1. What is the definition of a target function? In the sense of a real-life example, express the target function. How is a target function's fitness assessed?

Ans:- In the context of machine learning, the target function, also known as the objective function or the goal function, represents the mapping between the input variables (features) and the desired output variable (target) that a model aims to learn. It defines the relationship or pattern that the model needs to capture in order to make accurate predictions or classifications.

The target function can vary depending on the specific problem and the nature of the data. Here's a real-life example to illustrate the concept of a target function:

Example: House Price Prediction Suppose you have a dataset containing information about houses, such as their size, number of bedrooms, location, etc., along with their corresponding sale prices. The goal is to develop a machine learning model that can predict the sale price of a house given its features.

In this case, the target function would be the mapping between the input variables (house features) and the desired output variable (sale price). The target function would capture the underlying relationship between the features and the sale price, considering factors such as the size of the house, number of bedrooms, location, and any other relevant features.

The fitness of a target function is assessed based on how well it captures the underlying patterns or relationships in the data. The fitness evaluation involves comparing the predictions made by the target function to the actual target values in the dataset. This comparison is done using appropriate evaluation metrics, such as mean squared error (MSE) or mean absolute error (MAE), which quantify the difference between the predicted values and the actual values.

The goal in assessing the fitness of a target function is to minimize the prediction error, i.e., to make the predicted values as close as possible to the actual values. The fitness assessment helps in evaluating and comparing different models or algorithms, selecting the best-performing model, and optimizing the model's parameters or hyperparameters to improve its predictive accuracy.

It's important to note that in machine learning, the target function is not always explicitly defined or known beforehand. In many cases, the goal is to learn an approximation of the true target function from the available data using various learning algorithms and techniques.

1. What are predictive models, and how do they work? What are descriptive types, and how do you use them? Examples of both types of models should be provided. Distinguish between these two forms of models.

Ans:- Predictive Models: Predictive models in machine learning are designed to make predictions or estimates about future or unseen data based on patterns or relationships learned from historical or labeled data. These models learn from the input variables (features) and their corresponding target variables (labels) to generalize and make predictions on new, unseen data.

How Predictive Models Work:

Data Preparation: The historical or labeled data is collected and preprocessed, including handling missing values, encoding categorical variables, and scaling numeric features.

Training: The preprocessed data is divided into a training set and a validation set. The predictive model is trained on the training set using various learning algorithms such as linear regression, decision trees, support vector machines, or neural networks. During training, the model learns the underlying patterns and relationships in the data.

Evaluation: The trained model's performance is evaluated using the validation set, where it makes predictions on the validation data and compares them to the actual values. Evaluation metrics such as accuracy, precision, recall, or mean squared error are used to assess the model's predictive performance.

Prediction: Once the model is trained and evaluated, it can be used to make predictions on new, unseen data. The model takes the input features of the unseen data and applies the learned patterns and relationships to generate predictions or estimates.

Example of Predictive Model: Prediction of Customer Churn: A telecom company wants to predict which customers are likely to churn or cancel their services. They collect historical data about customers' usage, demographics, and customer churn status. Using this data, they build a predictive model that learns the patterns and factors associated with customer churn. The trained model can then predict the likelihood of churn for new customers based on their characteristics.

Descriptive Models: Descriptive models aim to describe or summarize the patterns and relationships in the data. They focus on understanding and interpreting the data rather than making predictions. Descriptive models are used to gain insights, identify patterns, and visualize the data to aid decision-making.

How Descriptive Models Work:

Data Exploration: The data is analyzed and explored using descriptive statistics, visualization techniques, and data mining methods. This exploration helps in understanding the characteristics and patterns in the data.

Pattern Discovery: Descriptive models identify and summarize the relationships, trends, or patterns in the data using techniques such as clustering, association rules, or anomaly detection. These models reveal hidden structures or relationships within the data.

Visualization: Descriptive models often involve visualizing the data through charts, graphs, or interactive dashboards. This visual

representation helps in presenting and interpreting the patterns and insights in an easily understandable way.

Example of Descriptive Model: Customer Segmentation: A retail company wants to understand its customer base and segment them based on their purchasing behavior. They analyze the historical transaction data and use clustering techniques to group customers with similar buying patterns. This descriptive model helps the company identify different customer segments, such as high-value customers, occasional buyers, or price-sensitive customers.

Distinguishing Between Predictive and Descriptive Models: The main difference between predictive and descriptive models lies in their purpose and focus. Predictive models aim to make predictions or estimates about future or unseen data, leveraging the learned patterns from historical data. Descriptive models, on the other hand, focus on describing and summarizing the patterns and relationships within the data to gain insights and understanding.

Predictive models use supervised learning algorithms and require labeled data for training and evaluation. They are used when the goal is to make predictions, classifications, or estimates on unseen data.

Descriptive models use unsupervised learning algorithms and focus on summarizing, visualizing, and interpreting the patterns in the data. They are used when the goal is to understand the data, identify hidden structures or relationships, and gain insights for decision-making.

1. Describe the method of assessing a classification model's efficiency in detail. Describe the various measurement parameters.

Ans:-Assessing the efficiency of a classification model involves evaluating its performance in predicting class labels for a given dataset. There are various measurement parameters commonly used to assess the performance of classification models. Let's explore some of them:

Accuracy: Accuracy measures the overall correctness of the model's predictions by calculating the ratio of correctly predicted instances to the total number of instances in the dataset. It provides a general measure of the model's performance but may not be suitable when the classes are imbalanced.

Confusion Matrix: A confusion matrix provides a detailed breakdown of the model's predictions by showing the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). It helps in understanding the types of errors the model is making.

Precision: Precision measures the proportion of correctly predicted positive instances (true positives) out of all instances predicted as positive (true positives + false positives). It indicates the model's ability to avoid false positives

1. i. In the sense of machine learning models, what is underfitting? What is the most common reason for underfitting?

underfitting refers to a situation where a model is too simple or lacks the capacity to capture the underlying patterns and relationships in the data. Underfitting occurs when a model performs poorly both on the training data and new, unseen data.

The most common reason for underfitting is a lack of model complexity or insufficient model capacity. This means that the model is not able to capture the complexity and nuances of the data. Some common reasons for underfitting include:

Model simplicity: When the model used is too simple and has limited flexibility, it may struggle to capture the complex relationships present in the data. For example, using a linear regression model to fit a non-linear relationship between variables may result in underfitting.

Insufficient training: If the model is not trained with enough data or the training data does not represent the full range of patterns and variations in the actual data, the model may not learn the underlying patterns effectively.

Inappropriate feature selection: If the model is trained with a limited set of features that do not adequately represent the true relationship between the input and output variables, it can lead to underfitting. Important features may be omitted, resulting in a poor model fit.

Over-regularization: Regularization techniques such as L1 or L2 regularization are used to prevent overfitting, but excessive regularization can lead to underfitting. If the regularization parameter is set too high, the model's ability to capture the complex patterns in the data is diminished.

To address underfitting, one can consider the following approaches:

Increase model complexity: Use a more complex model that can better capture the relationships in the data. For example, using a higher-degree polynomial regression instead of a simple linear regression.

Feature engineering: Explore additional features or transform existing features to better represent the underlying patterns in the data. This can help the model capture more nuanced relationships.

Gather more data: Increase the size of the training dataset to provide the model with more diverse and representative examples, enabling it to learn more effectively.

Adjust regularization: If regularization is being used, try reducing the regularization strength or adjusting the hyperparameters to find the right balance between preventing overfitting and allowing the model to capture important patterns.

By addressing the reasons for underfitting and optimizing the model's complexity, data representation, and regularization, one can improve the model's ability to capture the underlying patterns in the data and reduce underfitting.

ii. What does it mean to overfit? When is it going to happen?

Overfitting refers to a situation where a model performs exceptionally well on the training data but fails to generalize to new, unseen data. It occurs when the model learns the training data too closely, capturing both the underlying patterns and the noise or random fluctuations in the data.

Overfitting typically happens in the following scenarios:

Insufficient training data: When the training dataset is small, the model may learn the noise or specific patterns present in that particular dataset instead of the general underlying patterns. This leads to overfitting because the model is unable to generalize well to new data.

Model complexity: Complex models with a high number of parameters or a large number of features have a greater capacity to fit the training data. If the complexity of the model exceeds the complexity of the underlying patterns in the data, the model can start capturing noise or outliers, leading to overfitting.

Overemphasis on training data: If the model is trained for too many iterations or epochs, it may start memorizing the training data instead of learning the general patterns. This results in overfitting, as the model becomes too specialized for the training data.

Incorrect feature selection: If the model is trained with irrelevant or redundant features, it may learn the noise or variations in those features, leading to overfitting. Proper feature selection and preprocessing are important to avoid overfitting.

The consequences of overfitting are:

Poor generalization: An overfitted model is highly specific to the training data and fails to generalize well to new data. Its performance on unseen data is usually poor.

Increased error on test data: Overfitting leads to higher errors on test data or real-world scenarios because the model is making predictions based on noise or specific patterns that are not present in new data.

To mitigate overfitting, the following approaches can be used:

Increase training data: Collecting more training data helps the model to capture a wider range of patterns and reduce the influence of noise.

Simplify the model: Use a simpler model with fewer parameters or features to avoid capturing noise or overfitting the training data. This can be achieved by reducing the model complexity or applying techniques like regularization.

Cross-validation: Perform cross-validation to evaluate the model's performance on multiple folds of the data. This helps to assess the model's generalization ability and detect overfitting.

Regularization: Apply regularization techniques like L1 or L2 regularization to penalize overly complex models. This helps in preventing the model from overfitting the training data.

The goal is to strike a balance between model complexity and the ability to generalize well to new data, thereby avoiding both underfitting and overfitting.

iii. In the sense of model fitting, explain the bias-variance trade-off.

Bias-variance trade-off is a fundamental concept in model fitting that involves finding the right balance between bias and variance in order to achieve optimal predictive performance.

Bias refers to the error introduced by approximating a real-world problem with a simplified model. A model with high bias makes strong assumptions about the data, leading to systematic errors. This can result in underfitting, where the model fails to capture the underlying patterns in the data.

Variance, on the other hand, refers to the variability in model predictions that occurs due to sensitivity to fluctuations in the training data. A model with high variance is highly flexible and can capture complex relationships, but it may also fit the noise or random fluctuations in the training data. This can lead to overfitting, where the model performs well on the training data but fails to generalize to new data.

The bias-variance trade-off arises from the inherent trade-off between model complexity and model performance:

A simple model, such as linear regression, has low variance but high bias. It makes strong assumptions about the data and may not capture complex relationships accurately. It is more prone to underfitting.

A complex model, such as a deep neural network, has high variance but low bias. It can capture complex relationships and fit the training data well. However, it is more prone to overfitting as it may also capture noise or random fluctuations in the training data.

To strike the right balance between bias and variance, the goal is to find an optimal level of model complexity that minimizes the overall error. This can be achieved through techniques such as regularization, cross-validation, and model selection.

Regularization methods, such as L1 or L2 regularization, help control model complexity by adding a penalty term to the objective function. This reduces the variance and helps prevent overfitting.

Cross-validation allows us to assess the model's performance on unseen data by splitting the data into training and validation sets. It helps in estimating the trade-off between bias and variance and aids in selecting the appropriate model complexity.

Model selection involves choosing the best model from a set of candidate models with different levels of complexity. The goal is to find the model that minimizes the overall error, balancing bias and variance.

The bias-variance trade-off highlights the need to find the right level of model complexity that minimizes both bias and variance, leading to optimal predictive performance. It is a key consideration in model fitting to ensure that the model generalizes well to new, unseen data.

1. Is it possible to boost the efficiency of a learning model? If so, please clarify how.

Ans:- Yes, it is possible to boost the efficiency of a learning model through various techniques and approaches. Here are some common methods to improve model efficiency:

Feature Engineering: Feature engineering involves creating new features or transforming existing ones to better represent the underlying patterns in the data. By selecting and creating informative features, the model can focus on the most relevant information and improve its efficiency.

Model Selection: Choosing the right model architecture or algorithm for the problem at hand is crucial for efficiency. Different models have different strengths and weaknesses, and selecting the most appropriate one can significantly improve performance. It involves considering factors such as model complexity, interpretability, and scalability.

Hyperparameter Tuning: Models often have hyperparameters that control their behavior, such as learning rate, regularization strength, or the number of hidden layers. Tuning these hyperparameters using techniques like grid search, random search, or Bayesian optimization can help find the optimal settings for improved model efficiency.

Regularization Techniques: Regularization methods like L1 or L2 regularization can prevent overfitting by adding penalty terms to the model's objective function. Regularization helps to control model complexity and improve generalization, leading to better efficiency.

Ensemble Methods: Ensemble methods combine multiple models to make predictions. Techniques like bagging (e.g., Random Forests) and boosting (e.g., AdaBoost, Gradient Boosting) can improve efficiency by reducing bias and variance and enhancing overall predictive performance.

Model Optimization: Optimizing the model's computational performance can enhance efficiency. This can involve techniques such as using optimized libraries or frameworks, parallel computing, or leveraging hardware accelerators like GPUs.

Data Augmentation: Data augmentation techniques involve creating additional training data by applying transformations or perturbations to the existing data. This helps increase the diversity of the training set and can improve the model's ability to generalize.

Early Stopping: Early stopping is a technique where the model training is stopped before convergence if the validation performance starts to degrade. This prevents overfitting and improves efficiency by avoiding unnecessary training iterations.

Transfer Learning: Transfer learning leverages pre-trained models on large-scale datasets and fine-tunes them on a smaller target dataset. By utilizing the learned representations from pre-training, it can boost efficiency by leveraging prior knowledge.

Model Deployment: Efficient deployment of models, such as using optimized inference techniques, model compression, or deployment on edge devices, can improve efficiency in real-time applications.

It's important to note that the specific approach to boosting model efficiency may vary depending on the problem, the dataset, and the available resources. It often involves a combination of techniques tailored to the specific context to achieve the desired improvement in efficiency.

1. How would you rate an unsupervised learning model's success? What are the most common success indicators for an unsupervised learning model?

Ans:- Evaluating the success of an unsupervised learning model can be challenging as there is no ground truth or target variable to directly compare predictions against. However, there are several common indicators and evaluation methods that can be used to assess the performance of unsupervised learning models:

Clustering Evaluation: If the unsupervised learning task involves clustering, various evaluation metrics can be used to assess the quality of the clusters. Some commonly used metrics include the Silhouette coefficient, Calinski-Harabasz index, Davies-Bouldin index, and within-cluster sum of squares (WCSS). These metrics measure the compactness and separation of the clusters, and a higher value indicates better clustering performance.

Visualization and Interpretation: Visualization techniques, such as scatter plots, heatmaps, or dimensionality reduction methods like t-SNE or PCA, can provide insights into the structure and patterns within the data. Examining the visual representation of the data can help understand if the model has captured meaningful relationships or groupings.

Domain Knowledge Alignment: In unsupervised learning, success can also be evaluated by assessing if the discovered patterns, groups, or associations align with existing domain knowledge or expert insights. This involves examining the results of the model in the context of the problem domain and determining if the identified patterns are meaningful and useful.

Internal Consistency: Internal consistency measures assess how well the data points within a cluster are similar to each other and dissimilar to points in other clusters. Metrics like the average within-cluster distance or the ratio of within-cluster distance to between-cluster distance can provide insights into the compactness and separation of clusters.

Outlier Detection: Unsupervised learning models can also be evaluated based on their ability to detect outliers or anomalies in the data. Metrics like the outlier detection rate or the area under the receiver operating characteristic (ROC) curve can be used to assess the model's performance in identifying unusual or anomalous data points.

Reconstruction Error: In some unsupervised learning techniques, such as autoencoders or dimensionality reduction methods like PCA, the reconstruction error can be used as an evaluation metric. The lower the reconstruction error, the better the model's ability to capture the essential features or structure of the data.

It's important to note that the choice of evaluation metrics depends on the specific unsupervised learning task and the objectives of the analysis. Sometimes, a combination of multiple evaluation methods may be used to get a comprehensive understanding of the model's success. Additionally, the evaluation of unsupervised learning models is often subjective and requires domain expertise to interpret the results in a meaningful way.

1. Is it possible to use a classification model for numerical data or a regression model for categorical data with a classification model? Explain your answer.

Ans:- No, it is not appropriate to use a classification model for numerical data or a regression model for categorical data without appropriate modifications. Here's why:

Classification Model for Numerical Data: A classification model is designed to predict categorical labels or classes based on input features. It works by learning decision boundaries or rules to assign data points to specific classes. Numerical data, on the other hand, represents continuous values. Attempting to use a classification model directly on numerical data would not make sense as the model cannot effectively learn and predict continuous values. It would result in a mismatch between the nature of the data and the assumptions made by the classification algorithm.

Regression Model for Categorical Data: A regression model is designed to predict continuous numerical values based on input features. It models the relationship between the input variables and the target variable to make predictions. Categorical data represents discrete categories or labels. Using a regression model directly on categorical data would not be appropriate because the model would try to fit a continuous function to discrete categories, leading to incorrect and meaningless predictions. The regression model assumes a continuous target variable, so it would not capture the inherent nature of categorical data.

However, it is possible to handle numerical data in a classification model by discretizing it into categorical bins or using techniques like ordinal encoding or one-hot encoding to convert it into a suitable format for classification. Similarly, for categorical data, techniques like label encoding or one-hot encoding can be used to convert it into a suitable format for regression. These encoding techniques help represent categorical variables as numerical features that can be used by regression models.

In summary, it is essential to choose the appropriate model based on the nature of the data and the objective of the problem. Using the wrong model type can lead to inaccurate results and misinterpretations.

1. Describe the predictive modeling method for numerical values. What distinguishes it from categorical predictive modeling?

Ans:- The predictive modeling method for numerical values, also known as regression modeling, is used to predict continuous numerical outcomes based on input features. It aims to establish a mathematical relationship between the independent variables (input features) and the dependent variable (target variable) to make predictions on new data. Here are some key characteristics of predictive modeling for numerical values:

Target Variable: In numerical predictive modeling, the target variable is a continuous numerical variable that we want to predict. It can be a quantity, measurement, or numerical outcome of interest, such as predicting house prices, stock prices, or sales revenue.

Model Type: Regression models are commonly used for numerical predictive modeling. These models estimate the relationship between the input features and the target variable by fitting a mathematical function or equation. Popular regression algorithms include linear regression, polynomial regression, decision trees, random forests, support vector regression (SVR), and neural networks.

Evaluation Metrics: The performance of a numerical predictive model is typically assessed using metrics that measure the accuracy or goodness of fit between the predicted values and the actual values of the target variable. Common evaluation metrics for regression models include mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), R-squared (coefficient of determination), and adjusted R-squared.

Interpretability: In numerical predictive modeling, the focus is often on understanding the relationship between the input features and the target variable. Regression models provide coefficients or weights that quantify the impact of each feature on the target variable. This allows for interpretation and understanding of the direction and magnitude of the relationship between variables.

On the other hand, categorical predictive modeling, also known as classification modeling, is used to predict categorical labels or classes based on input features. It involves determining which class or category an observation belongs to. Here are some distinctions between numerical and categorical predictive modeling:

Target Variable: In categorical predictive modeling, the target variable is a categorical variable with discrete classes or labels. Examples include predicting whether an email is spam or not, classifying images into different object categories, or predicting customer churn (yes or no).

Model Type: Classification models are used for categorical predictive modeling. These models learn decision boundaries or rules to assign

data points to specific classes based on the input features. Common classification algorithms include logistic regression, decision trees, random forests, support vector machines (SVM), and neural networks.

Evaluation Metrics: Evaluation metrics for classification models are different from those used for numerical predictive modeling. Metrics such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (ROC AUC) are used to assess the performance of classification models.

Interpretability: While interpretation is possible in classification modeling, the focus is often more on prediction and classification accuracy rather than understanding the relationship between variables.

In summary, the main distinction between numerical and categorical predictive modeling lies in the nature of the target variable, the type of model used, the evaluation metrics employed, and the level of interpretability and understanding of the relationship between variables.

1. The following data were collected when using a classification model to predict the malignancy of a group of patients' tumors:

i. Accurate estimates – 15 cancerous, 75 benign

ii. Wrong predictions – 3 cancerous, 7 benign

Determine the model's error rate, Kappa value, sensitivity, precision, and F-measure.

Ans:--- To calculate the error rate, Kappa value, sensitivity, precision, and F-measure, we need to know the definitions of these metrics in the context of binary classification:

True Positive (TP): The model correctly predicts a positive (cancerous) instance. True Negative (TN): The model correctly predicts a negative (benign) instance. False Positive (FP): The model incorrectly predicts a positive instance (false alarm). False Negative (FN): The model incorrectly predicts a negative instance (missed detection). Using these definitions, we can calculate the metrics as follows:

Error Rate: The error rate measures the overall accuracy of the model's predictions. Error Rate = (FP + FN) / (TP + TN + FP + FN)

In this case: Error Rate = (3 + 7) / (15 + 75 + 3 + 7) = 0.1

Kappa Value: The Kappa value measures the agreement between the model's predictions and the actual labels, taking into account the agreement that could occur by chance. Kappa Value = (Accuracy - Expected Accuracy) / (1 - Expected Accuracy)

In this case: Accuracy = (TP + TN) / (TP + TN + FP + FN) = (15 + 75) / (15 + 75 + 3 + 7) = 0.9 Expected Accuracy = ((TP + FP) *(TP + FN)* + *(TN + FP)* (TN + FN)) / (TP + TN + FP + FN)^2 = ((15 + 3) *(15 + 7)* + *(75 + 3)* (75 + 7)) / (15 + 75 + 3 + 7)^2 = 0.742

Kappa Value = (0.9 - 0.742) / (1 - 0.742) = 0.424

Sensitivity (Recall): Sensitivity measures the proportion of actual positive instances (cancerous tumors) that are correctly identified by the model. Sensitivity = TP / (TP + FN)

In this case: Sensitivity = 15 / (15 + 3) = 0.833

Precision: Precision measures the proportion of predicted positive instances that are actually positive (cancerous tumors). Precision = TP / (TP + FP)

In this case: Precision = 15 / (15 + 7) = 0.682

F-measure: The F-measure combines precision and sensitivity into a single metric that balances both measures. F-measure = 2 *(Precision* Sensitivity) / (Precision + Sensitivity)

In this case: F-measure = 2 *(0.682* 0.833) / (0.682 + 0.833) = 0.750

1. Make quick notes on:

2. The process of holding out

The process of holding out refers to reserving a portion of the available data for testing purposes. Typically, the dataset is split into a training set and a holdout set or validation set. The training set is used to train the model, while the holdout set is used to evaluate its performance on unseen data.

1. Cross-validation by tenfold

Cross-validation by tenfold, also known as 10-fold cross-validation, is a popular technique for model evaluation. The dataset is divided into ten equal parts or folds. The model is trained and evaluated ten times, each time using a different combination of nine folds for training and one fold for testing. This technique provides a more robust estimation of the model's performance by averaging the results across all ten iterations. It helps to reduce the impact of randomness in the training and testing data splits.

1. Adjusting the parameters

Adjusting the parameters refers to finding the optimal values for the hyperparameters of a machine learning algorithm. Hyperparameters are settings that are not learned from the data but are set by the user before training the model. Different values of hyperparameters can

significantly impact the performance of the model. The process of adjusting the parameters involves trying different combinations of hyperparameter values and evaluating the model's performance. Techniques like grid search or random search can be used to systematically explore the hyperparameter space and find the best combination that yields the highest performance. Adjusting the parameters is essential to optimize the model's performance and ensure it generalizes well to new data.

1. Define the following terms:

2. Purity vs. Silhouette width

Purity is a measure used in clustering analysis to evaluate the quality of a clustering result. It measures the extent to which samples within each cluster belong to the same class or category. Higher purity indicates that the clusters contain samples from predominantly the same class, while lower purity suggests mixed or overlapping classes within clusters. Silhouette width is another measure used in clustering analysis to assess the quality of clustering. It quantifies the compactness of clusters and the separation between different clusters. Higher silhouette width values indicate well-separated and distinct clusters, while lower values indicate overlapping or poorly separated clusters.

1. Boosting vs. Bagging

Boosting and Bagging are ensemble learning methods used to improve the performance of machine learning models by combining multiple base models. Boosting is a sequential ensemble method where each subsequent model in the sequence is trained to correct the mistakes made by the previous models. It focuses on difficult samples and assigns higher weights to them, allowing subsequent models to learn from the errors. Bagging, on the other hand, is a parallel ensemble method where each base model is trained independently on different subsets of the training data. It aims to reduce the variance of the model by averaging the predictions of multiple models. Bagging assigns equal weights to all training samples and focuses on reducing the impact of outliers and noisy data.

1. The eager learner vs. the lazy learner

The eager learner, also known as the eager classifier or eager model, is a type of machine learning algorithm that eagerly builds a classification model during the training phase. It pre-processes the training data and constructs a model representation that can be used for prediction directly. Examples of eager learners include decision trees, neural networks, and rule-based classifiers. The eager learner requires a significant amount of computational resources and time to build the model upfront, but it can provide faster predictions during the testing phase. In contrast, the lazy learner, also known as the lazy classifier or lazy model, defers the processing of the training data until a query or prediction is made. It stores the training data and uses it directly during the testing phase to make predictions. Examples of lazy learners include k-nearest neighbors (KNN) and case-based reasoning (CBR) algorithms. The lazy learner has lower computational requirements during the training phase but may have slower prediction times compared to eager learners.