20CB/CS/DS/IT602

Hall Ticket Number:

III/IV B.Tech (Regular) DEGREE EXAMINATION

July/August, 2023 — **Common to CB,CSE,DS &IT Branches**

Sixth Semester — **Machine Learning**

Time: Three Hours — **Maximum:** 70 Marks

Answer question 1 compulsory. **(14X1 = 14Marks)**

Answer one question from each unit. **(4X14=56 Marks)**

			CO	BL	M
1	a)	What is machine learning and how does it differ from traditional programming?	CO1	L1	1M
	b)	List out any five different context where machine learning is applicable.	CO1	L1	1M
	c)	What is mini-batch gradient descent and how does it differ from batch gradient descent?	CO1	L2	1M
	d)	Define Overfitting and Underfitting.	CO2	L1	1M
	e)	Define entropy and write down its formula.	CO2	L2	1M
	f)	What is logistic regression, and how does it handle binary classification problems?	CO2	L1	1M
	g)	How is accuracy measured using cross-validation?	CO3	L2	1M
	h)	Discuss the precision/recall trade-off and how it impacts classifier performance	CO3	L1	1M
	i)	How does the hierarchical clustering algorithm work, and what are its advantages?	CO4	L2	1M
	j)	Define support vector.	CO4	L2	1M
	k)	Write about Lasso Regression	CO1	L1	1M
	l)	Differentiate Supervised Learning and Unsupervised Learning	CO4	L1	1M
	m)	What is Confusion Matrix? How to create it?	CO3	L1	1M
	n)	Explain about Bagging and Voting.	CO3	L1	1M
		Unit-I			
2	a)	What are the main types of machine learning systems? Describe each type and provide examples of their applications?	CO1	L1	7M
	b)	What are the typical steps involved in developing a machine learning application? Provide an overview of the process. What are some evaluation metrics used to assess the performance of machine learning models,	CO1	L2	7M
		(OR)			
3	a)	How do you choose the right machine learning algorithm for a given problem or task? Explain the process of understanding the problem domain, data characteristics, and selecting the appropriate algorithm based on factors like the nature of the problem, available data, and desired outcomes?	CO1	L2	7M
	b)	What are the key Python libraries used in machine learning? Provide an overview of few libraries and their roles in tasks such as data manipulation, visualization, and model training?	CO1	L1	7M
		Unit-II			
4	a)	Explain in detail about CART algorithm with an example.	CO2	L2	7M
	b)	Explain in detail about SVM with an example.	CO2	L2	7M
		(OR)			
5	a)	How to measure the accuracy of a classifier? Explain the concept of cross-validation and how it helps in estimating the model's performance on unseen data?	CO2	L1	7M
	b)	How can you train and use a Naïve Bayes classifier for classification tasks? Explain the steps involved in estimating the class priors and conditional probabilities and making predictions based on the maximum posterior probability?	CO2	L3	7M

P.T.O

Unit-III

6	a)	What is the precision/recall trade-off, and how does it relate to the classifier's decision threshold? Discuss how adjusting the threshold affects the precision and recall values and how to choose an appropriate threshold based on the specific application requirements?	CO3	L3	7M
	b)	How do ensemble methods address the bias-variance trade-off? Discuss how combining multiple classifiers reduces variance and can help to create more robust and accurate models?	CO3	L2	7M
		(OR)			
7		Explain in detail about Ada boost and Gradient boost with examples.	CO3	L1	14M
		Unit-IV			
8	a)	How does the growth function in computational learning theory relate to the sample complexity? Explain the growth function as a measure of the number of distinct dichotomies that can be generated by a hypothesis space and its impact on the learnability of a concept?	CO4	L3	7M
	b)	What is the Gaussian mixture model (GMM) in unsupervised learning? Explain the concept of representing the underlying data distribution as a mixture of Gaussian components and using the Expectation-Maximization (EM) algorithm to estimate the parameters?	CO4	L3	7M
		(OR)			
9	a)	How does hierarchical clustering work? Explain the top-down and bottom-up approaches of building a hierarchical cluster tree by iteratively merging or splitting clusters based on similarity measures?	CO4	L1	7M
	b)	Explain in detail about K-means Clustering Algorithm with an example.	CO4	L2	7M

1. a. What is machine learning and how does it differ from traditional programming?

Ans: Machine learning is a subfield of artificial intelligence (AI) that focuses on the development of algorithms and models that enable computers to learn and improve from experience without being explicitly programmed. The goal of machine learning is to allow computers to make data-driven decisions or predictions based on patterns and insights found in data. In traditional programming, the developer writes explicit instructions and rules for the computer to follow in order to perform a specific task or solve a particular problem. These rules are created based on the developer's understanding of the problem domain and how to tackle it.

b. List out any five different context where machine learning is applicable.

Ans: i. Natural Language Processing (NLP)

ii. Image and Video Analysis

iii. Recommender Systems

iv. Healthcare

v. Autonomous Systems

vi. Fraud Detection

c. What is mini-batch gradient descent and how does it differ from batch gradient descent?

Ans: In batch gradient descent, it is said that one iteration of gradient descent update takes the processing of whole entire dataset, which makes an epoch. On the other hand, in mini batch algorithm an update is made after every mini batch and once every mini batch is done, one epoch is completed.

d. Define Overfitting and Underfitting.

Ans: A machine learning algorithm is said to have underfitting when it cannot capture the underlying trend of the data, i.e., it only performs well on training data but performs poorly on testing data. (It’s just like trying to fit undersized pants!)

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

e. Define entropy and write down its formula.

Ans: It is a measure of randomness (or) unpredictability in the dataset.

$$\text{Entropy (S)} = - P/(P+N) \log_2 (P/P+N) – N/P+N \log_2 (N/P+N)$$

f. What is logistic regression, and how does it handle binary classification problems?

Ans: Logistic regression is one of the most popular algorithms for binary classification. Logistic regression is usually used for Binary classification problems. Binary Classification refers to predicting the output variable that is discrete in two classes. It uses the Sigmoid function to solve the classification problems.

g. How is accuracy measured using cross-validation?

Ans: The model is trained on n-p data points and later tested on p data points. The same process is repeated for all possible combinations of p from the original sample. Finally, the results of each iteration are averaged to attain the cross-validation accuracy.

h. Discuss the precision/recall trade-off and how it impacts classifier performance.

Ans: Precision and recall are performance metrics used for pattern recognition and classification in machine learning. Precision is defined as the ratio of correctly classified positive samples (True Positive) to a total number of classified positive samples (either correctly or incorrectly). The recall is calculated as the ratio between the numbers of Positive samples correctly classified as Positive to the total number of Positive samples. The recall measures the model's ability to detect positive samples. The higher the recall, the more positive samples detected.

i. How does the hierarchical clustering algorithm work, and what are its advantages?

Ans: Hierarchical clustering starts by treating each observation as a separate cluster. Then, it repeatedly executes the following two steps:

i. Identify the two clusters that are closest together.

ii. Merge the two most similar clusters. This iterative process continues until all the clusters are merged.

The advantage of Hierarchical Clustering is we don't have to pre-specify the clusters.

j. Define support vector.

Ans: The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

k. Write about Lasso Regression.

Ans: Lasso regression is a regularization technique. It is used over regression methods for a more accurate prediction. Lasso regression stands for Least Absolute Shrinkage and Selection Operator.

It adds penalty term to the cost function. This term is the absolute sum of the coefficients. As the value of coefficients increases from 0 this term penalizes, cause model, to decrease the value of coefficients to reduce loss.

l. Differentiate Supervised Learning and Unsupervised Learning.

Ans: Supervised learning is a machine learning method in which models are trained using labelled data. In supervised learning, models need to find the mapping function to map the input variable (X) with the output variable (Y). Supervised learning needs supervision to train the model, which is similar to as a student learns things in the presence of a teacher. Supervised learning can be used for two types of problems: Classification and Regression.

Unsupervised learning is another machine learning method in which patterns inferred from the unlabelled input data. The goal of unsupervised learning is to find the structure and patterns from the input data. Unsupervised learning does not need any supervision. Instead, it finds patterns from the data by its own. Unsupervised learning can be used for two types of problems: Clustering and Association.

m. What is Confusion Matrix? How to create it?

Ans: A confusion matrix is a chart or table that summarizes the performance of a classification model or algorithm for machine learning processes. Confusion matrices help with predictive analysis and can be effective tools for evaluating what functions a machine learning system performs correctly and incorrectly.

The matrix displays the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) produced by the model on the test data. For binary classification, the matrix will be of a 2X2 table, For multi-class classification, the matrix shape will be equal to the number of classes i.e for n classes it will be nXn. A 2X2 Confusion matrix is shown below for the image recognition having a Dog image or Not Dog image.

n. Explain about Bagging and Voting.

Ans: In Voting, we consider multiple models, and each model will be trained on the given dataset. Each model will predict the output for the new data or testing data. Use all the predictions to det the final prediction. If it is classification problem, then we should take major voting. If it is Regression problem, then should take average value of all the predictions.

In Bagging, the input data is divided into small groups rather than considering multiple models. Each group is given for training to the same model. From each group, the model will predict the value and we can use all the predictions to get the final prediction. If it is classification problem, then we should take major voting. If it is Regression problem, then should take average value of all the predictions.

UNIT-I

2. a. What are the main types of machine learning systems? Describe each type and provide examples of their applications?

Ans:

Machine learning systems can be broadly categorized into three main types based on the type of learning they employ: supervised learning, unsupervised learning, and reinforcement learning. Let's describe each type and provide examples of their applications:

i. Supervised Learning:

Supervised learning is a type of machine learning where the algorithm is trained on a labeled dataset, meaning each data point is paired with its corresponding target label. The goal is for the algorithm to learn the mapping between input features and their corresponding outputs so that it can make predictions on new, unseen data.

Example applications:

a. Image Classification: Identifying objects or patterns in images, such as classifying images of animals, vehicles, or identifying diseases from medical images.

b. Speech Recognition: Converting spoken language into text, like voice assistants or transcription services.

c. Spam Detection: Identifying emails or messages as spam or non-spam based on their content.

d. Sentiment Analysis: Determining the sentiment of a piece of text, such as identifying whether a review is positive or negative.

ii. Unsupervised Learning:

In unsupervised learning, the algorithm is trained on an unlabeled dataset, meaning there are no target labels provided. The system's goal is to find patterns, structures, or relationships within the data without any explicit guidance.

Example applications:

a. Clustering: Grouping similar data points together based on their features, such as customer segmentation or grouping similar news articles.

b. Anomaly Detection: Identifying unusual or rare events, such as fraudulent transactions or abnormal behavior in systems.

c. Dimensionality Reduction: Reducing the number of features or variables while preserving important information, often used for data visualization or feature extraction.

iii. Reinforcement Learning:

Reinforcement learning involves training an agent to make decisions in an environment to achieve a specific goal. The agent interacts with the environment, receives feedback in the form of rewards or penalties based on its actions, and learns to maximize the cumulative reward over time.

Example applications:

a. Game Playing: Training agents to play video games, like AlphaGo, which learned to play Go at a world-champion level.

b. Robotics: Teaching robots to perform tasks and navigate their environments effectively.

c. Autonomous Vehicles: Training self-driving cars to make driving decisions based on the surrounding environment and traffic conditions.

d. Recommender Systems: Providing personalized recommendations to users, such as in movie recommendations or product recommendations on e-commerce platforms.

b. What are the typical steps involved in developing a machine learning application? Provide an overview of the process. What are some evaluation metrics used to assess the performance of machine learning models.

Ans:

Developing a machine learning application involves several distinct steps to go from problem formulation to model deployment. Here's an overview of the typical process:

i. Problem Definition and Data Collection:

- Define the problem you want to solve and the objectives you want to achieve with machine learning.
- Gather relevant data for training, validation, and testing.

ii. Data Preprocessing:

- Clean and preprocess the data by handling missing values, outliers, and noise.
- Normalize, scale, or standardize features to ensure consistent data representation.
- Perform feature engineering to create new meaningful features from the existing ones.

iii. Data Splitting:

- Split the dataset into training, validation, and testing sets.

- The training set is used to train the model, the validation set helps tune hyperparameters, and the testing set evaluates the final model's performance.

iv. Model Selection:

- Choose the appropriate machine learning algorithm or model architecture based on the problem type (classification, regression, etc.) and characteristics of the data.
- Experiment with different algorithms and techniques to find the best fit.

v. Model Training:

- Train the selected model on the training data using an appropriate optimization algorithm.
- Adjust model parameters to minimize the loss function (error) and improve predictive performance.

vi. Hyperparameter Tuning:

- Tune hyperparameters (parameters that are not learned during training) to optimize the model's performance.
- This is often done using techniques like grid search, random search, or more advanced methods like Bayesian optimization.

vii. Model Evaluation:

- Evaluate the model's performance on the validation set to ensure it generalizes well to new, unseen data.
- Choose appropriate evaluation metrics based on the problem type (classification, regression, etc.).

viii. Model Optimization:

- Refine the model based on evaluation results, adjusting hyperparameters or making design changes.
- Avoid overfitting by incorporating regularization techniques or using larger datasets.

ix. Model Testing and Validation:

- Test the optimized model on the testing set to assess its real-world performance.
- Validate that the model meets the predefined objectives and delivers the desired results.

x. Deployment and Monitoring:

- Deploy the trained model into a production environment where it can make predictions on new, unseen data.
- Implement monitoring mechanisms to track the model's performance in real-world scenarios and make updates if necessary.

Common Evaluation Metrics:

i. Classification:

a. Accuracy: Proportion of correctly classified instances.

b. Precision: Proportion of true positive predictions among all positive predictions.

c. Recall (Sensitivity): Proportion of true positive predictions among all actual positives.

d. F1-Score: Harmonic mean of precision and recall, useful when balancing precision and recall.

e. ROC-AUC: Area under the Receiver Operating Characteristic curve, indicating the model's ability to distinguish between classes.

ii. Regression:

a. Mean Absolute Error (MAE): Average absolute difference between predicted and actual values.

b. Mean Squared Error (MSE): Average squared difference between predicted and actual values.

c. Root Mean Squared Error (RMSE): Square root of MSE, providing interpretable units.

d. R-squared (Coefficient of Determination): Proportion of variance in the dependent variable explained by the model.

iii. Clustering:

a. Silhouette Score: Measures the quality of cluster separation.

b. Inertia: Sum of squared distances between data points and their cluster's center.

c. Adjusted Rand Index (ARI): Measures the similarity between true and predicted cluster assignments.

3. a. How do you choose the right machine learning algorithm for a given problem or task? Explain the process of understanding the problem domain, data characteristics, and selecting the appropriate algorithm based on factors like the nature of the problem, available data, and desired outcomes?

Ans:

Choosing the right machine learning algorithm for a given problem involves a systematic process that takes into account various factors related to the problem domain, data characteristics, and desired outcomes. Here's a step-by-step guide on how to approach this process:

1. Understand the Problem Domain:

- Gain a clear understanding of the problem you're trying to solve and the goals you want to achieve with machine learning.

- Determine whether the problem is a classification, regression, clustering, recommendation, or another type.

2. Analyze Data Characteristics:

- Examine the nature of the data you have:

- Is the data structured (tabular) or unstructured (text, images, audio)?

- What are the dimensions and size of the dataset?

- Are there missing values, outliers, or noise in the data?

3. Determine Data Availability:

- Assess the amount of labeled data available for supervised learning tasks.

- Consider whether the dataset is balanced or imbalanced in terms of class distribution.

- Evaluate the quality and reliability of the data.

4. Consider Model Complexity:

- Determine whether the problem requires a simple or complex model.

- A complex problem might benefit from deep learning, while a simple problem might be solved using linear models.

5. Choose Based on Problem Characteristics:

- Classification: If the problem involves categorizing data into classes, consider algorithms like Logistic Regression, Decision Trees, Random Forests, Support Vector Machines (SVM), or deep learning models like Convolutional Neural Networks (CNNs) for image classification.

- Regression: For predicting continuous values, options include Linear Regression, Decision Trees, Random Forests, Gradient Boosting, or even neural networks.

- Clustering: When the goal is to group similar data points, algorithms like K-Means, Hierarchical Clustering, or DBSCAN could be suitable.

- Recommendation: Collaborative Filtering, Matrix Factorization, or deep learning-based approaches can be used to build recommendation systems.

6. Consider Algorithm's Suitability:

- Evaluate whether the algorithm is suitable for handling the specific problem characteristics.

- Consider the strengths and weaknesses of each algorithm and how they align with your problem.

7. Check Scalability and Efficiency:

- Depending on the size of your dataset, consider the algorithm's computational requirements.

- Some algorithms might be more efficient for large datasets, while others might struggle with scalability.

8. Experiment and Compare:

- It's a good practice to experiment with multiple algorithms and evaluate their performance using appropriate metrics on a validation set.

- Compare how well different algorithms perform and choose the one that provides the best results.

9. Iterative Process:

- The selection process might involve iterations as you fine-tune hyperparameters, preprocess data differently, or explore additional algorithms.

10. Seek Expert Advice:

- If you're uncertain, consider seeking advice from domain experts or machine learning practitioners who have experience with similar problems.

11. Keep Learning:

- Machine learning is a dynamic field, and new algorithms are constantly being developed. Stay up-to-date with the latest advancements and research.

In summary, choosing the right machine learning algorithm involves a combination of understanding the problem, analyzing data characteristics, evaluating algorithm suitability, experimenting, and making informed decisions based on your goals and resources. It's often an iterative process that requires careful consideration and exploration to achieve the best results.

b. What are the key Python libraries used in machine learning? Provide an overview of few libraries and their roles in tasks such as data manipulation, visualization, and model training?

Ans:

Python has a rich ecosystem of libraries that are widely used in machine learning tasks. Some of the key libraries include:

1. NumPy:

- Role: NumPy provides support for efficient array operations and mathematical functions, making it the fundamental package for numerical computations in Python.

- Data Manipulation: NumPy arrays are used for handling and manipulating numerical data, enabling vectorized operations and efficient computation.

2. Pandas:

- Role: Pandas offers data structures and functions for efficiently handling and analyzing structured data (tabular or time-series) in Python.

- Data Manipulation: Pandas DataFrames and Series provide powerful tools for data cleaning, transformation, aggregation, and handling missing data.

3. Matplotlib:

- Role: Matplotlib is a plotting library that enables the creation of static, interactive, and animated visualizations in Python.

- Data Visualization: Matplotlib provides functions to create various types of plots, including line plots, scatter plots, histograms, and more.

4. Seaborn:

- Role: Seaborn is built on top of Matplotlib and offers a higher-level interface for creating aesthetically pleasing statistical visualizations.

- Data Visualization: Seaborn simplifies the creation of complex visualizations, such as heatmaps, pair plots, and distribution plots.

5. Scikit-learn:

- Role: Scikit-learn is a machine learning library that provides simple and efficient tools for data mining and data analysis.

- Model Training: Scikit-learn includes a wide range of algorithms for classification, regression, clustering, dimensionality reduction, and more. It also offers tools for model selection, evaluation, and hyperparameter tuning.

6. TensorFlow and PyTorch:

- Role: These libraries are used for building and training deep learning models.

- Model Training: TensorFlow and PyTorch provide powerful frameworks for defining, training, and deploying neural network models. They offer GPU acceleration and support for automatic differentiation.

7. Keras:

- Role: Keras is an API that runs on top of TensorFlow, Theano, or Microsoft Cognitive Toolkit (CNTK). It provides a high-level interface for building and training neural networks.

- Model Training: Keras simplifies the process of creating complex neural network architectures and offers an easy-to-use interface for training and evaluation.

8. XGBoost and LightGBM:

- Role: These libraries are used for gradient boosting, a popular ensemble learning technique.

- Model Training: XGBoost and LightGBM are optimized implementations of gradient boosting algorithms that provide high-performance and accurate models for regression and classification tasks.

9. NLTK and spaCy:

- Role: These libraries are focused on natural language processing (NLP) tasks.

- Text Processing: NLTK and spaCy offer tools for tokenization, part-of-speech tagging, named entity recognition, sentiment analysis, and more.

These are just a few of the many libraries available in the Python ecosystem for machine learning. Each library plays a crucial role in various stages of the machine learning workflow, from data manipulation and visualization to model training and deployment. Depending on your specific project requirements, you can combine these libraries to create comprehensive and effective machine learning pipelines.

UNIT-II

4. a. Explain in detail about CART algorithm with an example.

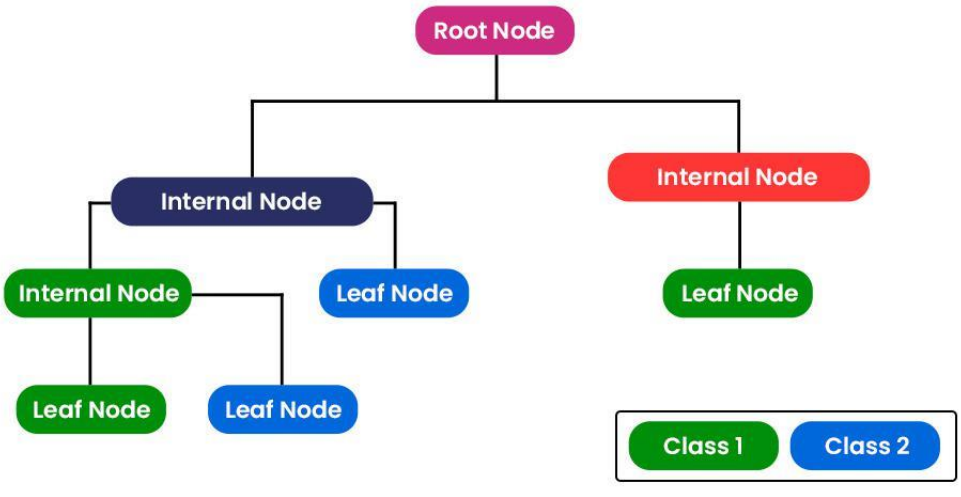
Ans:

CART is a predictive algorithm used in Machine learning and it explains how the target variable's values can be predicted based on other matters. It is a decision tree where each fork is split into a predictor variable and each node has a prediction for the target variable at the end.

In the decision tree, nodes are split into sub-nodes on the basis of a threshold value of an attribute. The root node is taken as the training set and is split into two by considering the best attribute and threshold value. Further, the subsets are also split using the same logic. This continues till the last pure sub-set is found in the tree or the maximum number of leaves possible in that growing tree.

The CART algorithm works via the following process:

- The best split point of each input is obtained.
- Based on the best split points of each input in Step 1, the new "best" split point is identified.
- Split the chosen input according to the "best" split point.
- Continue splitting until a stopping rule is satisfied or no further desirable splitting is available.



CART algorithm uses Gini Impurity to split the dataset into a decision tree. It does that by searching for the best homogeneity for the sub nodes, with the help of the Gini index criterion.

CART Algorithm Key Concepts:

- Gini Impurity: Used for classification tasks, Gini impurity measures the probability of a randomly selected element being misclassified. The goal is to minimize Gini impurity during the tree construction.
- Mean Squared Error (MSE): Used for regression tasks, MSE measures the average squared difference between predicted and actual values. The goal is to minimize MSE during the tree construction.
- Feature Selection: At each step, the algorithm selects the best feature and threshold for splitting based on the chosen criterion (Gini impurity or MSE).
- Pruning: Pruning is used to prevent overfitting by removing branches that do not contribute significantly to improved predictive performance.

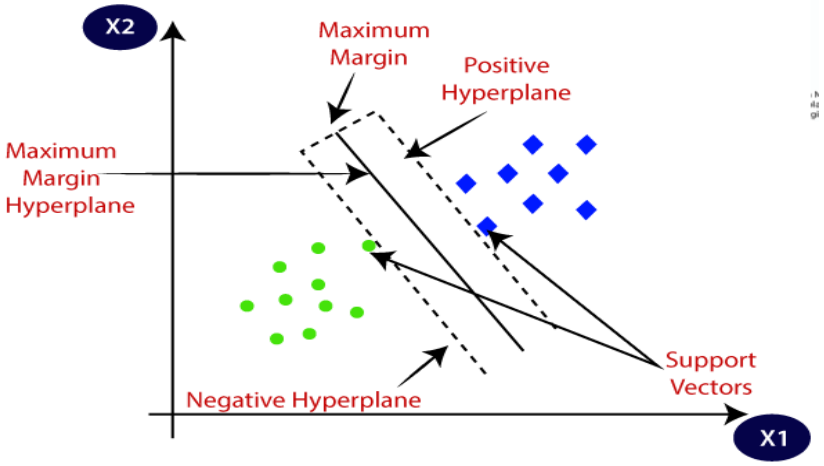
b. Explain in detail about SVM with an example.

Ans:

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat.

Types of SVM:

SVM can be of two types:

a. Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

b. Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

5. a. How to measure the accuracy of a classifier? Explain the concept of cross-validation and how it helps in estimating the model's performance on unseen data?

Ans:

In machine learning, we couldn’t fit the model on the training data and can’t say that the model will work accurately for the real data. For this, we must assure that our model got the correct patterns from the data, and it is not getting up too much noise. For this purpose, we use the cross-validation technique.

Cross validation is a technique used in machine learning to evaluate the performance of a model on unseen data. It involves dividing the available data into multiple folds or subsets, using one of these folds as a validation set, and training the model on the remaining folds. This process is repeated multiple times, each time using a different fold as the validation set. Finally, the results from each validation step are averaged to produce a more robust estimate of the model's performance.

The main purpose of cross validation is to prevent overfitting, which occurs when a model is trained too well on the training data and performs poorly on new, unseen data. By evaluating the model on multiple validation sets, cross validation provides a more realistic estimate of the model's generalization performance, i.e., its ability to perform well on new, unseen data.

Common Types of Cross-Validation:

a. K-Fold Cross-Validation

   K-fold cross-validation approach divides the input dataset into K groups of samples of equal sizes. These samples are called folds. For each learning set, the prediction function uses k-1 folds, and the rest of the folds are used for the test set. This approach is a very popular CV approach because it is easy to understand, and the output is less biased than other methods.

   The steps for k-fold cross-validation are:

   - Split the input dataset into K groups
   - For each group:
   - Take one group as the reserve or test data set.
   - Use remaining groups as the training dataset.
   - Fit the model on the training set and evaluate the performance of the model using the test set.

b. Stratified k-fold cross-validation:

   This technique is similar to k-fold cross-validation with some little changes. This approach works on stratification concept, it is a process of rearranging the data to ensure that each fold or group is a good representative of the complete dataset. To deal with the bias and variance, it is one of the best approaches.

   It can be understood with an example of housing prices, such that the price of some houses can be much high than other houses. To tackle such situations, a stratified k-fold cross-validation technique is useful.

c. Leave one out cross-validation:

This method is similar to the leave-p-out cross-validation, but instead of p, we need to take 1 dataset out of training. It means, in this approach, for each learning set, only one datapoint is reserved, and the remaining dataset is used to train the model. This process repeats for each datapoint. Hence for n samples, we get n different training set and n test set. It has the following features:

In this approach, the bias is minimum as all the data points are used.

The process is executed for n times; hence execution time is high.

This approach leads to high variation in testing the effectiveness of the model as we iteratively check against one data point.

d. Leave-P-out cross-validation:

In this approach, the p datasets are left out of the training data. It means, if there are total n datapoints in the original input dataset, then n-p data points will be used as the training dataset and the p data points as the validation set. This complete process is repeated for all the samples, and the average error is calculated to know the effectiveness of the model. There is a disadvantage of this technique; that is, it can be computationally difficult for the large p.

b. How can you train and use a Naïve Bayes classifier for classification tasks? Explain the steps involved in estimating the class priors and conditional probabilities and making predictions based on the maximum posterior probability?

Ans:

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

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

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

  Where,

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

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

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

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

- Naive Bayes Classifier Types:
  a. Multinomial Naive Bayes: Used for text classification, it's suitable when features represent counts or frequencies, such as word counts in a document.
  b. Gaussian Naive Bayes: Assumes that features follow a Gaussian (normal) distribution. It's suitable for continuous numerical features.
  c. Bernoulli Naive Bayes: Suitable for binary features (0 or 1), often used in text classification where features represent presence or absence of words.

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

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

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

UNIT-III

6. a. What is the precision/recall trade-off, and how does it relate to the classifier's decision threshold? Discuss how adjusting the threshold affects the precision and recall values and how to choose an appropriate threshold based on the specific application requirements?

Ans:

The precision/recall trade-off is a fundamental concept in classification that involves a balance between the precision and recall of a classifier's predictions. Precision and recall are two important evaluation metrics used to assess a classifier's performance, especially in imbalanced datasets where one class is much more frequent than the other.

- Precision: Precision measures the proportion of positive predictions made by the classifier that are actually correct. It focuses on minimizing false positive predictions. The formula for precision is:

  Precision = TP / (TP + FP)

  Where TP is the number of true positives and FP is the number of false positives.

- Recall (Sensitivity): Recall measures the proportion of actual positive instances that were correctly predicted by the classifier. It focuses on minimizing false negatives. The formula for recall is:

  Recall = TP / (TP + FN)

  Where TP is the number of true positives and FN is the number of false negatives.

The classifier's decision threshold plays a crucial role in determining how predictions are made. By default, many classifiers use a threshold of 0.5, meaning that if the predicted probability of the positive class is greater than or equal to 0.5, the instance is classified as positive; otherwise, it's classified as negative. However, adjusting the threshold can lead to changes in precision and recall values.

- Lowering the Threshold:

  - When the threshold is lowered, the classifier becomes more sensitive to positive instances, resulting in increased recall and potentially decreased precision.

  - More instances are predicted as positive, which reduces false negatives but may increase false positives.

- Raising the Threshold:

  - When the threshold is raised, the classifier becomes more conservative in predicting positive instances, resulting in increased precision and potentially decreased recall.

  - Fewer instances are predicted as positive, which reduces false positives but may increase false negatives.

Choosing an Appropriate Threshold:

The choice of threshold depends on the specific application requirements and the trade-off between precision and recall. Here are some considerations:

- High Recall, Lower Precision: In applications where missing positive instances is highly undesirable (e.g., medical diagnoses for serious conditions), a lower threshold might be preferred to maximize recall, even if it leads to some false positives.

- High Precision, Lower Recall: In applications where minimizing false positives is crucial (e.g., fraud detection), a higher threshold might be chosen to increase precision, even if it results in missing some positive instances.

- Balanced Trade-off: For applications where both precision and recall are important, you might aim for a balanced trade-off. You can analyze the precision-recall curve to identify a threshold that provides a good balance between the two metrics.

- Domain Knowledge: Domain expertise can guide the threshold choice. Understanding the consequences of false positives and false negatives in your specific context can help determine an appropriate threshold.

In summary, adjusting the threshold of a classifier affects its precision and recall values. The choice of threshold should be based on the specific goals and requirements of the application, striking a balance between identifying positive instances correctly and minimizing false predictions.

b. How do ensemble methods address the bias-variance trade-off? Discuss how combining multiple classifiers reduces variance and can help to create more robust and accurate models?

Ans:

Ensemble methods are machine learning techniques that combine the predictions of multiple base models (classifiers or regressors) to improve overall performance. These methods are particularly effective at addressing the bias-variance trade-off by mitigating the limitations of individual models.

- Bias refers to the error introduced by approximating a real-world problem, which may be complex, by a simplified model. High bias models tend to oversimplify and may underfit the data.
- Variance refers to the error introduced due to the model's sensitivity to small fluctuations in the training data. High variance models can fit the training data very well but perform poorly on unseen data (overfitting).

Ensemble methods tackle the bias-variance trade-off by leveraging the strengths of different base models to create a more robust and accurate model:

a. Reducing Variance:

- Ensemble methods combine multiple base models, often referred to as "weak learners," to create a "strong learner."
- By averaging or aggregating the predictions of individual models, the noise and errors in each model's predictions tend to cancel out. This leads to reduced variance and improved generalization to unseen data.

b. Increasing Predictive Power:

- Ensemble methods aim to capture different sources of information and patterns in the data that individual models might miss.
- This diversity in predictions helps to capture a broader range of features and relationships in the data, leading to improved model performance.

c. Handling Complex Relationships:

- Ensembles can capture complex relationships by combining different models with varying strengths and weaknesses.
- While one model might perform well on linear relationships, another might excel in capturing non-linear patterns. Ensembles allow these models to complement each other.

d. Error Correction:

- Ensemble methods can identify and correct misclassifications or errors made by individual models. By considering multiple perspectives, the ensemble is less likely to make the same mistakes.

Common Ensemble Methods:

a. Bagging (Bootstrap Aggregating): Creates multiple base models trained on bootstrapped subsets of the data. The final prediction is obtained by averaging or voting over the predictions of these models. Example: Random Forest.

b. Boosting: Sequentially trains multiple weak learners, where each new model focuses on correcting the errors made by the previous ones. The final prediction is a weighted combination of all models' predictions. Example: AdaBoost, Gradient Boosting Machines (GBM), XGBoost.

c. Voting (Majority Voting): Combines the predictions of multiple base models by majority voting (classification) or averaging (regression).

d. Stacking: Involves training multiple base models and using their predictions as features for a meta-model that makes the final prediction.

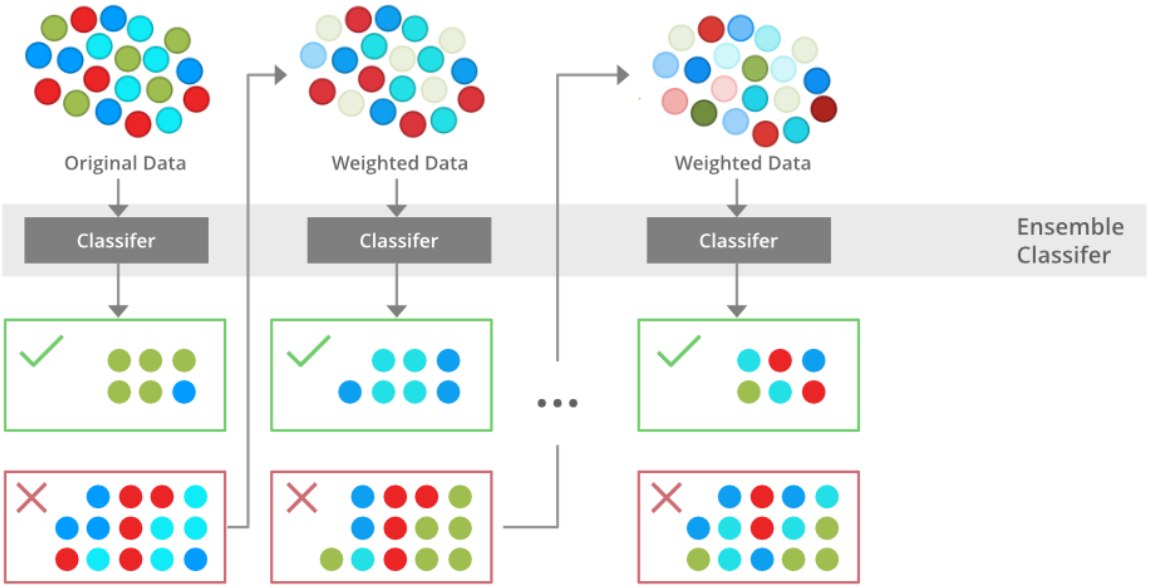
7. Explain in detail about Ada boost and Gradient boost with examples.

Ans:

Boosting is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.

Advantages of Boosting

- Improved Accuracy – Boosting can improve the accuracy of the model by combining several weak models' accuracies and averaging them for regression or voting over them for classification to increase the accuracy of the final model.
- Robustness to Overfitting – Boosting can reduce the risk of overfitting by reweighting the inputs that are classified wrongly.
- Better handling of imbalanced data – Boosting can handle the imbalance data by focusing more on the data points that are misclassified
- Better Interpretability – Boosting can increase the interpretability of the model by breaking the model decision process into multiple processes.



a. AdaBoost Algorithm

AdaBoost, also called Adaptive Boosting, is a technique in Machine Learning used as an Ensemble Method. The most common estimator used with AdaBoost is decision trees with one level which means Decision trees with only 1 split. These trees are also called Decision Stumps.

Sure, let's delve into the details of AdaBoost (Adaptive Boosting) and Gradient Boosting, two popular ensemble methods used in machine learning.

AdaBoost is an ensemble learning algorithm that aims to improve the performance of weak learners by sequentially combining their predictions. Weak learners are models that perform slightly better than random guessing. AdaBoost assigns higher weights to misclassified instances, allowing subsequent weak learners to focus on the misclassified cases.

Here's how AdaBoost works:

1. Initialization:

   - Assign equal weights to all training instances.

   - Choose a weak learner (e.g., decision stump, a simple decision tree with one level).

2. Training:

   - Train the weak learner on the training data with the assigned instance weights.

   - Calculate the weighted error, which is the sum of weights of misclassified instances.

3. Calculate Model Weight:

- Calculate the weight of the weak learner's prediction based on its accuracy in classifying instances.

- More accurate models get higher weights.

4. Update Instance Weights:

- Increase the weights of misclassified instances.

- Decrease the weights of correctly classified instances.

5. Normalize Weights:

- Normalize the instance weights so they sum up to 1.

6. Iteration:

- Repeat steps 2-5 for a predefined number of iterations or until a stopping criterion is met.

7. Final Prediction:

- Combine the predictions of weak learners using their model weights to make the final prediction.

Example of AdaBoost:

Consider a binary classification problem where you want to classify emails as spam or not spam (ham). AdaBoost might start with a decision stump (a weak learner) that uses a single feature to make a classification decision. After the first iteration, it increases the weight of misclassified instances, allowing the next weak learner to focus on correcting those mistakes. This process continues, and eventually, the ensemble combines the predictions of multiple weak learners to make a final decision.

b. Gradient Boosting:

Gradient Boosting is another ensemble method that sequentially combines weak learners to create a strong learner. However, unlike AdaBoost, Gradient Boosting focuses on minimizing the loss function using gradient descent, gradually improving the model's performance by fitting subsequent learners to the residuals (errors) of the previous ones.

Here's how Gradient Boosting works:

1. Initialization:

- Initialize the model with a constant value, typically the mean of the target variable.

2. Calculate Residuals:

- Calculate the residuals (errors) by subtracting the model's predictions from the actual target values.

3. Train Weak Learner:

- Train a weak learner on the residuals. The weak learner learns to predict the "correction" needed to improve the model's performance.

4. Update Model:

- Update the model's predictions by adding a fraction of the weak learner's predictions (learning rate) to the current predictions.

5. Repeat:

- Repeat steps 2-4 for a predefined number of iterations.

6. Final Prediction:

- The final model is a weighted combination of the predictions made by all the weak learners.

Example of Gradient Boosting:

Suppose you're working on a regression task to predict house prices. The initial model predicts the mean house price. In the first iteration, a weak learner is trained on the residuals, which represents the difference between the predicted prices and the actual prices. The weak learner learns to predict the correction needed to improve the model's predictions. The model's predictions are then updated by adding a fraction of the weak learner's predictions, adjusting the predictions towards the correct values.

In summary, both AdaBoost and Gradient Boosting are powerful ensemble methods that combine weak learners to create strong learners. While AdaBoost focuses on adjusting instance weights to prioritize misclassified instances, Gradient Boosting iteratively improves the model by fitting to residuals using gradient descent. Both methods effectively address the bias-variance trade-off and improve predictive accuracy.

UNIT-IV

8. a. How does the growth function in computational learning theory relate to the sample complexity? Explain the growth function as a measure of the number of distinct dichotomies that can be generated by a hypothesis space and its impact on the learnability of a concept?

Ans:

In computational learning theory, the growth function is a crucial concept that relates to the sample complexity of a hypothesis space and plays a significant role in understanding the learnability of a concept. The growth function measures the number of distinct dichotomies (distinct ways of labeling points as positive or negative) that can be generated by a hypothesis space of a given size.

a. Growth Function and Sample Complexity:

The growth function is defined for a particular hypothesis space and represents the maximum number of distinct dichotomies that can be achieved by any subset of the training data. The sample complexity refers to the minimum number of training examples needed to ensure that the learner can approximate the true concept (target function) within a desired level of accuracy.

The growth function's relationship to sample complexity is fundamental: A smaller growth function generally implies a smaller sample complexity, making the learning task easier. If the growth function is smaller, fewer training examples are required to ensure that the learner can fit the target function effectively.

b. Impact on Learnability:

The growth function directly impacts the learnability of a concept within a given hypothesis space. It provides insights into how many different possible dichotomies can be realized by the hypothesis space, indicating how expressive or complex the space is.

- If the growth function is small, the hypothesis space is constrained, and there are fewer possible dichotomies. This suggests that the concept is relatively easier to learn, as there are fewer ways it can be misclassified.

- If the growth function is large, the hypothesis space is more flexible and capable of accommodating a larger variety of dichotomies. This might imply that the concept is more complex and might require a larger number of training examples to effectively approximate.

In essence, a small growth function implies a more manageable hypothesis space with better generalization properties, leading to better learnability. A large growth function might indicate a more challenging learning task due to the space's higher complexity and potentially larger sample complexity.

Understanding the relationship between the growth function, sample complexity, and the complexity of the hypothesis space helps researchers and practitioners make informed decisions about the feasibility of learning a particular concept using a given set of training data and hypothesis space. It provides a theoretical foundation for assessing the trade-offs between model complexity, sample size, and learning performance.

b. What is the Gaussian mixture model (GMM) in unsupervised learning? Explain the concept of representing the underlying data distribution as a mixture of Gaussian components and using the Expectation-Maximization (EM) algorithm to estimate the parameters?

Ans:

A Gaussian Mixture Model (GMM) is a probabilistic model used in unsupervised learning to represent the underlying data distribution as a mixture of multiple Gaussian (normal) distributions. GMM assumes that the data is generated from a combination of several Gaussian distributions, each representing a distinct cluster or component within the data. GMM is particularly useful for modeling complex data distributions that might have overlapping clusters.

Concept of Gaussian Mixture Model:

Imagine you have a dataset that seems to consist of multiple groups or clusters, but you're not sure how many clusters there are or which data point belongs to which cluster. GMM provides a way to model such data by representing it as a weighted sum of Gaussian distributions:

$$P(x) = \Sigma\ w\_i * N(x \mid \mu\_i, \Sigma\_i)$$

Where:

- `P(x)` is the probability density of data point `x`.

- `w_i` is the weight (or mixing coefficient) of the i-th Gaussian component.

- `N(x | μ_i, Σ_i)` is the Gaussian distribution with mean `μ_i` and covariance `Σ_i`.

Expectation-Maximization (EM) Algorithm for Parameter Estimation:

Estimating the parameters of a GMM involves finding the means, covariances, and weights of the Gaussian components that best fit the data. The Expectation-Maximization (EM) algorithm is commonly used for this purpose. It's an iterative algorithm that alternates between two main steps: Expectation (E-step) and Maximization (M-step).

1. Expectation (E-step):

- In the E-step, for each data point, compute the probability that it belongs to each Gaussian component (responsibility).

- Calculate the posterior probabilities using Bayes' theorem and the current estimates of the parameters.

2. Maximization (M-step):

- In the M-step, update the parameters (mean, covariance, and weight) of each Gaussian component using the responsibilities computed in the E-step.

- Update the weights by averaging the responsibilities across all data points.

- Update the means and covariances using the weighted data points.

3. Iteration:

- Repeat the E-step and M-step until the algorithm converges, i.e., the parameters stabilize or the change in parameters becomes very small.

The EM algorithm aims to maximize the likelihood of the observed data under the GMM. Each iteration of the algorithm improves the parameter estimates to better capture the data distribution's characteristics.

Advantages and Considerations:

Gaussian Mixture Models have several advantages:

- They can capture complex data distributions with overlapping clusters.

- GMMs provide a soft assignment of data points to clusters through probabilities.

- They can be used for density estimation and data generation.

However, GMMs also have limitations:

- Determining the number of components (clusters) can be challenging.

- They can struggle with high-dimensional data due to the "curse of dimensionality."

- GMMs assume that each component is Gaussian and has equal covariance. This might not hold for all data.

GMMs are widely used in applications like image segmentation, anomaly detection, and speech recognition. While EM is the most common approach for parameter estimation, GMMs can also be extended to handle more complex distributions or relaxed assumptions.

9. a. How does hierarchical clustering work? Explain the top-down and bottom-up approaches of building a hierarchical cluster tree by iteratively merging or splitting clusters based on similarity measures?

Ans:

Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabelled datasets into a cluster and also known as hierarchical cluster analysis or HCA. In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the dendrogram. Sometimes the results of K-means clustering, and hierarchical clustering may look similar, but they both differ depending on how they work.

The hierarchical clustering technique has two approaches:

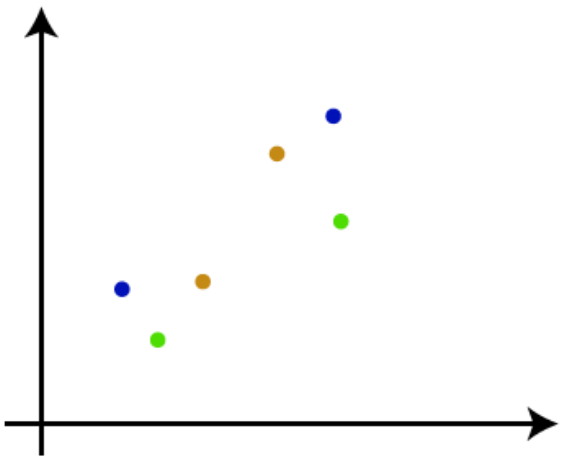
- Agglomerative: Agglomerative is a bottom-up approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.
- Divisive: Divisive algorithm is the reverse of the agglomerative algorithm as it is a top-down approach.

Agglomerative Hierarchical clustering:

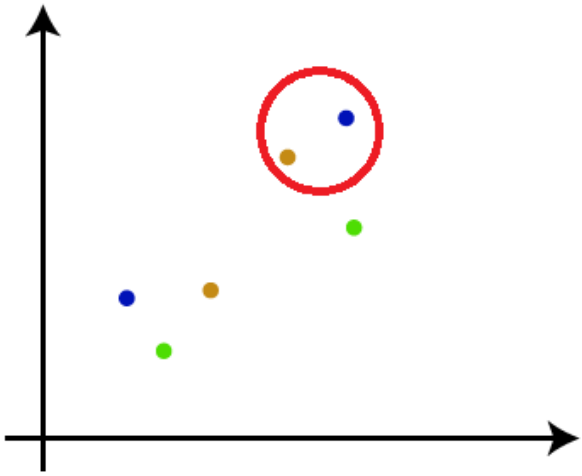
The agglomerative hierarchical clustering algorithm is a popular example of HCA. To group the datasets into clusters, it follows the bottom-up approach. It means, this algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters together. It does this until all the clusters are merged into a single cluster that contains all the datasets. This hierarchy of clusters is represented in the form of the dendrogram.

The working of the AHC algorithm can be explained using the below steps:

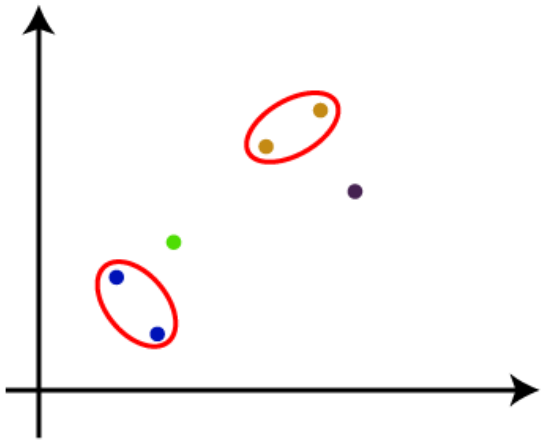
Step-1: Create each data point as a single cluster. Let's say there are N data points, so the number of clusters will also be N.



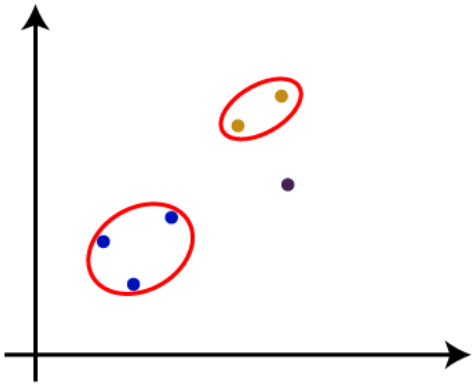
Step-2: Take two closest data points or clusters and merge them to form one cluster. So, there will now be N-1 clusters.

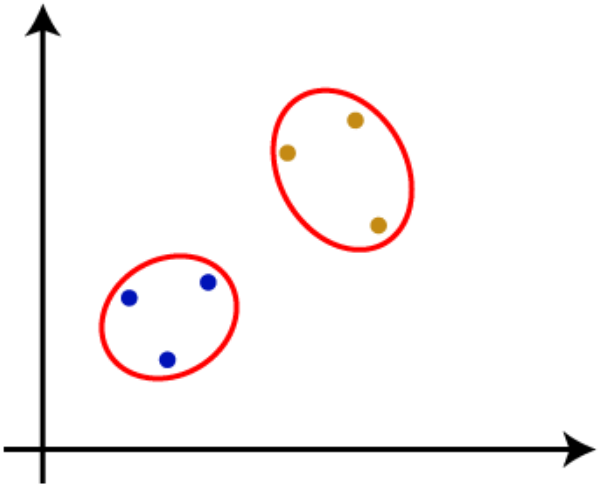


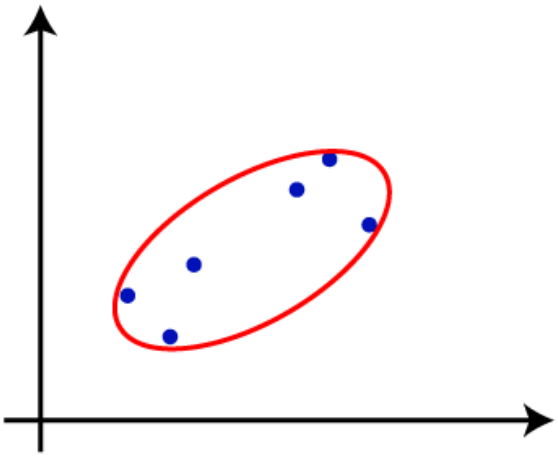
Step-3: Again, take the two closest clusters and merge them together to form one cluster. There will be N-2 clusters.



Step-4: Repeat Step 3 until only one cluster left. So, we will get the following clusters. Consider the below images:





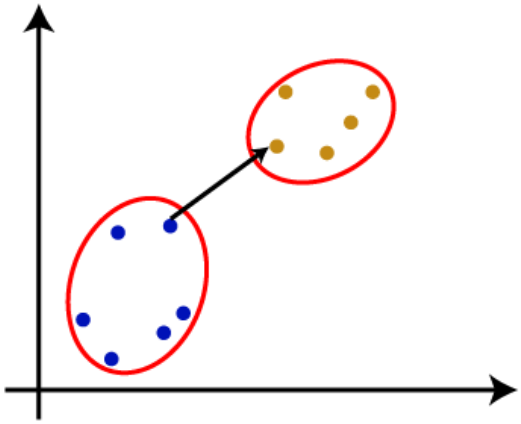


Step-5: Once all the clusters are combined into one big cluster, develop the dendrogram to divide the clusters as per the problem.

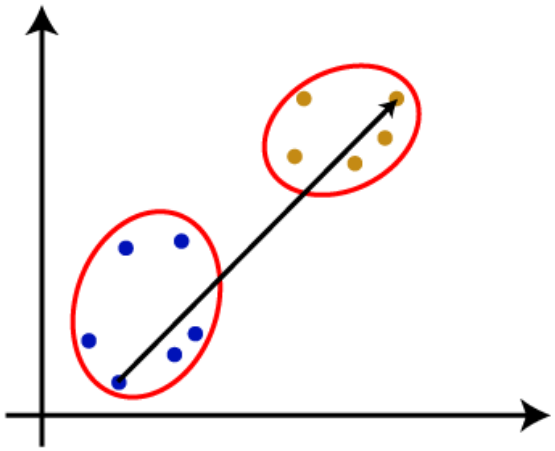
Measure for the distance between two clusters:

As we have seen, the closest distance between the two clusters is crucial for the hierarchical clustering. There are various ways to calculate the distance between two clusters, and these ways decide the rule for clustering. These measures are called Linkage methods. Some of the popular linkage methods are given below:

a. Single Linkage: It is the Shortest Distance between the closest points of the clusters. Consider the below image:

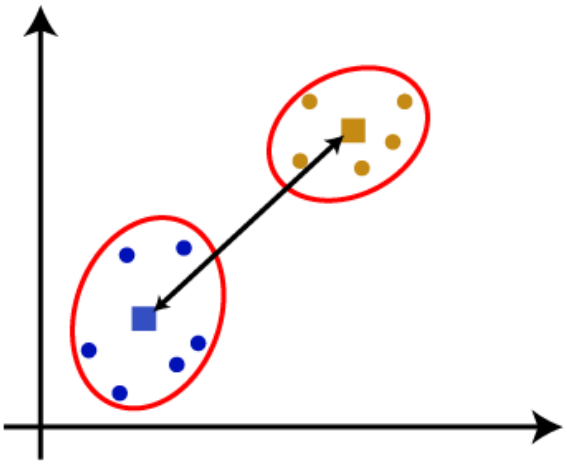


b. Complete Linkage: It is the farthest distance between the two points of two different clusters. It is one of the popular linkage methods as it forms tighter clusters than single-linkage.



c. Average Linkage: It is the linkage method in which the distance between each pair of datasets is added up and then divided by the total number of datasets to calculate the average distance between two clusters. It is also one of the most popular linkage methods.

d. Centroid Linkage: It is the linkage method in which the distance between the centroid of the clusters is calculated. Consider the below image:



b. Explain in detail about K-means Clustering Algorithm with an example.

Ans:

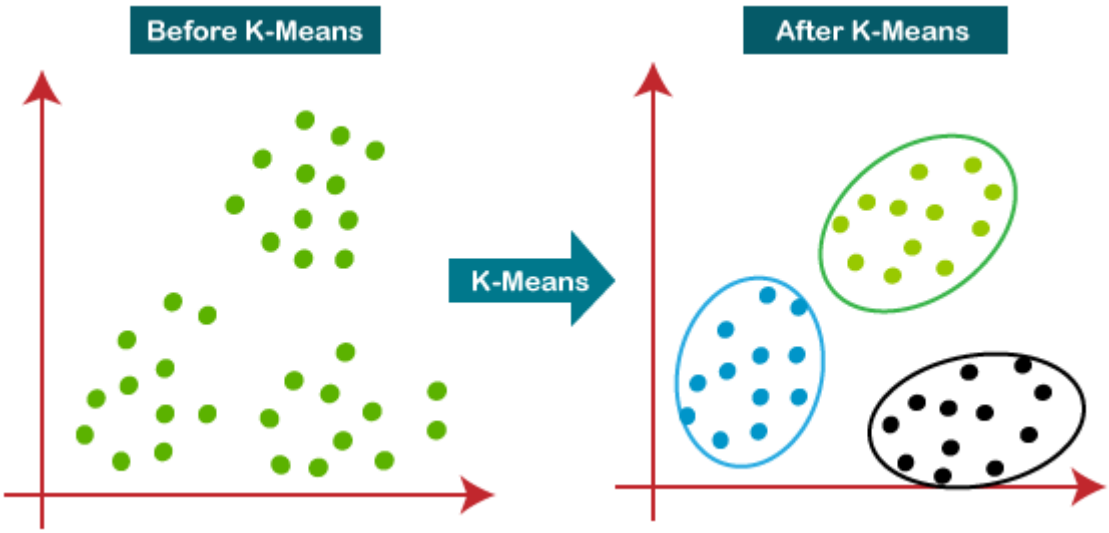
K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on. It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training. It is a centroid-based algorithm, where each cluster is associated with a centroid.

The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters. The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters. The below diagram explains the working of the K-means Clustering Algorithm:



The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

Consider any example.

Prepared By:	**Department**	**Signature**
J. Madhan Kumar	Computer Science & Engineering	
Taught By:	**Department**	**Signature**
1. J. Madhan Kumar	Computer Science & Engineering	
2. P.V.N. Srinivasa Rao	Computer Science & Engineering	
3. T. Krishna Kishore	Computer Science & Engineering	
4. R. Hima Bindu	Cyber Security & Data Science	