

- **Representation**: *How to effectively encode the relationships between variables in the graph structure?*
  - Domain knowledge plays a crucial role in defining these relationships.
- **Inference**: *How to answer queries about the probability of certain events given the model and observed data?*
  - This can involve computing marginal probabilities or finding the most probable explanation for a set of observations.
  - Techniques like `message passing algorithms` are used for inference.
- **Learning**: *How to learn the parameters of the model from data?*

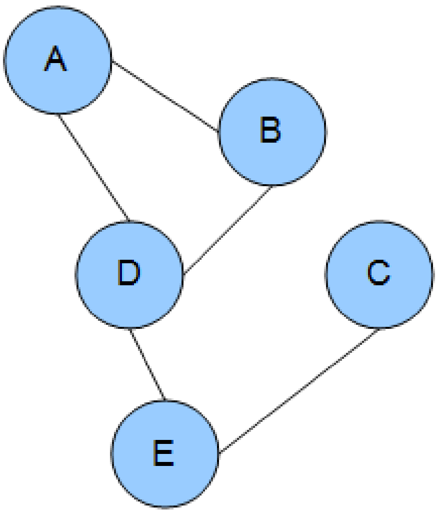
- This involves algorithms like `parameter estimation` and `structure learning`.

Types of PGMs

- **Directed Acyclic Graphs (DAGs) or Bayesian Networks**:
  - Represent causal relationships between variables.
  - Efficient inference for specific queries.



- **Markov Networks**:
  - Capture undirected dependencies between variables.
  - Can model complex interdependence but inference can be intractable for large models.



PGMs Application

- Machine vision
- NLP
- Recommendation systems
- Bioinformatics
- Healthcare Application Genetic Analysis
- Robotics
- Finance
- Social Network Analysis
- Time series analysis
- Cybersecurity

Bayesian Networks

- Also known as `belief networks`, or `probabilistic networks`.
- A specific type of `Probabilistic Graphical Model (PGM)` that utilizes `Directed Acyclic Graphs (DAGs)` to represent relationships between variables.
- **Key Components**:
  - Nodes: Represent random variables relevant to the engineering problem which can be discrete or continuous.
  - Directed Edges(Arc):
    - Illustrate causal dependencies between variables.
    - An arrow from A to B signifies that A directly influences the probability of B.
  - Conditional Probability Tables (CPTs):
    - Quantify these dependencies. Each node has a
    - CPT that specifies the probability of its state given the states of its parent nodes (nodes with arrows pointing towards it).
  - Joint Probability Distribution: A BN represents the joint probability distribution of a set of variables.

Construction of Bayesian Networks

- **Determining Structure**: The structure of a BN can be designed based on `domain expertise`, or learned from data using various algorithms such as `constraint-based` algorithms (e.g., PC algorithm) and `score-based` algorithms (e.g., Bayesian Information Criterion).
- **Parameter Estimation**: Once the structure is determined, the next step is to estimate the conditional probability tables (CPTs) for each node. Parameters can be learned from data using maximum likelihood estimation or Bayesian estimation.

Inference in Bayesian Networks

- **Types**:
  - Predictive Inference: Given observations for some variables, calculate the posterior probabilities for others.
  - Diagnostic Inference: Given an effect, compute the likelihood of different causes.
  - Inter-causal Inference: Understanding how evidence about one cause affects the belief in other causes.

Advantages of Bayesian Networks

- Engineering Design: `Risk Analysis`, and `Descision Support System`.
- Systems and Control Engineering: `Fault Detection`, and `Control Systems`.
- Software Engineering: `Bug Prediction`, and `Effort Estimation`.
- Intuitive Representation
- Efficient Inference
- Incorporation of Prior Knowledge
- Handling Uncertainty

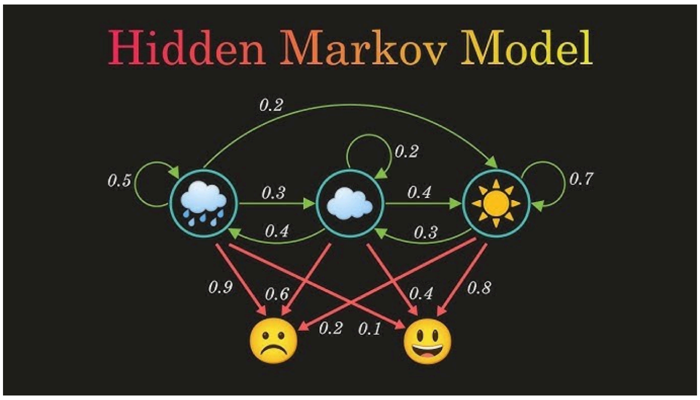
Hidden Markov Models

- A way to predict hidden states of a system based on observable outcomes.

- **Components**:
    - Hidden States: These are the actual conditions (unobservable states) we can't see directly that influence the observable outputs.
    - Observations: The data we observe, influenced by the hidden states.
    - Transitions: The chances of moving from one hidden state to another.
    - Emissions: Chances of seeing an observation from a hidden states.
    - Initial State: Starting probabilities of the hidden states.

HMM Chain

- A process governed by a Hidden Markov Model (HMM) where:
    - `States are hidden`
    - `Observable outcomes`
    - `Markov property` : The current state depends only on the immediate previous state, not the full history.
- It is called a "chain" because the hidden states and their transitions form a sequential chain-like structure.



Applications of HMMs

- Speech Recognition
- Part of Speech Tagging
- Time Series Analysis
- Bioinformatics
- Anomaly Detection

Undirected Graphical Models and Variable Elimination

- `Undirected Graphical Models (UGMs)` , also known as `Markov Random Fields (MRFs)` , are probabilistic models that represent the relationships between random variables using an undirected graph. In these models:
    - `Nodes` : Represent random variables.
    - `Edges` : Represent direct probabilistic dependencies between the variables.
- `Variable Elimination` :
    - An algorithm used in `UGMs` (and other probabilistic models) for efficiently computing marginal distributions or performing inference.
    - A method for marginalizing out variables from a joint probability distribution.

- Instead of computing the full joint distribution, intermediate results are reused, avoiding redudant calculations.

Chapter 7: Time Series Analysis

What's Time Series Analysis?

- A method of analyzing a collection of data points over a period of time.
- Instead of recording data points intermittently or randomly, time series analysts record data points at consistent intervals over a set period of time.
- **Data collected**:
  - Time series data: It is a collection of observations on the values that a variable takes at various points in time.
  - Cross-sectional data: Data from one or more variables that were collected simultaneously.
  - Pooled data: It is a combination of cross-sectional and time-series data.
- The variable varies according to the probability distribution, showing which value Y can take and with which probability those values are taken:

$$Y_t = \mu t + \varepsilon t$$

  - Each instance of Y_t is the result of the signal μt,
  - εt is the noise term here.

Reasons to use time series analysis

- **Features**: Time series analysis can be used to track features like:
  - `Trend` : increasing or decreasing pattern.
  - `Seasonality` : cyclic patterns are the ones that repeat after a certain interval of time.
  - `Variability`
- **Forecasting**: Time series analysis can aid in the prediction of stock prices.
- **Inferences**: You can predict the value and draw inferences from data using Time series analysis.
- **Non-stationary data**: Time series analysis is used to track data that is constantly fluctuating over time or is affected by time.

Application of Time series analysis

- Rainfall measurements
- Automated stock trading
- Industry/weather forecast
- Temperature readings
- Sales forecasting

Models of Time series analysis

- **Classification**: It identifies and assigns categories to the data.

- **Curve Fitting**: It plots data on a curve to investigate the relationships between variables in the data.
- **Descriptive Analysis**: Patterns in time-series data, such as trends, cycles, and seasonal variation, are identified.
- **Explanative analysis**: It attempts to comprehend the data and the relationships between it and cause and effect.
- **Segmentation**: It splits the data into segments to reveal the source data's underlying properties.

Autoregressive Integrated Moving Average (ARIMA)

- Also known as `The Box Jenkins` method.
- A popular statstical methid used for time series analysis forecasting.
- Well suited for short-term forecasts and particularly useful when data show evidence of non-stationarity, where the statistical properties of the process generating the data change over time.
- **Parameters (Components)**:
    - Autoregressive Component:
        - Represents the number of lag observations included in the model.
        - It captures the relationship between an observation and a specified number of lagged observations.
        - Denoted by p:
            - When the value of p is 0, it means there is `no correlation` in the series.
            - When the value of p is 1, it means that the `auto-correlation` is up to one lag.
    - Moving Average:
        - Represents the size of the moving average window, which incorporates the dependency between an observation and a residual error applied to lagged observations.
        - Denoted by q, When $q = 1$, it means that there is an `error term`.
    - Integration:
        - This is the number of times the raw observations are differenced to make the time series stationary.
        - `Stationarity` means that the `statistical properties` of the series like `mean` and `variance` do not change over time.
        - Integration is denoted by d. When the value of d is 0, the series is `stationary`.
        - When the value of d is 1, the series is `not stationary`, and you can make it stationary by taking the difference.

How ARIMA works?

- `ARIMA` models both the trends and random fluctuations in the data to forecaast a future point in the series.
- The goal is to describe the autocorrelations in the data with an `ARIMA` model.
- **Steps**:
    - Identification:

        - Determine whether the time series is `stationary` and identify the order of differencing (d) needed.
        - Use plots and statistical tests (like the `Augmented Dickey-Fuller` test) to determine stationarity.
    - Estimation:
        - Choose the lag values for the `AR` (p) and `MA` (q) components.
        - This can be done by examining autocorrelation and partial autocorrelation plots.
    - Model Fitting:
        - Using statistical software, fit the ARIMA model to the data.
        - This involves estimating the parameters that best adapt the model to the historical data.
    - Diagnostic Checking: After fitting the model, check the `residuals` to ensure there are no patterns (meaning residuals are `white noise`).
    - Forecasting: Use the model to forecast future values, adjusting back the differencing if necessary.

AR and MA models

- The `AR model` only depends on past values (lags) to estimate future values.

$$AR(p) : x_t = \alpha + \sum_{i=1}^{p} \beta_i x_{t-i} + \epsilon$$

- The value p determines the number of past values p will be taken into account for the prediction.
- The higher the order of the model, the more past values will be taken into account.
- The `MA model` , on the other hand, depends on past forecast errors to make predictions.

$$x_t = \mu + \sum_{i=1}^{q} \Phi_i \epsilon_{t-1}$$

- The MA model can simply be thought of as the linear combination of q past forecast errors.

Stationary vs Non-Stationary Data

- **Stationary Data**:
    - The statistical properties (e.g., `mean` , `variance`) remain constant over time, and the `time series does not depend on the time variable` .
    - This consistency allows for predictive models that rely solely on past values, leading to more reliable predictions.
- **Non-Stationary Data**:
    - The statistical properties change over time, often exhibiting `trends` , `seasonality` , or `varying variances` .
    - This dependency on the time variable can result in less reliable predictions if not properly accounted for.

Seasonal Decomposition in Time Series Analysis

- A technique used in time series analysis to decompose the observed data into several components, typically including `trend`, `seasonality`, and `residuals` (or irregular components).
- This method helps in understanding the underlying patterns of the time series, which can improve forecasting and provide insights into the dynamics of the data.
- It's used to represent a time series as a `sum` or `product` of three components:
  - The `linear trend`,
  - The `periodic (seasonal) component`,and
  - The `random residuals`.
- The seasonal decomposition is useful in analysis of time series affected by factors that change in time in a cyclic (periodical) manner.
- Any periodical component with a known period may be described in this way.
- **Components**:
  - Trend:
    - Reflects the long term progression of the series(increasing or decreasing).
    - Shows the general pattern of data over a long period, unaffected by short-term fluctuations.
  - Seasonal:
    - Exhibits the repeating short-term (hourly, daily, weekly, monthly, and yearly) cycle in the series.
  - Residual:
    - The unexpected event left after the trend and seasonal components are removed from the data.
    - Random or irregular fluctuations that aren't accounted for by the seasonal and trend components.
- **Methods**:
  - Let's consider S_t is the seasonal component, T_i is the trend cycle component, and R_t is the reminder component.
  - `Additive decomposition`: appropriate if the magnitude of the seasonal fluctuations, or the variation around the trend-cycle, does not vary with the level of the time series.
$$y_t = S_i + T_i + R_i$$
  - `Multiplicative decomposition`: appropriate when the variation in the seasonal pattern, or the variation around the trend-cycle, appears to be proportional to the level of the time series.
$$y_t = S_i \times T_i \times R_i$$

Steps in Seasonal Decomposition

1. **Determine the Model**: Decide whether to use an additive or multiplicative model based on the nature of the seasonal effect's variation with the level of the time series.

2. **Estimate the Trend**: This can be done using methods like moving averages or fitting a smoothing curve.
3. **Estimate the Seasonal Component**: Once the trend is removed, the seasonal component can be estimated by averaging, for each time unit, over all periods.
4. **Calculate the Residuals**: The residuals are calculated by removing the trend and seasonal components from the original data (depending on whether the model is additive or multiplicative).
5. **Analysis and Interpretation**: Analyze the components to gain insights into the characteristics of the data.

Software and Tools that perform seasonal decomposition

- R: The `stl()` function in R provides a robust method for decomposing a time series into seasonal, trend, and irregular components, especially for handling the additive decomposition.
- Python: The `seasonal_decompose()` function from `statsmodels.tsa.seasonal` can perform both additive and multiplicative decomposition of time series data.

Chapter 8: Neural Networks

What is a Neural Network?

- A collection of `neurons` , or `nodes` linked together in a fashion that mimics the human brain.

Importance of Neural Networks

- `Artificial Neural Networks (ANN)` learn very efficiently and adaptively which makes them have the capability to learn `how` so solve a specific problem from the training data it receives.

How Do Neural Networks Learn?

- During training, the network is fed data that is already `labeled` with correct answers.
- It makes predictions based on its current state of weights and biases, compares these predictions to the correct answers, and then adjusts its weights and biases to improve its predictions.
- This process is repeated many times, and the network improves gradually.

Training Process

1. **Forward Propagation**: The data is passed through the network to generate an output.
2. **Loss Function**: This calculates the difference between the network's prediction and the actual outcome.
3. **Backpropagation**: This process adjusts the weights and biases of the network to minimize the loss.
4. **Optimization Algorithms**: These are methods like gradient descent that guide the adjustment of weights to reduce the loss most efficiently.

Types of Learnings in Neural Networks

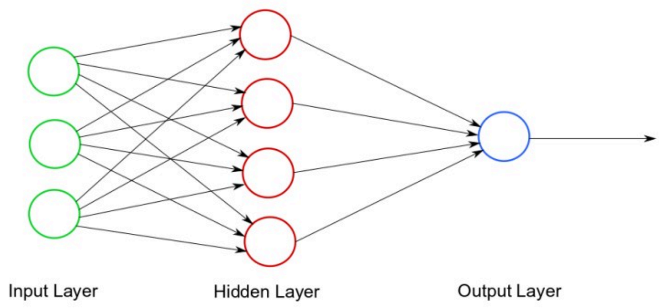
- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Applications of Neural Networks

- Image Recognition
- Speech Recognition
- Natural Language Processing (NLP)

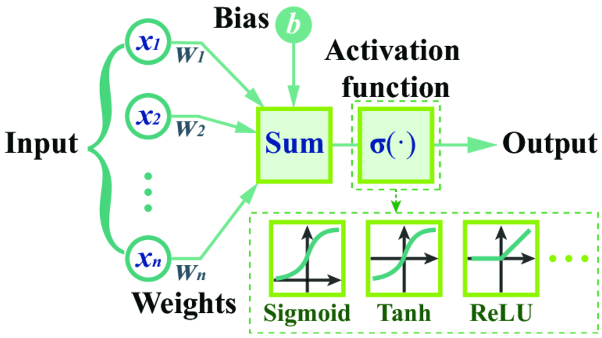
How Does a Neural Network work?

- **Layers**:
  - Input Layer: Receives raw data, like pixels from an image.
  - Hidden Layers: Perform the bulk of the processing, often multiple layers are stacked.
  - Output Layer: Produces the final result, like a classification
- **Connections**:
  - Each neuron connects to others in the next layer with associated weights.
  - `Weights` determine the influence of one neuron on another.



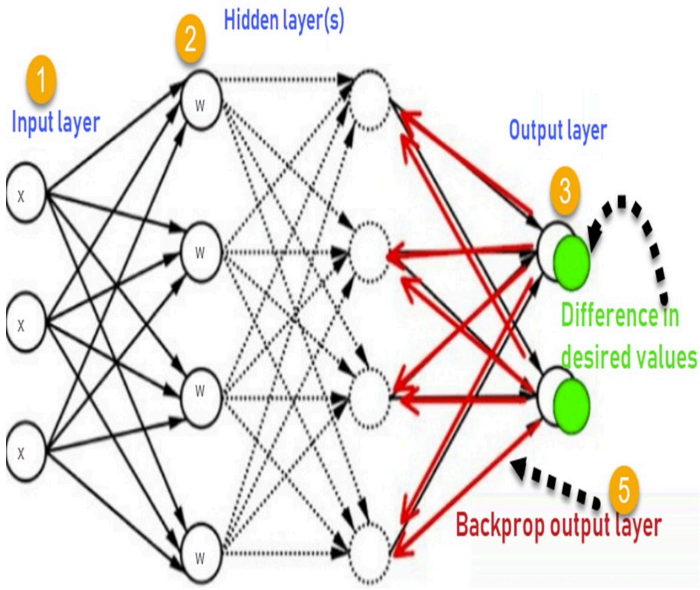
Forward Propagation

- **Input Layer**: Each feature in the input layer is represented by a node on the network, which receives input data.
- **Weights and Connections**:
  - The weight of each neuronal connection indicates how strong the connection is.
  - Throughout training, these weights are changed.
- **Hidden Layers**:
  - Each hidden layer neuron processes inputs by multiplying them by weights, adding them up, and then passing them through an activation function.
  - By doing this, non-linearity is introduced, enabling the network to recognize intricate patterns.
- **Output**: The final result is produced by repeating the process until the output layer is reached.



Back Propagation

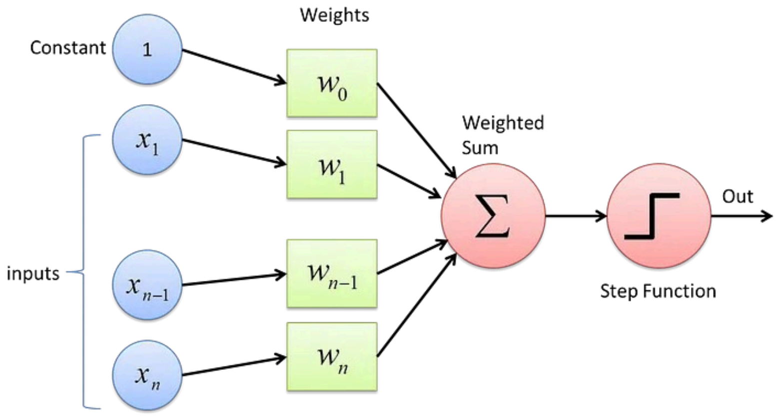
- **Loss Calculation**:
  - The network's output is evaluated against the real goal values, and a loss function is used to compute the difference.
  - For a regression problem, the `Mean Squared Error (MSE)` is commonly used as the cost function.
- **Loss Function**: MSE
- **Gradient Descent**:
  - Gradient descent is then used by the network to reduce the loss.
  - To lower the inaccuracy, weights are changed based on the derivative of the loss with respect to each weight.
- **Adjusting weights**: The weights are adjusted at each connection by applying this iterative process, or backpropagation, backward across the network.
- **Training**: During training with different data samples, the entire process of forward propagation, loss calculation, and backpropagation is done iteratively, enabling the network to adapt and learn patterns from the data.
- **Actvation Functions**:
  - Model non-linearity is introduced by activation functions like the rectified linear unit (ReLU) or sigmoid.
  - Their decision on whether to "fire" a neuron is based on the whole weighted input.



Types of Neural Networks

- **Feedforward Networks(Perceptron)**:

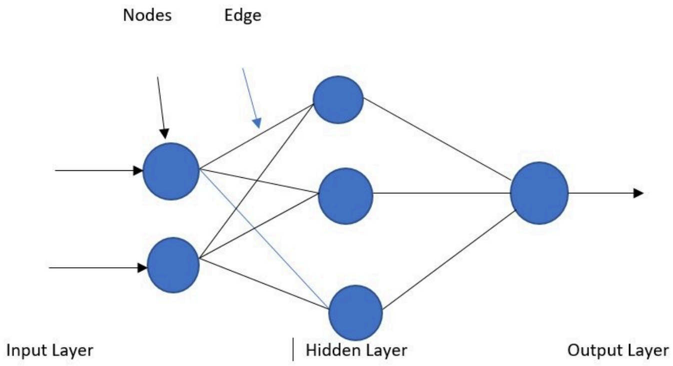
- A simple artificial neural network architecture in which data moves from input to output in a single direction (single layer).
- It has `input`, `hidden`, and `output` layers; `feedback loops` are absent.
- Appropriate for regression and pattern recognition.
- Perceptrons can only learn linear relationships between inputs and outputs.



- Steps:
  - All the inputs x are multiplied with their weights w.
  - Add all the multiplied values and call them Weighted Sum.
  - Apply that weighted sum to the correct `Activation Function` which are used to map the input between the required values like (0, 1) or (-1, 1).

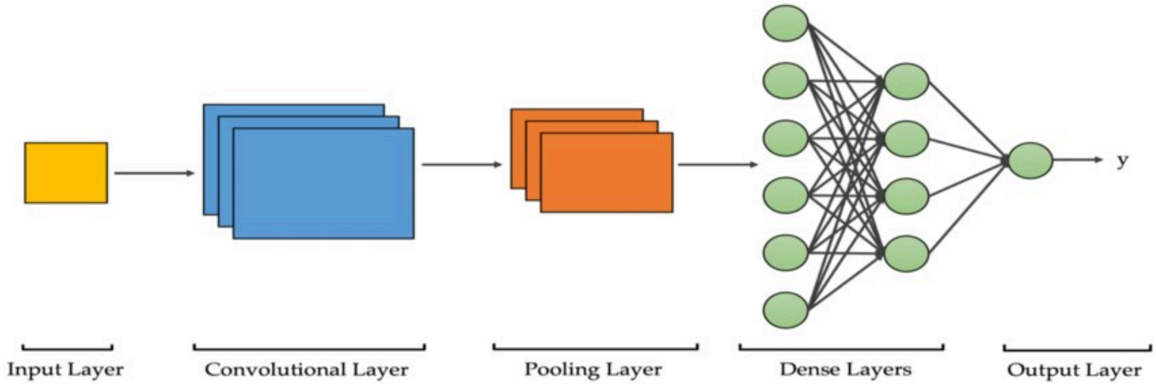
- **Multilayer Perceptron (MLP)**:
  - A type of `feedforward neural network` with three or more layers, including an `input` layer, one or more `hidden` layers, and an `output` layer.
  - It uses `nonlinear activation functions`.
  - A more complex network with multiple hidden layers between the input and output layers.
  - It uses a BackPropagation algorithm for training the model.
  - A class of Deep Learning.
  - Layers:
    - `Input Layer`:
      - It is the initial or starting layer of the Multilayer perceptron which takes input from the training data set and forwards it to the hidden layer.
      - There are n input nodes in the input layer which depends on the number of dataset features.
      - Each input vector variable is distributed to each of the nodes of the hidden layer.
    - `Hidden Layer`:
      - Performs the main computations of the neural network.
      - Connections between nodes (edges) have weights that influence the computations.
      - Activation functions are applied to introduce `non-linearity`.
      - The number of hidden layers and nodes must be carefully chosen to balance model `complexity` and `performance`, `avoiding underfitting` or `overfitting`.
    - `Output Layer`:
      - Produces the final result of the model.

- The number of nodes depends on the task: one for regression, or multiple for classification tasks with multiple classes.



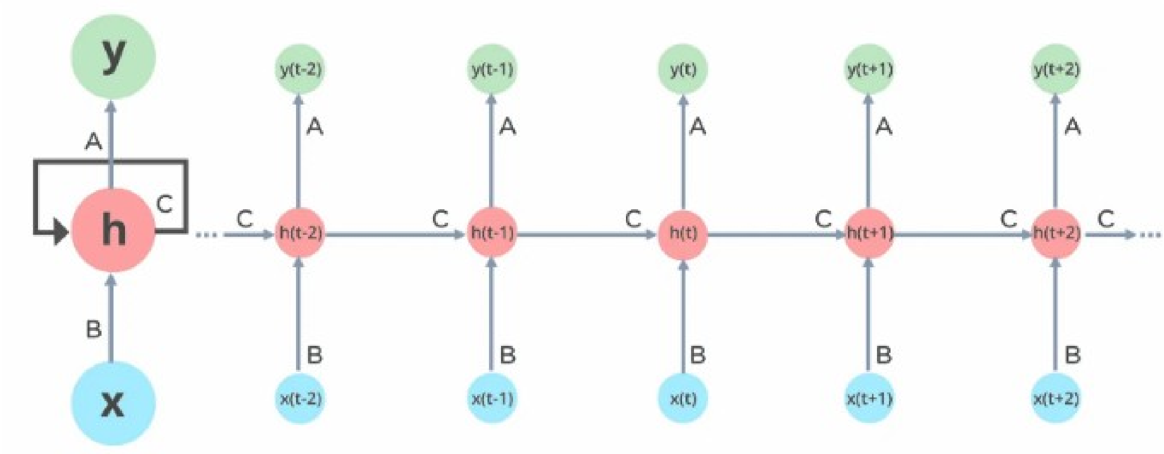
- **Convolutional Neural Network(CNN)**:
  - A specialized deep learning models designed for tasks like `image classification` and `object recognition`.
  - They process input images through multiple layers to extract and analyze features, ultimately producing a predicted output.
  - Layers:
    - Convolutional Layer:
      - The convolutional layer is responsible for extracting features from the input image.
      - It performs a convolution operation on the input image, where a filter or kernel is applied to the image to identify and extract specific features.
    - Pooling Layer:
      - The pooling layer is responsible for reducing the spatial dimensions of the feature maps produced by the convolutional layer.
      - It performs a down-sampling operation to reduce the size of the feature maps and reduce computational complexity.
    - Activation Layer:
      - Applies a non-linear activation function, such as the ReLU function, to the output of the pooling layer.
      - This function helps to introduce non-linearity into the model, allowing it to learn more complex representations of the input data.
    - Fully Connected Layer:
      - A traditional neural network layer that connects all the neurons in the previous layer to all the neurons in the next layer.
      - This layer is responsible for combining the features learned by the convolutional and pooling layers to make a prediction.
    - Normalization Layer: Performs normalization operations, such as batch normalization or layer normalization, to ensure that the activations of each layer are well-conditioned and prevent overfitting.
    - Dropout Layer:
      - The dropout layer is used to prevent overfitting by randomly dropping out neurons during training.
      - This helps to ensure that the model does not memorize the training data but instead generalizes to new, unseen data.

- Dense Layer:
  - Final layer in a Convolutional Neural Network (CNN), combines the extracted features from convolutional and pooling layers to make the final prediction.
  - It works by:
    - Flattening the activations from previous layers into a 1D vector.
    - Performing a weighted sum of the inputs.
    - Applying an activation function to generate the final output.
  - The dense layer plays a crucial role in producing the final predictions in tasks such as classification or regression.



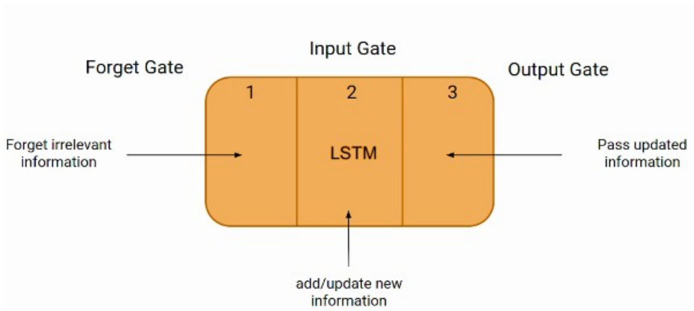
- Benefits:
  - Feature extraction
  - Spatial invariance
  - Robust to noise
  - Transfer learning
  - Performance
- Limitations:
  - Computational cost
  - Overfitting
  - Lack of interpretability
  - Limited to grid like structures

- **Recurrent Neural Network (RNN)**:
  - Neural Network designed for sequential data processing, making them ideal for tasks like time series prediction and natural language processing.
  - They use feedback loops to retain information from previous steps, enabling the network to consider contextual dependencies and influence future predictions
  - RNNs outperform standard neural networks when working with sequential data, such as text or time series.
  - The input layer takes in data and passes it to the middle (hidden) layers for processing.
  - The hidden layers in a standard neural network may have unique activation functions, weights, and biases without memory of previous layers.
  - In a Recurrent Neural Network (RNN):
    - The same parameters (activation functions, weights, and biases) are standardized across layers.
    - Instead of multiple distinct hidden layers, RNNs use a single hidden layer that loops over sequential data as many times as needed, enabling memory and contextual processing.



- RNN were created because there were a few issues in the feed-forward neural network:
    - Cannot handle sequential data,
    - Considers only the current input and
    - Cannot memorize previous inputs.
- Advantages:
    - Ability To Handle Variable-Length Sequences
    - Memory Of Past Inputs
    - Parameter Sharing
    - Non-Linear Mapping
    - Sequential Processing
    - Flexibility
    - Improved Accuracy

- **Long Short-Term Memory (LSTM)**:
    - Introduce a concept called a `memory cell`.
    - This cell acts like a conveyor belt that can store information for long periods.
    - Crucial information from past inputs can persist in the cell, while irrelevant details can be selectively forgotten.
    - To control the flow of information through the cell, LSTMs use three special gates:
        - Forget Gate: Decides what information to forget from the cell's current state.
        - Input Gate: Determines what new information to store in the cell.
        - Output Gate: Controls what information from the cell state to include in the output of the LSTM.



- Steps:
    - `Input Layer`: Receives the current input in the sequence.
    - `Forget Gate`:
        - Analyzes the previous cell state and current input.

        - It outputs a value between 0 and 1 for each element in the cell state, indicating how much to forget (0) or retain (1).
    - `Input Gate`:
        - Processes the previous cell state and current input.
        - It generates a candidate value for the new information to be stored in the cell and another value deciding which parts of the candidate value to incorporate.
    - `Cell State Update`:
        - The forget gate's output is multiplied by the previous cell state to determine what to forget.
        - The input gate's output is multiplied by the candidate value to decide what new information to add. These are summed to update the cell state.
    - `Output Gate`:
        - Examines the current cell state and prior hidden state.
        - It outputs a value between 0 and 1 for each element in the cell state, determining how much of the cell state to include in the output.
    - `Hidden Layer`:
        - The output gate's output is multiplied by the cell state to create the output of the LSTM cell, which is then passed to the next LSTM cell in the sequence or used for prediction.
- Benefits:
    - Learning Long-Term Dependencies
    - Selective Memory