MACHINE LEARNING

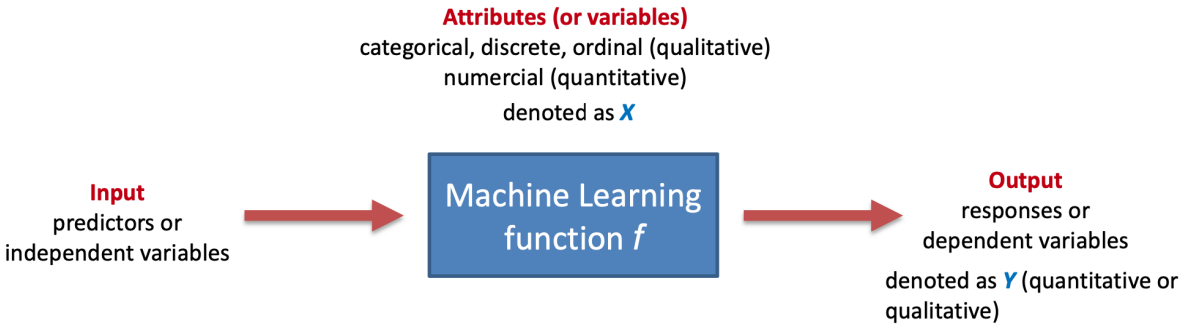
Topic 1: Model Evaluation

Supervised Learning: Training the data with desired output values that are provided before training.
Unsupervised Learning: Training the data without desired output values
Semi-supervised Learning: Training the data with small amounts of desired values, iteratively training the model using the most confident predictions
Active Learning: Training the data with and without the labels. Interactive labeling of training data by asking a human expert to label the least confident data samples.
Federated Learning: Training data is localized, and learn a central model suing distributed local data without sharing the data.
Reinforcement Learning: An agent learn by interacting with the environment through actions that are either rewarded or penalized.

Active learning vs Supervised learning: The main difference is how they use the labeled data. In supervised the labeled data is used to train the algorithm, and in active learning, the labeled data is selectively used to improve the algorithm.

Semi-supervised vs Unsupervised: The main difference is how they use the labeled data. In semi-supervised, some labeled data is used to guide the learning process, and in unsupervised n, labeled data is used.

Machine Learning Process:



The objective of machine learning is not to build a function that will overfit, but a set of predictors tries to learn ways to distinguish two or more classes, and then it can also have a target task, either classification or regression.

Classification is the task of learning a target function f that maps an input X to one of the predefined class labels Y.
Regression is the task of learning a target function f that maps an input X to a value in a range of real values Y.

Bias variance Decomposition: A technique used to analyze the expected prediction error of a machine learning model. The expected prediction error is the average difference between the predicted and actual values of the target variables.

- Assume the data complies to the following statistical model:

$$Y = f(X) + \varepsilon$$

f is a function, attribute set X, and produces output Y, and epsilon is the error

- The relationship between X and Y is *non-deterministic (noise)*
- But we assume:

$$\mathrm{E}(\varepsilon) = 0 \qquad \mathrm{Var}(\varepsilon) = \sigma_\varepsilon^2$$

E(e) is the expected noise to be zero, but we have some variance Var(e), which is some value greater than 0. And the function f is fixed that doesn't change.

- The metric to optimize is MSE (mean squared error):

$$\mathrm{MSE} = \mathbb{E}[(y - \hat{f}(x))^2]$$

We use one metric to see how well we perform, MSE(Mean squared Error). The difference between the predicted value and the value that we expect to predict and square it up, and we take the average.

- Bias: over different realizations of the training set, how much does $\hat{f}(X)$ change:

$$\mathrm{bias}[\hat{f}(x)] = \mathbb{E}[\hat{f}(x)] - f(x)$$

F hat (X) is the predictions, so over different realizations of the training set, how much do predictions change.

So, bias is the Expectation (E) of the predicted function(f hat X) minus the actual function that we are trying to find.

- Variance: over different realizations of the training set, how much does $\hat{f}(X)$ deviate from its expectation:

$$\text{var}(\hat{f}(x)) = \mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2]$$

F hat (X) is the predictions, so over different realizations of the training set, how much do predictions deviate from its expectations.
So, variance is we take the f hat and its expectations and square them. So we check how much this f changes when using a different training set.

The bias-variance decomposition says that the Expectation of the MSE is the sum of the expectation of the bias square, the variance, and the standard deviation of the error.

Topic 2: Regression Analysis:

What is regression, and how can we solve it: Regression is used to predict a continuous output variable based on one or more input variables. The goal is to establish a relation between the input variables(independent variables or predictors)and output variables(dependent or response). The formula for simple linear regression is Y= mX+b, where Y is the response variable, X is the independent variable, where m is the estimated slope, and b is the estimated intercept.

Linear Regression: The supervised Machine learning model in which the model finds the best fit linear line between the dependent and independent variables.
How do we fit: We attempt to find the best-fit line by minimizing the difference between the actual and predicted. But positive differences can offset the negative ones. That's why we take the squared differences.

Logistic Regression: It is used to calculate the probability of a binary(yes/no) event occurring. The output can be only 0 and 1. The goal is to find the S curve by which we can classify the data. We can fit it by using the method called maximum likelihood. It maximizes the likelihood function.

What is the connection between linear and logistic regression: They both are supervised machine learning algorithms, and both of them use the linear equation for prediction. Linear regression is

used to handle the regression problems, whereas logistic regression is used to handle the classification problems.

What is Regularization, and why do we need it: Techniques that are used to calibrate machine learning models in order to minimize the adjusted loss. To avoid overfitting, and underfitting, we need Regularization.
Ridge regression: Shrinks the regression coefficients by imposing a penalty on their size. The ridge coefficients minimize a penalized residual sum of squares. This shrinks the coefficients towards zero (L2 ridge penalty), and it helps to reduce the model complexity and multi-collinearity.

Lasso Regression: It is a shrinkage method like ridge. The only difference is instead of taking the square of the coefficients; magnitudes are taken into account. This type of regularization (L1) can lead to zero coefficients, i.e., some of the features are completely neglected for the evaluation of output. This performs feature selection.

What regression do we use for feature selection and why: Lasso regression because discarding a feature will make its coefficient equal to 0.

Topic 3: Ensemble:

What are an ensemble and its goal: Ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. The main goal is to improve the performance of a model or to reduce the likelihood of an unfortunate selection of a poor one.

How is bias-variance tradeoff addressed by ensemble model: By combining multiple models to improve overall prediction and reduce overfitting. An ensemble model can reduce the variance by combining multiple models that are trained on different subsets of data, thereby reducing the likelihood of overfitting. An ensemble model can reduce the bias by combining multiple models that are trained on different algorithms to reduce the risk of making overly simple assumptions.

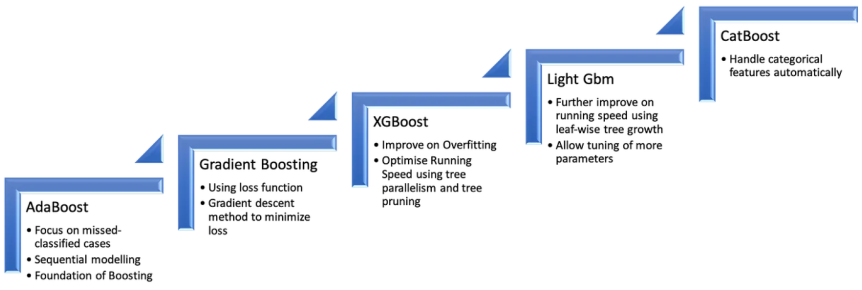
Difference between Bagging and Boosting: Bagging involves fitting many decision trees on different samples of the dataset and averaging the predictions. Boosting involves adding ensemble members sequentially to correct the predictions made by prior models and outputs a weighted average of the predictions.

Differences Between Bagging and Boosting

S.NO	Bagging	Boosting
1.	The simplest way of combining predictions that belong to the same type.	A way of combining predictions that belong to the different types.
2.	Aim to decrease variance, not bias.	Aim to decrease bias, not variance.
3.	Each model receives equal weight.	Models are weighted according to their performance.
4.	Each model is built independently.	New models are influenced by the performance of previously built models.
5.	Different training data subsets are randomly drawn with replacement from the entire training dataset.	Every new subset contains the elements that were misclassified by previous models.
6.	Bagging tries to solve the over-fitting problem.	Boosting tries to reduce bias.
7.	If the classifier is unstable (high variance), then apply bagging.	If the classifier is stable and simple (high bias) the apply boosting.
8.	Example: The Random Forest model uses Bagging.	Example: The AdaBoost uses Boosting techniques

Brief Differences between Bagging and Boosting

How does XG Boost work, and why is it powerful: XG Boost is an efficient implementation of gradient boosting that can be used for regression predictive modeling. XG Boost minimizes the regularized(L1, L2) objective function that combines a convex loss function and a penalty term for model evaluation. XG Boost performs better than Random Forest when we have a class imbalance. XG Boost always gives more importance to functional space when reducing the cost of a model, while RF tries to give more preference to hyperparameters to optimize the model. The disadvantage is it cannot perform on sparse or unstructured data.



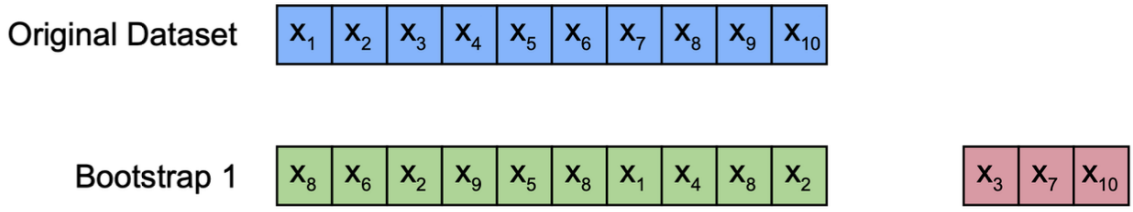
Comparing Models

Topic 4: Model Evaluation:

Cross-validation and Nested cross-validation: Cross-validation is used to evaluate a model's performance by partitioning the data into subsets and training the model on some of them while validating it on others. Nested cross-validation is a more advanced technique that further divides the data into training and validation sets and uses an inner loop to tune the model's hyperparameters. It provides a more reliable estimate of the model's performance on unseen data while ensuring the model is properly tuned.

Bootstrapping: Create B replications(of the same size) of a training set of size N by selecting examples uniformly with replacement. Train on each replication b and test on the whole training set.

Leave-one-out bootstrap: Train on each replication b and test each sample, not in b.
An example is below:



Why leave one out of bootstrap: To address the problem of limited data in statistical analysis. When the sample size is small, it can be difficult to estimate the availability of a statistic or the accuracy of a predictive model. This provides a way to estimate these quantities by repeatedly resampling the data and computing the static or fitting the model on each sample.

The VC dimension: This is a measure of the capacity of a set of functions that can be learned by a statistical binary classification algorithm. It is used frequently to guide the model selection process while developing machine learning applications.

Topic 5: CNNs:

Main components of CNN and how they are structured: There are three main layers in the CNN Convolution layer, Pooling, and Fully connected layer.

Convolution Layer: It has a number of filters that perform the operation of convolution. They consist of a set of learnable filters that slide across the input image, producing feature maps that capture the presence of specific visual patterns or features. Each filter is a small matrix of weights that are learned during training. The output of each filter is passed through a non-linear activation function such as ReLU to introduce non-linearity.

Pooling layer: The pooling layers are used to reduce the spatial size of the feature maps generated by the convolution layers while retaining important information. This is done by applying pooling operations, such as max-pooling or average pooling, to a small region of the feature map.

Fully connected layers: Are used to classify the features learned by the convolution and pooling layers. They consist of a set of neurons that are connected to all the neurons in the previous layer. The output is passed through a softmax activation function to generate class probabilities.

Dropout: This is a regularization technique used to prevent overfitting. It randomly drops out a fraction of the neurons in the network during training.

What is 1x1 convolution, and why do we need it: It is a convolution with some special properties. It can be used for dimensionality reduction, efficient low-dimensional embeddings, and applying nonlinearity after convolutions. They have an effect when the image has multiple channels, and Dimensionality reduction reduces the number of channels to the number of 1x1 filters.

What happens if we remove the pooling layer: The spatial size of the feature maps would remain the same as the input image, which could result in a large number of parameters in the network and increase the computational cost. And the network may become more sensitive to small translations in the input image, which could reduce its performance on some tasks. In some cases, it can help to preserve more spatial information in the feature maps, which could be useful for tasks such as object localization or segmentation. In some cases, replacing the pooling layer with a stride in the convolution layer can also achieve similar spatial dimensionality reduction.

What happens if we drop the convolution layer: The training process can be slow.

What happens if we drop the Fully connected layer: It is only done on the fully connected layer because it has a lot of parameters, and they might also cause overfitting by co-adapting themselves.

Pretrianed CNN:
Pretrain a neural network model on a source dataset.

- Create a new neural network model by
    - Copying the pre-trained model
    - Removing the output layer
    - Adding an output layer
    - Initializing randomly the model parameters of the new output layer

- Train the new model on the target dataset
- The output layer will be trained from scratch, while the parameters of all other layers are fine-tuned based on the parameters of the source model.

Why is deep learning popular: It minimizes human action therefore, its algorithms conduct feature extraction on themselves.

Backpropagation: This is used to compute the gradient of the loss function with respect to the weights and biases of the network.

- Treat unfolded network as one big feed-forward network
- This network takes in the whole sequence as input
- We compute the gradients through the usual back-propagation
- We update the shared weights

Gradient descend: It is an optimization algorithm, to update the weights and biased of the network during training. The goal is to minimize the cost function or the loss function of the network. Step involved:

- Initialize the weights and biases of the network with random values.
- Forward pass: Pass an input through the network to compute the output.
- Compute the loss: compute the difference between the predicted output and the true output
- Backward pass: Compute the gradient of the loss with respect to the weights and biases of the network using backpropagation.
- Update the weights and biases: Use the computed gradients to update the weights and biases of the network in the direction of steepest descent using a learning rate hyperparameter.
- Repeat steps 2-5 for multiple iterations until the network converges, or a maximum number of interactions is reached.

Topic 6: RNNs + Transformers + Attention:

What are the main differences between RNN LSTM and GRU:

- Architecture: All three are based on the same architecture, but they differ in the way they process and store information.
    - RNN: Simple structure with a single layer of neurons
    - LSTM and GRU: Has more complex architecture with multiple layers
- Memory capacity: LSTM and GRU have high memory capacity compared to RNN, which makes them better suited for a long sequence of data. They can selectively store information, while RNNs have a tendency to forget the older information.

- Gating mechanisms: LSTM and GRU have a gating mechanism that allows them to selectively control the flow of information between the different layers of the network. LSTM has three gates, input, output, and forget, and GRU has reset and update.
- Training time: LSTM and GRU have a longer training time compared to RNN because of their complex memory architecture and additional parameters. However, they achieve better performance on complex tasks that require the processing of a long sequence of data.

What if we have a short sequence of data: We will select a simple RNN architecture, which may be sufficient to process the data and make accurate predictions. This is because RNNs have lesser parameters, making them faster to train and less prone to overfitting when dealing with smaller datasets.

What do gates do in LSTM: We have three gates in LSTM

- Forget gate: Takes as input the previous hidden state and the current input, and decides which information from the previous hidden state should be forgotten.
- Input gate: This gate decides which information from the current input should be added to the memory.
- Output gate: This gate decides which information from memory should be used to compute the output of the current time step.

What do gates do in GRU: We have two gates in GRU:

- Reset gate: This gate determines how much of the previous hidden state should be forgotten.
- Update gate: This gate determines how much of the current input should be added to the current hidden state.

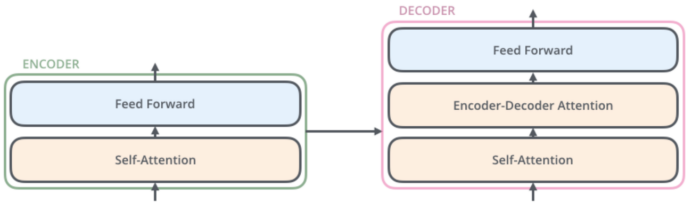
Difference between LSTM and GRU: Write about gates and:
GRU have lesser parameters compared to LSTM, which makes it a bit faster than LSTM and less prone to overfitting. However, LSTM is considered the most powerful than GRU and is better suited for tasks that require the processing of a longer sequence of data.

How RNN is applied to predict the next word in a sentence: To predict the next word in a sentence, a common approach is used called teacher forcing. This involves training the RNN to predict the next word in the sequence based on the previous words in the sequence that have already been seen. During the training, the input to the RNN is the previous word in the sequence, and the output is the predicted next word. The predicted next word is then fed back into the RNN as input for the next time step.

What is the benefit of attention, and how is it applied: Attention allows the model to focus on specific parts of the input sequence when making a prediction. The main benefit of attention is

that it enables the model to selectively attend to different parts of the input sequence, giving it the flexibility to handle long and complex input sequences without losing information. This lead to better accuracy and more efficient training compared to models that do not use attention. To apply attention, the decoder layer calculates the attention weights that indicate how much attention should be given to each hidden state produced by the encoder. These attention weights are then used to compute a weighted sum of the encoder's hidden states, which is used as input to the decoder at each time step.

Attention between encoder-decoder:



Encoders: all are identical in structure, but they won't share the same weights.

- Inputs flow through a self-attention layer: a layer that helps the encoder look at other words in the input sentence as it encodes a specific word.
- Outputs of the self-attention layer are fed to a feed-forward neural network
- The exact same feed-forward network is independently applied to each position

Decoders: has both those layers, but in between them there is an attention layer that helps the decoder focus on relevant parts of the input sequence. By allowing the decoder to attend to different parts of the input sequence at each time step, attention can significantly improve the accuracy of the model, particularly for long input sequences. Additionally, attention can help to improve the interpretability of the model, by highlighting which parts of the input sequence are most relevant for each element of the output sequence.

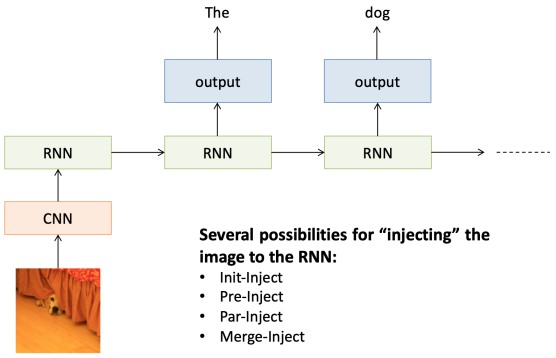
What is self-attention: It allows the model to attend different positions in the input sequence in order to compute a representation for each element of the sequence. It allows the model to attend to different parts of the same sequence, rather than attending to different sequences.
Step 1: Define three vectors Query vector, Key vector, and Value vector
Step 2: calculate the self-attention score: the dot product of the query vector and key vector of the respective word.
Step 3-4: Apply normalization by dividing the scores by 8
Step 5-6: Scaling and summation: multiply each value vector by the softmax score
Step 7: calculate the self-attention matrix

What are transformers and their main components: That adopt the mechanism of self-attention, differentially weighting the significance of each part in the input data.
The main components are

- Encoder: The encoder processes the input sequence and produces a set of hidden representations for each element of the sequence.
- Decoder: The decoder generates the output sequence based on the hidden representations produced by the encoder.
- Self-attention: It allows the model to attend different positions in the input sequence in order to compute a representation for each element of the sequence. It allows the model to attend to different parts of the same sequence, rather than attending to different sequences.
- Encoder-Decoder attention: That allows the decoder to attend to different parts of the encoder output when generating each element of the output sequence.
- Positional Encoding: Used to inject information about the position of each element in the sequence into the model, in order to allow the model to capture the order of elements.

What happens if we drop a layer in transformers? Does it still work?: It is possible to drop certain parts of the architecture and still obtain useful results in some cases.
For example, if the task involves processing a short sequence of data, it may be possible to drop the encoder-decoder attention and only use self-attention with the encoder and decoder.
Similarly, if the input sequence is known to have a fixed length, it may be possible to remove the positional encoding.

If we have images and texts, in hand, we have some cnn architecture and LSTM. How do you use those together to solve the problem of image captioning:



One method is the encoder-decoder architecture. CNN is used as an encoder to produce a fixed-length vector representation of the image, and the LSTM is used as a decoder to generate a sequence of words that describe the image. Specifically, the encoder CNN is used to extract a set of features from the image, which are then fed into the LSTM decoder. The LSTM decoder

generates a sequence of words, one at a time, based on the features and the previous words in the sequence. At each time step, the LSTM calculates a set of attention weights over the image features, indicating which parts of the image are most relevant for generating the current word. In summary, to solve the problem of image captioning using both CNN and LSTM, one can use an encoder-decoder architecture with an attention mechanism, where the CNN is used to encode the image. The LSTM is used to generate the caption, while the attention mechanism is used to allow the LSTM to attend to different parts of the image.

Topic 7: Explainability:

What is Model Explainability: It is the ability to understand and interpret the decisions made by a machine learning model, It refers to the process of making a model's predictions and decision-making process transparent and understandable to humans.

There are several reasons why model explainability is important.

First, it can help build trust in the model by allowing users to understand why the model is making certain decisions. This is particularly important in high-stakes applications such as healthcare, where incorrect decisions could have serious consequences.

Second, model explainability can help identify and correct biases in the model. If a model is making decisions based on biased or discriminatory factors, it is important to be able to identify these factors and address them.

Third, model explainability can provide insights into the underlying data and help users understand the relationships between different variables. This can be particularly useful in fields such as science and engineering, where understanding the underlying processes is important.

Finally, model explainability can help improve the model itself. By understanding how the model is making decisions, researchers can identify areas where the model is performing poorly and work to improve it.

What are the main explainability models in machine learning: There are several main approaches to model explainability in machine learning.

- Feature Importance: This approach measures the importance of each feature or variable in the model's decision-making process. Feature importance can be determined using various methods, such as permutation feature importance or partial dependence plots.
- Decision Trees: Decision trees are interpretable models that make decisions by splitting the data based on the values of different features. The decision-making process of a

decision tree can be visualized and interpreted, making it a useful tool for model explainability.

- Local Interpretable Model-Agnostic Explanations (LIME): LIME is a model-agnostic approach to explainability that provides local explanations of individual predictions. LIME creates a simplified, interpretable model that approximates the original model in a small region around the prediction of interest.
- Shapley Values: Shapley values are a concept from cooperative game theory that can be used to explain the contribution of each feature to a particular prediction. Shapley values take into account all possible combinations of features and their contributions to the prediction.
- Integrated Gradients: Integrated gradient is a method that assigns an importance score to each input feature by integrating the gradients of the model's output with respect to the input feature over a given path. Integrated gradients provide a more nuanced explanation of feature importance than other methods.

These are just a few examples of the main approaches to model explainability in machine learning. The choice of approach will depend on the specific needs of the application and the type of model being used.

What is the trade-off between explainability and model performance: This trade-off occurs because interpretable models tend to have fewer parameters and rely on simpler algorithms, making them easier to understand but potentially less flexible in handling complex data. On the other hand, more complex models with a large number of parameters may be able to better fit the data and achieve higher performance but may be more difficult to interpret.

In some cases, it may be possible to balance the trade-off between model performance and explainability by using techniques such as feature selection, regularization, or simpler model architectures that can still achieve good performance while maintaining interpretability.

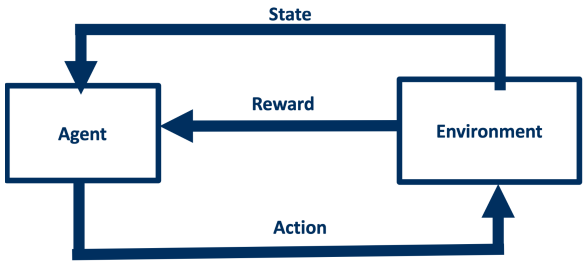
Ultimately, the choice of model will depend on the specific needs of the application. In some cases, such as in high-stakes applications like healthcare, explainability may be more important than performance, while in other cases, such as in image or speech recognition, performance may be the primary concern.

Topic 8: Reinforcement Learning:

What is RL: RL stands for Reinforcement Learning, which is a type of machine learning in which an agent learns to make decisions through trial and error interactions with an environment. In RL, the agent receives feedback in the form of rewards or penalties for its actions and adjusts its behavior accordingly to maximize its cumulative reward over time.

The process can be divided into three main components: The Environment, The Agent, and The Policy. The environment is the world in which the agent operates and receives feedback. The agent is the learner who interacts with the environment and makes decisions. The policy is the set of rules the agent uses to determine its actions based on its current state and the reward it has received.

What is Markov's Decision Process: It has both state and as well as dynamics between one state and another state. The next state is dependent on the current state. At each time step, the agent observes the current state and selects an action to take. The action leads the agent to a new state, and the process repeats until a terminal state is reached. The goal of Markov's Decision Process is to find a policy that maximizes the expected sum of discounted rewards over time. This can be done using various algorithms, such as value iteration or policy iteration.



Markov's Decision Process

What is Q-Learning, ad how does it work: It is used to find the optimal policy in Markov's Decision Process. It is based on the idea of learning a Q-function, which is a function that estimates the expected cumulative reward of taking a particular action in a particular state and following an optimal policy thereafter.
During the learning process, the agent interacts with the environment, observes the current state, selects an action based on its current Q-function estimate, receives a reward, and updates its Q-function. The agent continues to repeat this process until it has learned an optimal policy that maximizes its cumulative reward.

Policy Gradient Methods: The basic idea behind policy gradient methods is to use gradient ascent to update the policy parameters in the direction of a higher expected reward.
This is done by computing the gradient of the expected cumulative reward with respect to the policy parameters and using it to update the parameters at each time step.
Policy gradient methods have several advantages over value-based methods. They can handle both continuous and discrete action spaces, and they can learn stochastic policies that can explore multiple solutions.

However, they can be computationally expensive, and they may suffer from high variance in the gradient estimate. To address these issues, several extensions and improvements to policy gradient methods have been proposed, such as actor-critic methods and trust region optimization.

Off-policy vs. On-policy:
On-policy methods learn by interacting with the environment using the current policy, meaning that the agent learns from the experience of its own actions. The policy is updated based on the rewards obtained by following the policy during the interaction with the environment. The most common on-policy method is the SARSA algorithm, which updates the Q-values based on the current state, action, reward, next state, and next action.

Off-policy methods, on the other hand, learn by interacting with the environment using a different policy than the one being updated, meaning that the agent learns from the experience of actions taken under a different policy. The policy being updated is typically the one with the highest expected reward, rather than the one being followed during the interaction. The most common off-policy method is Q-learning, which updates the Q-values based on the current state, action, reward, and maximum Q-value over all possible next actions.