1. Difference between Supervised and Unsupervised Learning .

Supervised learning is a machine learning technique where the model is trained on labelled data, meaning the input data is associated with the desired output. In other words, the algorithm is given a set of input-output pairs and learns to map the inputs to the correct outputs. Supervised learning is used for prediction and classification tasks where the goal is to make accurate predictions on new, unseen data based on patterns learned from the labelled training data.

Unsupervised learning, on the other hand, is a machine learning technique where the model is trained on unlabelled data, meaning the input data is not associated with any desired output. In this case, the algorithm tries to find patterns or structure in the input data without any prior knowledge of what the output should be. Unsupervised learning is used for exploratory analysis and data visualization, as well as for clustering and anomaly detection.

1. Reinforcement Learning

Type of machine learning technique that enables an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences.

reinforcement learning uses rewards and punishments as signals for positive and negative behaviour. In Reinforcement learning the goal is to find a suitable action model that would maximize the total cumulative reward of the agent.

Key terms in RL:

Environment : Physical world in which agent operates .

State : Current Situation of the agent.

Reward : Feedback from the environment

Policy: Method to map agent’s state to actions

Value:Future reward that an agent would receive by taking an action in a particular state.

Take the case of student in a classroom. The environment is like the classroom itself - it's the physical world in which the student operates. The "state" is like the current situation the student finds themselves in - are they sitting at their desk, are they listening to the teacher, are they confused about something?

The reward is like the grade the student gets on a test or quiz. It's feedback from the environment about how well the agent is doing. If the agent makes a good decision, it might get a high reward. If it makes a bad decision, it might get a low reward.

The policy is like the strategy the student uses to study for a test or quiz. It's the method the agent uses to figure out what actions to take in each state.

Value is like the idea of getting a good grade on a future test or quiz. It's the idea of thinking about what might happen in the future if the agent takes a particular action in a particular state. For example, if the agent decides to study really hard for a test, it might get a good grade, which is a high-value outcome.

Imagine you're playing a game where you have to collect as many coins as possible. You know that some areas of the game have lots of coins, while others have very few. Now, you have two options:

Exploit: Go to the areas you already know have lots of coins and keep collecting them.

Explore: Try new areas of the game you haven't been to before, even if you're not sure how many coins are there.

If you only exploit, you might miss out on finding new areas with even more coins. But if you only explore, you might waste time going to areas with very few coins. This is called the Exploration vs Exploitation trade-off.

In order to make the best overall decision, the agent (or the player in the game) needs to find a balance between exploring new areas and exploiting the ones it already knows are good. This might involve making short-term sacrifices - like spending some time exploring even though you know there aren't many coins there right now - to get more information and make better decisions in the future

Q-learning and SARSA are two different ways that a computer program or agent can learn how to make good decisions in a given situation. They both work by using feedback from the environment to figure out which actions are good and which are bad.

One key difference between Q-learning and SARSA is how they explore the environment to figure out what actions to take. Q-learning tends to be more exploratory - it tries lots of different actions to see what works best. SARSA, on the other hand, tends to be more conservative - it sticks to the actions it already knows are good.

However, both Q-learning and SARSA are similar in terms of how they exploit the knowledge they already have. They use the feedback from the environment - called the "reward" - to update their understanding of which actions are good and which are bad. In other words, they both try to get as much reward as possible by making good decisions.

Q-learning is what we call an "off-policy" method. This means that it learns the value of actions based on what would happen if it took the "best" action in a given situation, even if it doesn't actually take that action. SARSA, on the other hand, is an "on-policy" method. This means that it learns the value of actions based on the action it actually takes in a given situation.

However, both Q-learning and SARSA have some limitations. They are both "model-free" methods, which means they don't have a good way of estimating what might happen in situations they haven't seen before. In other words, they can't figure out what might happen in a new state they've never been in before. This makes them less flexible and less useful in some situations.

1. Types of Variables:

Numeric Variable : Discrete , Continues

Categorical Variable : Binomial , Nominal , Ordinal

Discrete numeric Variable : Variable which takes discrete values ie values from a set of whole numbers only.It can take countably finite values

Eg: Number of cars passing through a toll gate every minute.

Continuous numeric variable : Variable which can have infinite number of values withn a range.

Eg : Amount of rainfall in millimeters.

Nominal categorical Variable : Has no order and have 2 or more than 2 categories .

Ex : Blood Type , Political Party

Binary Categorical Variable :Has no order and strictly 2 categories .

Ex : presence or absence of a disease.

Ordinal Categorical variable : Ordered Nominal Variable.

Ex:Education-Primary,Secondary,High School , College and University.

1. Difference between Multiple Linear Regression and Multiplicative Linear Regression.

Multiple linear regression is a type of linear regression that involves two or more independent variables to predict the dependent variable.

For example, if we want to predict a person's salary based on their age, education, and years of experience, we can use multiple linear regression. Here, salary is the dependent variable, and age, education, and years of experience are the independent variables.

On the other hand, multiplicative regression is a different method used to find a relationship between variables. Instead of predicting the dependent variable directly from the independent variables, multiplicative regression involves finding a mathematical formula that describes how the variables relate to each other.

For example, Multiplicative regression is also useful when dealing with variables that have a proportional relationship. For example, if we want to predict the area of a rectangle based on its length and width, using multiplicative regression would be appropriate because the area is proportional to the length times the width.

However, the disadvantage of using multiplicative regression is that the results can be more difficult to interpret compared to multiple linear regression. The formula derived from multiplicative regression may not have a straightforward interpretation, and it may be challenging to understand the impact of each independent variable on the dependent variable.

1. Pros and Cons of Logistic Regression.

Pros:

It can handle both continuous and categorical independent variables.

Relativly Robust to outlier

Cons:

Logistic regression assumes that the relationship between the independent variables and the logit of the dependent variable is linear, which may not always be the case.

It requires a large sample size to produce stable and reliable results

It assumes that the independent variables are not highly correlated with each other, otherwise it can lead to multicollinearity problems.

It is not suitable for modeling dependent variables that have more than two categories

1. Explain bias and Variance with Real world examples .

Bias : Inability to capture true relationship is called bias. A model with high bias may underfit the data, meaning it is too simple and cannot capture the complexity of the data.

Variance: variance refers to the variability of the model's predictions for different training sets. A model with high variance may overfit the data, meaning it is too complex and has learned the noise in the training data, which can lead to poor performance on new data.

Predicting Housing Prices: Suppose you are trying to predict housing prices based on various features like location, square footage, and number of bedrooms. A model with high bias may assume that location is the only important feature and make overly simplistic predictions based on that assumption. A model with high variance may try to fit the data too closely and end up making inaccurate predictions for new, unseen data points.

Medical Diagnosis: Suppose you are trying to diagnose a medical condition based on various symptoms and test results. A model with high bias may oversimplify the diagnosis process and not take into account important factors that can affect the diagnosis. A model with high variance may be too sensitive to small variations in the data and produce inconsistent diagnoses for the same patient.

1. What are the splitting Criterions for ID3,C4.5,CART,CHAID and MARS ?

ID3 uses the entropy measure to select the best attribute to split the data

C4.5 uses information gain. C4.5 also has some additional features such as the ability to handle continuous attributes and missing data, and it can handle more than two classes in the target variable.

CART can handle both classification and regression problems, and it uses a different splitting criterion called the Gini index. It is also known for its ability to handle large datasets and high-dimensional data.

CHAID (Chi-squared Automatic Interaction Detector) is a decision tree algorithm that uses the chi-squared test to select the best attribute to split the data. CHAID is often used in market research and social sciences because it can handle categorical variables with many levels and it can identify interactions between variables.

MARS is used for regression problems .It does not use a splitting criterion like traditional decision tree algorithms. Instead, it constructs a set of basis functions and uses a forward selection process to iteratively add them to the model.

1. What do you mean by giving high weightage to wrong classified value in boosting ?

In boosting, the algorithm tries to improve its performance by iteratively learning from the mistakes made in the previous iterations. When a sample is misclassified in a particular iteration, the boosting algorithm assigns a higher weight to that sample in the next iteration so that it has a greater impact on the model's learning.

Giving high weightage to wrong classified values means that the boosting algorithm assigns a higher weight to the misclassified samples in subsequent iterations, which means that the model will focus more on these samples when trying to improve its performance. This approach helps the model to better learn from the mistakes it has made and adjust its decision boundaries accordingly.

However, assigning too much weight to misclassified samples can lead to overfitting, which means that the model becomes too specialized to the training data and may not perform well on new, unseen data. Therefore, it is important to carefully balance the weight assigned to misclassified samples to achieve the best possible performance without overfitting.

1. Difference between AdaBoost , Gradient Boost and XGBoost

Adaboost : Adaboost is a boosting algorithm that adjusts the weights of misclassified samples to improve model performance. It assigns higher weights to the misclassified samples and trains the model on the weighted data to reduce the error rate. Adaboost is computationally efficient , but it is sensitive to noisy data.

Gradient Boosting: Gradient Boosting is a boosting algorithm that uses the gradient descent optimization technique to minimize the loss function of the model. It builds the model by adding weak learners to the model sequentially, each time fitting the residual error of the previous model. Gradient Boosting can handle a variety of loss functions and is less sensitive to noisy data than Adaboost.

XGBoost :XGBoost is an optimized implementation of the Gradient Boosting algorithm that uses a variety of techniques to improve performance and scalability. It includes features such as regularization, parallel processing, and tree pruning to improve accuracy and reduce overfitting. XGBoost is highly customizable and is considered one of the best-performing gradient boosting algorithms.

In summary, Adaboost is a simple and computationally efficient boosting algorithm, Gradient Boosting is a more powerful and flexible algorithm that can handle a variety of loss functions, and XGBoost is an optimized implementation of Gradient Boosting that includes additional features to improve performance and scalability.

1. Give an example of Stacking.

We want to build a model to diagnose whether a patient has diabetes or not based on their medical history and test results. We can use stacking to combine the predictions of several base models trained on different subsets of features to improve the accuracy of our diagnosis model.

We start by training several base models on different subsets of features. For example, we can train a logistic regression model on the patient's age and BMI, a decision tree model on the patient's medical history (e.g., family history of diabetes, history of gestational diabetes), and a random forest model on all features.

We use the trained base models to make predictions on the validation data.

We use the predictions from the base models as input features for a meta-model. For example, we can use a KNN as our meta-model.

We train the KNN on the validation data, using the predictions of the base models as input features and the true labels (i.e., whether the patient has diabetes or not) as the target variable.

We use the trained KNN to make predictions on new, unseen data.

The key idea behind stacking is to combine the predictions of several models that are trained on different subsets of features to improve the overall accuracy of the diagnosis model. By using a meta-model to learn from the predictions of the base models, we can capture more complex patterns in the patient's medical history and achieve better performance than any of the individual models on their own.