Join Community by clicking below links

Website Link

For question papers and notes visit website

Click here

Telegram Channel

Click here

WhatsApp Channel

Click here

Join WhatsApp Group

Q1

A) What do you mean by regression? Explain the concept of Lasso Regression and Ridge Regression.

Ans:

Regression –

- Regression is a method used to predict a continuous value (like price, marks, salary, etc.) based on input variables.
- It finds the relationship between independent variables (X) and a dependent variable (Y).
- Regression builds a best-fit line or curve that represents how changes in inputs affect the output.
- It helps in forecasting and understanding patterns in data, making it useful in many real-world applications.

Example

Predicting house price using features like area, number of rooms, and location.

Ridge Regression –

- Ridge Regression helps reduce overfitting in linear regression models.
  It does this by controlling how large the model's coefficients can become, which prevents the model from fitting noise in the training data.
- A penalty term is added to the original cost function of linear regression.
  This penalty discourages the coefficients from becoming too large, making the model simpler and more stable.
- The penalty used in Ridge Regression is based on the sum of squared coefficients (L2 regularization).
  This means the larger the coefficient, the more penalty is applied, pushing coefficients to be smaller.
- Ridge Regression cost function is:
  [
  J(w) = \frac{1}{2} \sum (y - h(y))^2 + \sum |w|^2
  ]
  Here, the first part measures prediction error, and the second part is the L2 penalty term.

- As the penalty strength increases, the coefficients shrink toward zero. This reduces the model's variance and helps it generalize better to unseen data.
- Main drawback: Ridge Regression never completely removes any feature. The coefficients shrink but never become exactly zero, so the model keeps all features even if some are less useful.

Lasso Regression :

- Lasso regression (L1 regularization) is used to stop overfitting in linear regression models by adding a penalty term to the cost function.
- Unlike Ridge regression (which uses squared coefficients), Lasso uses the sum of the absolute values of the coefficients as the penalty.
- Lasso regression tries to minimize the following cost function:
  [
  J(w) = \frac{1}{2}\sum (y - h(y))^2 + \sum |w|
  ]
  where:
  1. y = actual value
  2. $h(y)$ = predicted value
  3. w = feature coefficient
- Lasso regression can reduce some coefficients to zero, effectively removing unnecessary features.
- This behaviour makes Lasso useful for feature selection, especially when working with high-dimensional datasets.
- Lasso is helpful when many features may be unimportant or redundant, as it automatically eliminates them.
- The final model becomes less complex and easier to understand because only important features remain.
- By reducing overfitting, Lasso often improves predictive performance of the model.

B) Describe the Bias–Variance trade-off and its relationship to overfitting and underfitting.

Ans:

What is Bias?

- Bias refers to the error introduced when a model makes overly simple assumptions about the data.
- A model with high bias does not learn the underlying pattern well.
- It tends to produce predictions that are consistently wrong, no matter how much data it is trained on.

Example:
Using a simple linear model to fit a complex non–linear dataset.

What is Variance?

- Variance refers to the model's sensitivity to small variations in the training data.
- A model with high variance learns noise along with the actual pattern.
- It performs very well on training data but poorly on unseen data because it fails to generalize.

Example:
A highly complex model with too many parameters memorizing the training data.

Bias–Variance Trade-off (Core Concept)

- The Bias–Variance trade-off is about finding the right balance between a model being too simple (high bias) and too complex (high variance).
- The goal is to build a model that:
  - learns the true patterns in the data (low bias)
  - does not overreact to noise (low variance)
- Achieving this balance gives the lowest total prediction error, resulting in good generalization.

Mathematically:

$$\text{Total Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

You cannot reduce both bias and variance at the same time—reducing one usually increases the other.
This is the trade-off.

Relationship to Underfitting and Overfitting

A) Underfitting (High Bias, Low Variance)

- Model is too simple.
- It cannot capture the pattern of the data.
- Performs poorly on training and testing data.
- High error because the model has failed to learn.

Causes of Underfitting:

- Not enough features
- Oversimplified model
- Too much regularization
- Linear model used for non-linear data

Example:
Predicting temperature with a straight-line model when the relationship is curved.

B) Overfitting (Low Bias, High Variance)

- Model is too complex.
- Learns the training data perfectly—including noise and random fluctuations.
- Very low training error, but high testing error.

Causes of Overfitting:

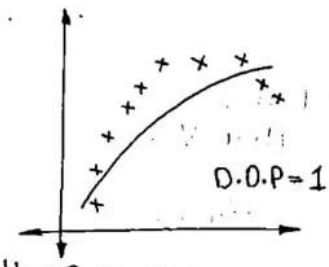
- Too many features
- Complex models (deep trees, higher-degree polynomials)
- Little or no regularization
- Small training dataset

Example:
A decision tree that splits until every training sample is perfectly classified.

Bais and Variance

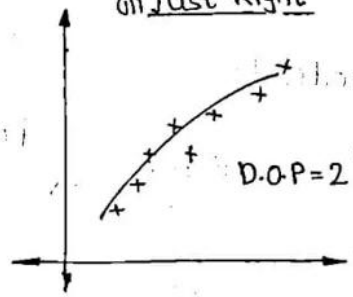
Regression:

(i) Underfitting



"High Bais, High Variance"

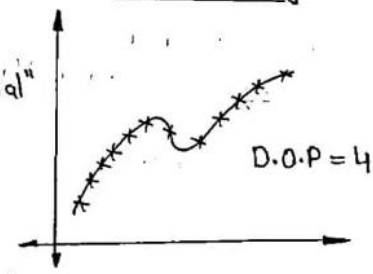
(ii) Just Right



"Low Bais, Low Variance"

(iii) Over-fitting

Note:
D.O.P =
"Degree of polynomial"



"Low Bais, High Variance"

(i) Model 1 (Underfitting): Train Accuracy ↓
Test Accuracy ↓

(ii) Model 3 (Overfitting): Train Accuracy ↑
Test Accuracy ↓

C) Write a short note on evaluation metrics. Explain MAE, RMSE, and R^2.

Evaluation Metrics :
Evaluation metrics are used to measure how well a regression model predicts the target variable. They help compare models, identify errors, and understand model performance.

---

1) MAE – Mean Absolute Error

- MAE measures the average absolute difference between actual values and predicted values.
- It shows how much error the model makes on average, without considering direction.
- MAE treats all errors equally (no extra penalty on large errors).
- It is simple, easy to interpret, and uses same units as the output variable.

Formula : MAE = \frac{1}{n} \sum |y - \hat{y}|

Interpretation

- MAE = 4 means: on average, predictions are 4 units away from actual values.
- Lower MAE = better performance.

---

2) RMSE – Root Mean Squared Error

- RMSE is the square root of the average squared errors.
- Unlike MAE, RMSE gives higher penalty to larger errors because errors are squared.
- It is useful when large prediction errors are unacceptable.
- RMSE is very sensitive to outliers, making it good for detecting large mistakes.

Formula : RMSE = \sqrt{\frac{1}{n} \sum (y - \hat{y})^2}

Interpretation

- RMSE = 6 means: predictions deviate by 6 units on average, with big errors greatly affecting the metric.
- Lower RMSE = better model.
- RMSE is usually greater than or equal to MAE.

3) R^2 – Coefficient of Determination

- R^2 measures how much of the variance in the target variable is explained by the model.
- It checks the model's overall goodness of fit.
- R^2 ranges from 0 to 1:
    1. 0 → Model explains nothing
    2. 1 → Model explains everything
- R^2 can also be negative if the model is worse than a horizontal line (mean predictor).

Formula : R^2 = 1 - \frac{SS_{res}}{SS_{tot}}

Interpretation

- $R^2 = 0.90$ means the model explains 90% of the variation in the data.
- Higher R^2 = better model.
- R^2 alone is not enough; should be checked along with MAE/RMSE.

Short Example :
Suppose actual values:
[10, 12, 13]
Predicted values:
[9, 11, 15]

1. Errors: 1, 1, 2
2. MAE: (1 + 1 + 2)/3 = 1.33
3. RMSE: sqrt((1² + 1² + 2²)/3) = sqrt(6/3) = 1.41
4. R^2: Higher value = better explanation of variance.

Q2)

A) What is linear regression? Explain least squares in the context of Linear Regression.

What is Linear Regression?

- Linear Regression is a statistical method used to **predict a continuous target variable (Y)** using one or more **input features (X)**.
- It assumes a **linear relationship** between input variables and the output.
- It fits a **straight line (for simple regression)** or a **hyperplane (for multiple regression)** through the data.
- The goal is to find the **best-fit line** that minimizes errors between predicted and actual values.

Equation of Linear Regression

For one feature (Simple Linear Regression):
$y = m x + c$

Where:

- **m** = slope
- **c** = intercept
- **x** = input variable
- **y** = predicted output

For Multiple Linear Regression: $y = w_1x_1 + w_2x_2 + ... + w_nx_n + b$

Purpose of Linear Regression

1. **Prediction** (e.g., salary, price, marks).
2. **Trend analysis** – understanding how variables are related.
3. **Forecasting** – based on past data.
4. **Understanding feature influence** on the output.

Least Squares in Linear Regression :

Definition

The **Least Squares Method** is a mathematical technique used to find the **best-fitting line** by minimizing the **sum of squared errors** between predicted and actual values.

What It Does

- Compares actual value (**y**) and predicted value (**ŷ**).
- Computes the **error** = (y − ŷ).
- Squares the error so that:
    1. Negative errors do not cancel positive ones.
    2. Larger errors receive a bigger penalty.
- Adds all squared errors and finds the line that **minimizes this total**.

Least Squares Formula (Cost Function)

$$J(w) = \frac{1}{2} \sum (y - \hat{y})^2$$

Where:

- **y** = actual value
- **ŷ** = predicted value
- **w** = model parameters (slope, intercept, coefficients)

How Least Squares Finds the Best Fit Line

1. Start with a random line.
2. Calculate total squared error.
3. Adjust slope (m) and intercept (c).
4. Keep adjusting until total error is minimized.
5. Final line is the **best fit line**.

Example

Suppose you want to predict house price based on area.

For each house:

- Actual price = 500k
- Predicted price = 480k
- Error = 500 − 480 = 20k
- Squared error = 20^2 = 400

Least squares will do this for all houses and choose the line that **makes the sum of squared errors minimum**.

B) What is Gradient Descent? Compare Batch Gradient Descent and Stochastic Gradient Descent.

1) What is Gradient Descent?

Gradient Descent is an **optimization algorithm** used to minimize a **cost function** in machine learning models, especially Linear Regression, Logistic Regression, and Neural Networks.

How it Works

- Gradient Descent starts with **initial random parameter values** (like weights).
- It calculates the **cost/error** of the model using the cost function.
- It computes the **gradient** (slope) of this cost function with respect to the parameters.
- Parameters are updated by **moving in the opposite direction of the gradient** because that direction **reduces the error**.
- This process repeats until the model reaches the **minimum error** or convergence.

Parameter Update Formula

$$w := w - \alpha \cdot \frac{\partial J(w)}{\partial w}$$

Where:

- w = model parameter
- α = learning rate
- $J(w)$ = cost function

---

2) Types of Gradient Descent

There are mainly two important types:

1. **Batch Gradient Descent (BGD)**
2. **Stochastic Gradient Descent (SGD)**

Aspect	Batch Gradient Descent (BGD)	Stochastic Gradient Descent (SGD)
Meaning	Uses the entire dataset to compute gradient for each update.	Uses only one training example to update parameters.
Parameter Updates	Updates parameters once p epoch (after full dataset).	Updates parameters after every sample (many updat per epoch).
Speed	Slow because it processes all data before updating.	Very fast, updates immediately after each dat point.
Gradient Quality	Stable, accurate gradient.	Noisy, fluctuating gradient
Convergence Behavior	Smooth and stable convergence path.	Zig-zag convergence but often faster.
Risk of Local Minim	Higher chance of getting stuck.	Lower chance because nois helps escape.
Memory Requirement	High — requires full datas in memory.	Very low — uses one sample at a time.
Best For	Small/medium datasets; high accuracy needed.	Large datasets; deep learning; online learning.
Error Surface Movement	Moves steadily in the true gradient direction.	Jumps around due to noise but heads toward minimum
Training Time	Long training time.	Shorter training time.
Computational Cost per Update	High.	Low.
Risk of Overshootin	Low.	Higher (due to noisy updates).

Q3

A) What is Multi-class Classification? Explain its variants.

1) What is Multi-class Classification?

Multi-class classification is a type of machine learning problem where the goal is to **classify an input into one of three or more possible classes**.

Key Points

1. It deals with **more than two categories** (e.g., classifying fruits into apple, banana, mango).
2. The model predicts **only one class at a time** from multiple options.
3. Algorithms like Logistic Regression, SVM, Decision Trees, Random Forest, and Neural Networks can be adapted for multi-class tasks.
4. Commonly used when the output variable is **categorical with more than two labels**.

Examples

- Classifying handwritten digits (0–9) → 10 classes
- Classifying animals into cat, dog, horse, cow → 4 classes
- Identifying types of flowers → multiple classes

Variants of Multi-class Classification

A) One-vs-Rest (OvR) / One-vs-All (OvA)

How it works

1. For **K classes**, build **K separate binary classifiers**.
2. Each classifier predicts whether a sample belongs to **that class** or **not**.
3. The classifier with the **highest output score** determines the final class.

Example

If classes = {A, B, C}, create:

- Classifier 1: A vs not-A
- Classifier 2: B vs not-B
- Classifier 3: C vs not-C

Advantages

- Simple and easy to implement
- Works well with algorithms like Logistic Regression, SVM

Disadvantages

- Can struggle when classes are imbalanced
- Predictions may overlap or conflict

B) One-vs-One (OvO)

How it works

1. For **K classes**, build classifiers for **every pair of classes**.
2. Total models = **K(K – 1) / 2**.
3. Each classifier predicts between **two classes only**.
4. Final class is chosen by **majority voting**.

Example

If classes = {A, B, C}, build:

- Model 1: A vs B
- Model 2: A vs C
- Model 3: B vs C

Advantages

- Small, simple classifiers
- Works well for SVMs

Disadvantages

- Large number of models for big K
- Slower prediction time

C) Softmax (Multinomial Logistic Regression)

How it works

1. Uses a **single model** that outputs probabilities for all classes.
2. Uses the **Softmax function** to convert raw scores into probabilities.
3. The class with the **highest probability** is selected.

Formula

$$
P(y = k \mid x) = \frac{e^{z_k}}{\sum_{i=1}^{K} e^{z_i}}
$$

Advantages

- Single model handles all classes
- Produces **probability distribution** across classes
- Standard method in **neural networks** and deep learning

Disadvantages

- Requires more computation
- Needs large data to perform well

B) Explain K-NN algorithm. What are its advantages and disadvantages?

K-NN Algorithm

1. Definition

- K-NN (K-Nearest Neighbors) is a **supervised learning algorithm** used for **classification** and **regression**.
- It is called "lazy learner" because it **does not build a model** during training; it simply **stores the data**.

How K-NN Works

1. **Choose the value of K**
    - K = number of nearest neighbors you want to consider.
2. **Calculate distance** between the new data point and all existing data points.
    - Common distances: Euclidean, Manhattan, Minkowski.
3. **Sort distances** in increasing order.
4. **Select K nearest neighbors** (the top K smallest distances).
5. **Perform Voting (for Classification)**
    - The class that appears most among the K neighbors becomes the decision.
6. **Take Average (for Regression)**
    - The prediction is the mean/average value of the K closest neighbors.
7. **Return Output**
    - K-NN gives the final class/label/value.

Example :

- Suppose K = 3
- Three nearest neighbors → **A, A, B**
- Majority is class **A** → prediction = **A**

K-NN Advantages :

1. **Very simple and intuitive** to understand.
2. **No training time** → fast training because it stores data only.
3. **Works well with small datasets**.
4. **Good for non-linear datasets**, because it makes no assumption about data distribution.
5. **Naturally handles multi-class classification**.
6. **Can adapt based on K value** (flexible).

K-NN Disadvantages :

1. **Slow at prediction time**
    - Because distance must be computed for all data points.
2. **Memory-heavy (High storage)**
    - Needs to store entire dataset.
3. **Sensitive to irrelevant or unscaled features**
    - Requires **feature normalization** (e.g., Min-Max, Standardization).
4. **Poor performance in high-dimensional data**
    - Suffers from **curse of dimensionality**.
5. **K value selection is difficult**
    - Small K → noisy
    - High K → too generalized
6. **Computationally expensive** for large datasets.

C) What are different distance metrics used in K-NN? Explain any three.

K-NN commonly uses the following distance measures:

1. Euclidean Distance
2. Manhattan Distance
3. Minkowski Distance
4. Hamming Distance
5. Chebyshev Distance
6. Cosine Similarity Distance
7. Mahalanobis Distance

1️⃣ Euclidean Distance :

Definition

- Measures the **straight-line** or **shortest** distance between two data points.
- It is the **default metric** for most K-NN implementations.

Formula

$$d = \sqrt{\sum (x_i - y_i)^2}$$

- It treats all features equally and assumes all features contribute proportionally.
- Very effective when feature scales are similar.
- **Sensitive to large differences**: big values impact the distance more.
- Easily influenced by **outliers**, because squaring amplifies large deviations.
- Works well when data is in **low-dimensional space**.
- Mirrors actual geometric (physical) distance.
- Often used in:
    - Image processing

- Face recognition
- Handwriting recognition
- Clustering algorithms like K-Means

2️⃣ Manhattan Distance (L1 Distance)

Definition

- Measures the distance by summing the **absolute** differences between coordinates.
- Called **City Block Distance** because movement is like city roads (up/down/left/right), not diagonal.

Formula

$$d = \sum |x_i - y_i|$$

Detailed Points

- Less sensitive to outliers because it **does not square** differences.
- Better for datasets with **high dimensionality** (100+ features).
- Works well when features are **sparse** (most values are zero), such as:
  - Text data
  - Document similarity
  - Bag-of-words models
- More robust when data has **extreme values** or noise.
- Produces simpler decision boundaries.
- Often used in:
  - NLP
  - Recommendation systems
  - Anomaly detection

✅ 3️⃣ Minkowski Distance — (Generalized Distance Metric)

Definition

- A **general form** of Euclidean and Manhattan distances.
- Uses a parameter **p** to adjust the type of distance.

Formula

$$d = \left(\sum |x_i - y_i|^p \right)^{1/p}$$

Special Cases

- If **p = 1 → Manhattan Distance**
- If **p = 2 → Euclidean Distance**

Detailed Points

- Very flexible: you can choose different **p values** to control the sensitivity.
- For **p > 2**, the distance becomes even more sensitive to large differences.
- As **p increases**, the metric gives more weight to **large deviations** in features.
- Helps tune K-NN for **best accuracy** with hyperparameter optimization.
- Works when the relationship between features is not clearly known.
- Used in:
    - Machine learning model tuning
    - K-NN optimization
    - Search and recommendation tasks
- Ideal for datasets where you want to experiment with multiple distance behaviors

Q 4)

A) Explain kernel methods suitable for SVM with examples.

- A kernel is a mathematical function used in SVM to transform data from low-dimensional space to high-dimensional space.
- It helps SVM classify data that is not linearly separable.
- Kernels allow SVM to create complex decision boundaries without explicitly converting data to high dimensions (this is called the kernel trick).

1️⃣ What is a Linear Kernel?

- A **Linear Kernel** is the simplest kernel used in Support Vector Machines (SVM).
- It does **not** transform data into higher dimensions.
- Instead, it uses the **dot product** of two feature vectors to measure similarity.

✓ **Formula:**

$$K(x, y) = x^T y$$

This means:
Similarity = multiplication of features of x and y.

When is a Linear Kernel Used?

The Linear Kernel is used when the data is:

- **Linearly separable** (can be separated with a straight line).
- **High-dimensional** (many features, e.g., text data).
- Not very complex or curved.

2 Polynomial Kernel

- The Polynomial Kernel allows SVM to create **curved decision boundaries**.
- It maps input data into a **polynomial feature space**.
- This helps classify data where the relationship between features is non-linear but follows a polynomial pattern.

✓ **Formula:**

$$K(x, y) = (x^T y + c)^d$$

Where:

- **d** = degree of polynomial
- **c** = constant (controls flexibility)

When is Polynomial Kernel Used?

- When the data shows **non-linear patterns**.
- When the model requires **curved boundaries**.
- When input features interact (e.g., x1 × x2).

3 RBF Kernel

- The **most widely used kernel** in SVM.
- Maps data into an **infinite-dimensional space**.
- Perfect for datasets with complex, irregular boundaries.

✓ **Formula:**

$$K(x, y) = e^{-\gamma \|x-y\|^2}$$

Where:

- **γ (gamma)** controls how tightly the curve fits.

When is RBF Kernel Used?

- When data is **highly non-linear**.

- When patterns are not clear or structured.
- When we don't know which kernel to choose → **RBF is default**.

Real-Life Examples

- Face recognition
- Medical diagnosis
- Fraud detection
- Weather pattern classification

4️⃣ Sigmoid Kernel?

- The Sigmoid Kernel behaves like a **two-layer neural network activation function**.
- Not commonly used in SVMs but useful for some patterns.

Formula:

$$K(x, y) = \tanh(\alpha x^T y + c)$$

When is Sigmoid Kernel Used?

- When data behaves like neuron activations.
- When dataset has patterns similar to neural networks.

How It Works (Step-by-Step)

1. Computes dot product and applies a **tanh activation**.
2. Produces non-linear separation.
3. Useful for datasets with neural-network-like structures.

B) What are different techniques used for outlier handling? Explain any two.

Different Techniques Used for Outlier Handling

1. Z-Score Method (Standard Deviation Method)
2. IQR Method (Interquartile Range)
3. Winsorization
4. Log Transformation / Scaling
5. Capping (Percentile Capping)
6. Removing Outliers Manually
7. Clustering-based Detection (K-Means, DBSCAN)
8. Model-based Detection (Isolation Forest, LOF)

1️⃣ Z-Score Method (Standard Deviation Method)

Meaning

- Z-score measures how far a data point is from the mean in terms of standard deviations.
- Helps identify points that are unusually high or low.

✓ Formula:

$$Z = \frac{(x - \mu)}{\sigma}$$

Where:

- x = data point
- μ = mean
- σ = standard deviation

How It Works

1. Calculate mean and standard deviation of the data.
2. Compute the Z-score for each value.
3. If Z-score > +3 or < –3 → treat as an outlier.
4. Handle them by removal, capping, or replacing with mean/median.

Advantages

- Easy to calculate.
- Works well for normally distributed data.

Disadvantages

- Doesn’t work well when data is not normally distributed.

- Sensitive to extreme values.

2️⃣ IQR Method (Interquartile Range)

Meaning

- IQR = difference between 75th percentile (Q3) and 25th percentile (Q1).
- Detects outliers based on spread of middle 50% of data.

Formula: IQR = Q3 - Q1

Outlier thresholds:

- Lower bound = Q1 − 1.5 × IQR
- Upper bound = Q3 + 1.5 × IQR

Any value outside this range = outlier.

How It Works

1. Sort the data.
2. Find Q1 and Q3.
3. Compute IQR.
4. Find upper & lower limits.
5. Outliers are those falling outside these limits.

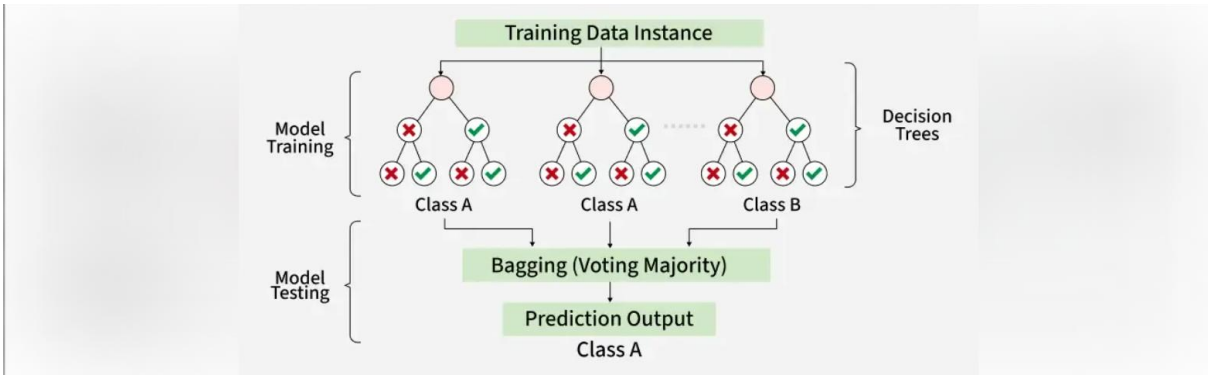
Advantages

- Works very well for skewed and non-normal data.
- Not affected by extreme values.

Disadvantages

- Not suitable for very small datasets.

C) With a suitable diagram, explain the Random Forest algorithm and the concept of bagging.



Random Forest Algorithm

The Random Forest algorithm extends bagging by introducing an additional layer of randomness, specifically in the feature selection process during the construction of each decision tree.

- **Bagging:**

  Multiple bootstrap samples are drawn from the original training data, as described in the bagging concept.

- **Random Feature Subsets:**

  For each individual decision tree, at each split point during its construction, only a random subset of features is considered for finding the best split, rather than all available features. This further decorrelates the individual trees, making the ensemble more robust.

- **Individual Tree Training:**
  A full decision tree is grown on each bootstrapped sample, using the random feature subset selection at each split.

- **Aggregation:**
  For a new data point, each tree in the forest makes a prediction. For classification, the final prediction is determined by a majority vote among the trees. For regression, the final prediction is the average of the predictions from all the trees.

Bagging (Bootstrap Aggregating)

Bagging is an ensemble meta-algorithm that improves the stability and accuracy of machine learning algorithms, reducing variance and helping to avoid overfitting. It involves two main steps:

- Bootstrapping:

  From the original training dataset, multiple new datasets are created by sampling with replacement. Each new dataset has the same size as the original, but due to sampling with replacement, some original data points may appear multiple times in a bootstrapped sample, while others may not appear at all.

- Aggregating:

  A base learner (e.g., a decision tree) is trained independently on each of these bootstrapped datasets. For a new data point, the predictions from all the individual models are combined (e.g., averaged for regression, or majority vote for classification) to produce a final, more robust prediction.

Q5)

A) Why is K-Medoid used? Explain the K-Medoid algorithm.

K-Medoid is used because it is a more robust clustering algorithm compared to K-Means, especially when the dataset contains noise, outliers, or non-Euclidean distances.

Reasons to use K-Medoid :

1. Robust to outliers
    - K-Means uses the mean, which can be pulled by extreme values.
    - K-Medoid uses medoids (actual data points), so outliers have less impact.

2. Works with any distance metric
    - K-Means works mainly with Euclidean distance.
    - K-Medoid can use Manhattan, cosine, Hamming, Jaccard, etc.

3. Better for categorical datasets
    - Since medoids are real data points, mixed or categorical data works better.

4. More stable clusters
    - Cluster centers do not move drastically between iterations.

5. Useful in applications where interpretability matters
    - Because the medoid is an actual sample, it is easy to interpret.

K-Medoid Algorithm (PAM: Partitioning Around Medoids)
The most famous version is PAM – Partitioning Around Medoids.

Step-by-Step Explanation :

1. Choose K initial medoids
    - Randomly select K actual data points as medoids.

2. Assign each data point to its nearest medoid
    - Based on chosen distance measure
      (e.g., Manhattan, Euclidean, Cosine, etc.)

3. Update medoids (swap step)
For each cluster:
    - Try swapping the medoid with a non-medoid point.
    - Compute the total cost (sum of distances).
    - If the cost reduces → accept the swap (new medoid).

4. Repeat
    - Continue assigning points + swapping
      until no improvement in total cost.

5. Final clusters formed
    - When no more swaps reduce cost.

B) Why is density-based clustering used? Explain any one method (DBSCAN / OPTICS / DENCLUE)

Density-based clustering is used because it can identify arbitrary-shaped clusters and outliers, doesn't require the number of clusters to be specified in advance, and works well with noisy data

Why density-based clustering is used

- Arbitrary-shaped clusters: It can find clusters of any shape, not just the spherical shapes often found by other methods like k-means.
- Outlier detection: It can distinguish between dense clusters and sparse areas, automatically identifying and removing noise and outliers.
- No need to pre-specify cluster count: Unlike some algorithms, you don't need to know the number of clusters beforehand.
- Handles noise: It is robust to noise in the data, as points in low-density regions are simply marked as noise

DSBSCAN :

DBSCAN is a density-based clustering algorithm that groups data points that are closely packed together and marks outliers as noise based on their density in the feature space. It identifies clusters as dense regions in the data space separated by areas of lower density.

Key Parameters in DBSCAN

1. eps: This defines the radius of the neighborhood around a data point. If the distance between two points is less than or equal to eps they are considered neighbors. A common method to determine eps is by analyzing the k-distance graph. Choosing the right eps is important:

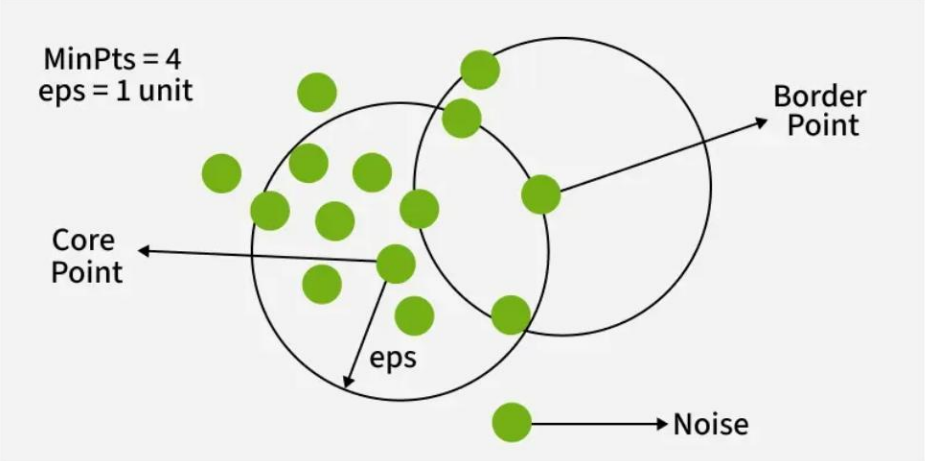
- If eps is too small most points will be classified as noise.
- If eps is too large clusters may merge and the algorithm may fail to distinguish between them.

2. MinPts: This is the minimum number of points required within the **eps** radius to form a dense region. A general rule of thumb is to set MinPts >= D+1 where **D** is the number of dimensions in the dataset.

How Does DBSCAN Work?

DBSCAN works by categorizing data points into three types:

1. Core points which have a sufficient number of neighbors within a specified radius (eplison)
2. Border points which are near core points but lack enough neighbors to be core points themselves
3. Noise points which do not belong to any cluster.



DBSCAN ALgorithm

(Optional)

Steps in the DBSCAN Algorithm

1. **Identify Core Points**: For each point in the dataset count the number of points within its eps neighborhood. If the count meets or exceeds MinPts mark the point as a core point.
2. **Form Clusters**: For each core point that is not already assigned to a cluster create a new cluster. Recursively find all density-connected points i.e points within the eps radius of the core point and add them to the cluster.
3. **Density Connectivity**: Two points a and b are density-connected if there exists a chain of points where each point is within the eps radius of the next and at least one point in the chain is a core point. This chaining process ensures that all points in a cluster are connected through a series of dense regions.
4. **Label Noise Points**: After processing all points any point that does not belong to a cluster is labeled as noise.

C) Apply the K-Means algorithm to cluster the given points into three clusters.

Q6)

A) Explain Hierarchical Clustering with an example. Differentiate Agglomerative vs Divisive.

Answer :

Hierarchical Clustering is an unsupervised learning method used to group similar data points into clusters based on their distance or similarity. Instead of choosing the number of clusters in advance, it builds a tree-like structure called a dendrogram that shows how clusters merge or split at different levels. It helps identify natural groupings in data and is commonly used in pattern recognition, customer segmentation, gene analysis and image grouping

Imagine we have four fruits with different weights: an apple (100g), a banana (120g), a cherry (50g) and a grape (30g). Hierarchical clustering starts by treating each fruit as its own group.

- Start with each fruit as its own cluster.
- Merge the closest items: grape (30g) and cherry (50g) are grouped first.
- Next, apple (100g) and banana (120g) are grouped.
- Finally, these two clusters merge into one.

Finally all the fruits are merged into one large group, showing how hierarchical clustering progressively combines the most similar data points

Types of Hierarchical Clustering

1. Agglomerative Clustering
2. Divisive clustering

Parameters	Agglomerative Clustering	Divisive Clustering
Approach	Bottom-up: Starts with individual points and merges them.	Top-down: Starts with all data in one cluster and splits.
Complexity Level	More computationally expensive due to pairwise distance calculations.	Less computationally expensive but requires careful cluster splitting.
Handling Outliers	Better at handling outliers, as outliers can be absorbed into larger clusters.	Outliers may lead to inefficient splitting and suboptimal results.
Interpretability	More interpretable due to clear cluster merging shown in the dendrogram.	Harder to interpret due to recursive splitting decisions.
Implementation	Scikit-learn provides multiple linkage methods (Ward, Complete, Average, Single).	Not widely implemented in Scikit-learn or SciPy.
Example Applications	Image segmentation, customer segmentation, document clustering, gene expression analysis.	Less common; used in hierarchical data analysis or domain-specific studies.

C) What is Outlier Analysis? Explain methods used for outlier detection in clustering.

Outlier Analysis (also called **anomaly detection**) refers to the process of identifying data points that are **significantly different** from the majority of the data.
These points deviate from normal patterns and may indicate:

- Errors in data
- Rare events
- Fraudulent activities
- Unusual behavior
- Noise in the dataset

In clustering, outliers are the points that **do not belong strongly to any cluster** because they lie **far from dense regions**.

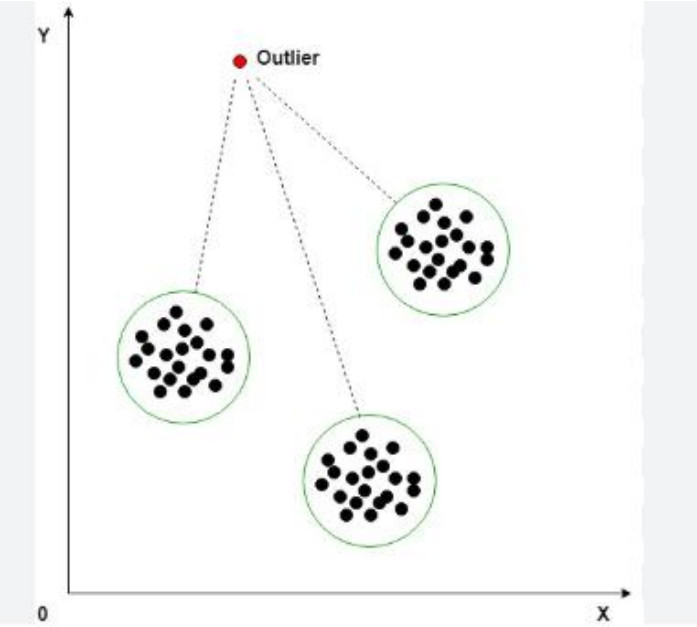


Fig : Outlier

Methods used for outlier detection in clustering.

1. **Density-Based Methods (DBSCAN)**
2. **Distance-Based Outlier Detection (K-Means)**
3. **LOF (Local Outlier Factor)**
4. **Hierarchical Clustering-Based Outlier Detection**

1. Density-Based Method (DBSCAN)

Detailed Points:

1. DBSCAN groups points based on density
    - A region is considered a cluster if it has enough points close to each other.
    - This helps separate dense clusters from sparse/noisy areas.
2. Uses two parameters: Eps and MinPts
    - Eps: How far we look around a point (radius).
    - MinPts: Minimum number of points needed to form a dense region.
3. Outlier Identification
    - If a point does not have enough neighbors within Eps, it cannot form or join a cluster.
    - Such isolated points are labeled noise, which are considered outliers.
4. Why it works well
    - Outliers naturally lie in low-density areas.
    - DBSCAN automatically separates them without extra logic.

---

2. Distance-Based Method (K-Means)

Detailed Points:

1. K-Means forms clusters around centroids
    - After clustering, each point is assigned to the nearest centroid.
2. Outlier Detection Using Distance
    - Calculate how far each point is from its cluster centroid.
    - Very large distance = unusual point = outlier.
3. Threshold Criteria
    - Points in the top 5% or 10% farthest distances are labeled as outliers.
    - Or: If distance > (mean distance + 2×standard deviation).
4. Why it helps find outliers
    - Normal points stay close to centroid.
    - Outliers remain far away because they don't fit well with any cluster.

---

3. LOF (Local Outlier Factor)

Detailed Points:

1. LOF compares a point's density with the density of its neighbors
    - Normal points: Similar density as neighbors.
    - Outliers: Much lower density (more isolated).
2. Local Reachability Density (LRD)
    - Measures how close a point is to its neighbors.
    - Low LRD = point is in a sparse region → likely outlier.
3. LOF Score Calculation
    - LOF = Average density of neighbors ÷ density of the point.
    - If LOF > 1 → lower density than neighbors.
    - If LOF > 1.5 or 2 → strong outlier.
4. Why LOF is useful
    - Detects local anomalies, even when the dataset has clusters with different densities.

- More accurate for real-world uneven clusters.

"DBSCAN,K-Means has already been explained in detail in the previous answer. Kindly refer to that section for the full explanation."

D) Differentiate K-Means and Spectral Clustering. Explain when to use Spectral Clustering.

Parameter	K-Means Clustering	Spectral Clustering
Approach	Partitioning-based algorithm	Graph-based algorithm using eigenvalues/eigenvectors
Type of Clusters	Finds spherical/convex clusters	Finds arbitrarily shaped, complex clusters
Input Requirements	Works directly on raw data	Works on similarity (affinity) matrix
Cluster Shape Limitations	Struggles with non-linear boundaries	Handles non-linear, complex manifold structures
Algorithm Dependence	Depends on centroid initialization	Depends on graph construction and eigen decomposition
Computational Cost	Low (O(n)) → efficient for large datasets	High ($O(n^3)$) → costly for large datasets
Sensitivity to Noise	Sensitive to outliers (centroid shifts easily)	More robust due to similarity graph structure
Number of Clusters (k)	Must be known beforehand	Also requires k, but more accurate separation
Best For	Large, simple, well-separated clusters	Complex shapes, non-linear structures, image segmentation

When to Use Spectral Clustering :

Use Spectral Clustering when:

1. Clusters Have Irregular or Non-Convex Shapes
   - Ideal for datasets where clusters are curved or connected in complex ways
     (e.g., spiral shapes, rings, intertwined patterns).

2. Data Cannot Be Separated with a Straight Line
   - Spectral clustering uses eigenvectors to project data into a space where clusters become separable.

3. You Have a Similarity Matrix Instead of Raw Data
   - Works well when you can define similarity (e.g., cosine similarity, RBF kernel).

4. K-Means Fails Due to Non-Linear Boundaries
   - K-Means assumes spherical clusters, spectral clustering does not.

5. Applications Involving Graphs or Images
   - Performs extremely well in:
     - Image segmentation
     - Social network community detection
     - Document/topic grouping
     - Speech separation
     - Bioinformatics

6. Datasets With Small to Medium Size
   - Because eigen-decomposition is expensive, it is ideal for datasets up to a few thousand samples.

BONUS QUESTION FROM THIS UNIT :

Compare Hierarchical clustering and K-means clustering.

Parameter	Hierarchical Clustering	K-Means Clustering
Approach	Builds a hierarchy of clusters (top-down or bottom-up).	Partitions data into *k* clusters based on centroids.
Cluster Structure	Forms a **tree-like structure (dendrogram)**.	Forms **flat, non-hierarchical** clusters.
Number of Clusters (k)	Not required initially; chosen later by cutting the dendrogram.	Must specify **k** before running the algorithm.
Cluster Shape	Can detect **arbitrary-shaped** and nested clusters.	Works best for **spherical/convex** clusters.
Computation Cost	More expensive ($O(n^2)$ or higher). Not suitable for very large datasets.	Fast and efficient (O(n)), good for large datasets.
Memory Usage	High, since distance matrix must be stored.	Low memory requirement.
Sensitivity to Noise	More robust; small linkages can separate outliers.	Sensitive to outliers; centroid shifts easily.

Q 7)

A) Explain Radial Basis Function networks and its building blocks

Radial Basis Function Network (RBFN) is a type of **artificial neural network** used mainly for **function approximation**, **regression**, **classification**, and **time-series prediction**.
It is similar to a feed-forward network but has a special hidden layer that uses **radial basis functions** (mostly Gaussian functions).

RBF Networks work in **three layers**:

1. **Input Layer**
2. **Hidden Layer (with RBF neurons)**
3. **Output Layer (linear neurons)**

Why is it called "Radial Basis"?

Because the activation of each hidden neuron depends on the **distance** between the input vector and the **center** of that hidden neuron.

Example:
If the input is close to the center → high activation
If the input is far from the center → low activation

This creates **localized responses**, unlike sigmoid networks that respond globally.

Building Blocks of RBF Networks

1. Input Layer

- Passes input features directly to the hidden layer.
- No computation is done here.
- If input has *n features*, input layer has *n nodes*.

---

2. Hidden Layer (Core of RBF Network)

This layer uses **Radial Basis Functions**—normally **Gaussian functions**.

Each hidden neuron has:

1. **Center (Ci)**

- A reference point representing where the neuron should respond the most.
- Centers are usually chosen using **K-means clustering**.

2. **Spread / Width (σi)**
   - Controls how wide the Gaussian curve is.
   - Larger σ → broader response
   - Smaller σ → localized response
3. **Activation Function**
   The most used RBF function is the **Gaussian Function**:

$$\phi_i(x) = e^{-\frac{||x-C_i||^2}{2\sigma_i^2}}$$

Where:

- x= input vector
- C_i= center of the i-th neuron
- σ_i= width of the Gaussian
- $\phi_i(x)$= output of RBF neuron

Interpretation

- If $x$is close to C_i→ activation ≈ 1
- If far → activation ≈ 0

This makes RBF models **fast** and **highly interpretable**.

---

3. Output Layer

- Takes weighted sum of hidden layer outputs.
- Activation is **linear**.

$$y = \sum_{i=1}^{m} w_i \phi_i(x)$$

Output Layer Characteristics:

- **Simple linear combination**

- Training is fast (uses least squares)
- Good for both **classification** and **regression**

B) Explain Recurrent Neural Networks and its applications in brief

A **Recurrent Neural Network (RNN)** is a type of neural network designed to process **sequential data** such as text, speech, time-series, video frames, or sensor readings.

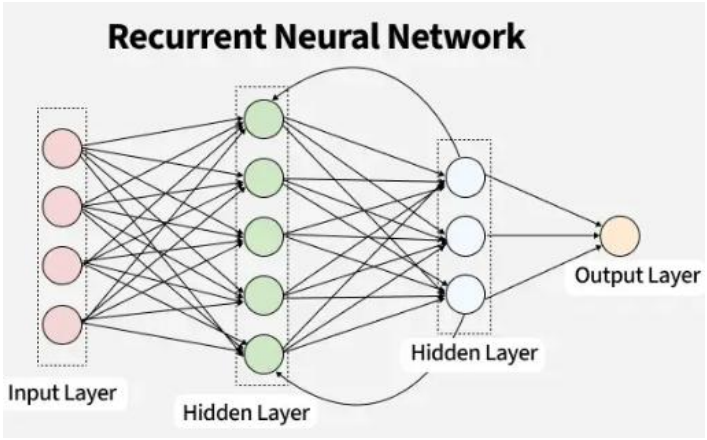
Unlike feedforward networks, an RNN has **feedback connections**, meaning:

The output of the previous step becomes an input for the next step.

This creates a **memory of past inputs**, allowing the network to learn patterns over time.

How RNNs work

- **Sequential processing:** RNNs process data one step at a time. At each step, they take an input and an activation from the previous step to produce an output and an updated "hidden state".
- **Memory:** The hidden state acts as the network's memory, carrying information from past inputs to help inform current predictions. This "memory" allows the network to understand context and dependencies in a sequence, such as the relationship between words in a sentence.
- **Looping mechanism:** The "recurrent" nature comes from a loop within the network, where the output of one step is fed back as an input to the next, creating a cycle that allows information to persist over time.



Applications of RNNs

- **Natural Language Processing (NLP):**
    - **Machine Translation:** Translating text from one language to another.
    - **Language Modeling:** Predicting the next word in a sentence for tasks like text generation and virtual assistants.
    - **Sentiment Analysis:** Determining the emotional tone of a piece of text.

- **Speech:**
    - **Speech Recognition:** Converting spoken language into written text.
    - **Text-to-Speech:** Generating human-like speech from text.

- **Time-Series Analysis:**
    - **Forecasting:** Predicting future values in a sequence, such as stock prices or weather patterns.

- **Other applications:**
    - **Music Generation:** Creating new sequences of musical notes.
    - **Video Analysis:** Understanding actions and events in video sequences.

C) What is multilayer perceptron? Describe with diagram.

Multi-Layer Perceptron (MLP) consists of fully connected dense layers that transform input data from one dimension to another.

It is called multi-layer because it contains an input layer, one or more hidden layers and an output layer.

The purpose of an MLP is to model complex relationships between inputs and outputs.

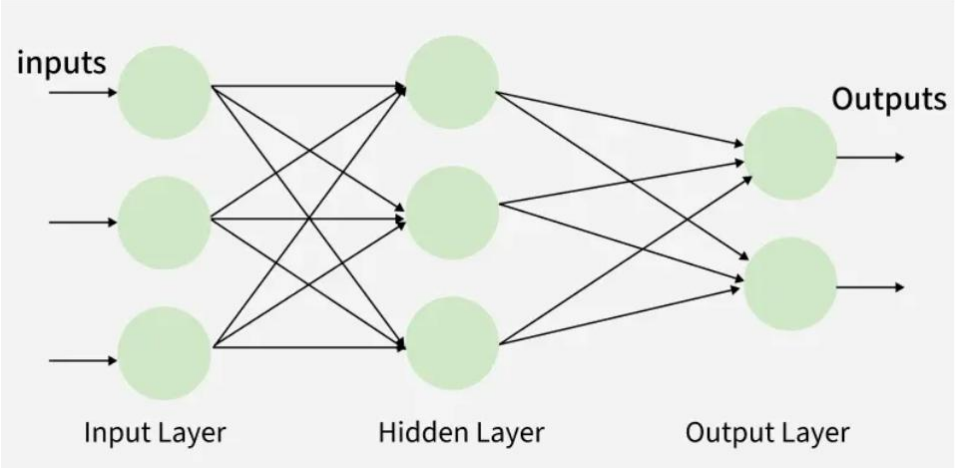


Fig: MLP

- **Input Layer**: Each neuron or node in this layer corresponds to an input feature. For instance, if you have three input features the input layer will have three neurons.
- **Hidden Layers**: MLP can have any number of hidden layers with each layer containing any number of nodes. These layers process the information received from the input layer.
- **Output Layer**: The output layer generates the final prediction or result. If there are multiple outputs, the output layer will have a corresponding number of neurons.

Every connection in the diagram is a representation of the fully connected nature of an MLP. This means that every node in one layer connects to every node in the next layer. As the data moves through the network each layer transforms it until the final output is generated in the output layer.

How It Works :

- **Forward Propagation:**

Input data flows from the input layer, through the hidden layers, to the output layer. At each neuron, a weighted sum of inputs is calculated, a bias is added, and an activation function is applied.

- **Backpropagation:**

The output of the network is compared to the true target values to calculate an error. This error is then propagated backward through the network, and the weights and biases of the connections are adjusted to minimize this error using optimization algorithms like gradient descent. This process allows the MLP to learn and improve its performance over time.

Q8)

A) draw and Explain Convolution Neural Network (CNN) with suitable example.

- **Convolutional Neural Network (CNN)** is an advanced version of **artificial neural networks (ANNs)**, primarily designed to extract features from grid-like matrix datasets.
- This is particularly useful for visual datasets such as images or videos, where data patterns play a crucial role.
- CNNs are widely used in **computer vision** applications due to their effectiveness in processing visual data.
- CNNs consist of multiple layers like the input layer, Convolutional layer, pooling layer, and fully connected layers.

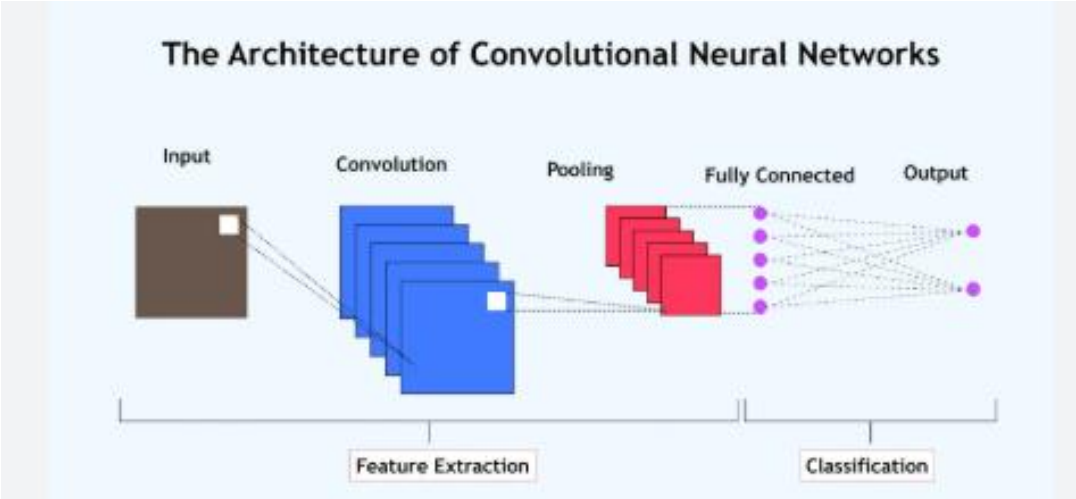


Fig :CNN Architecture

1. Input Layer

- Receives the **raw image data**.
- Input is usually a **3D tensor** → *(height × width × channels)*
  Example: 224 × 224 × 3 for an RGB image.
- No computation happens here — it only passes data to the next layer.

---

2. Convolutional Layer

- **Core layer of CNN** responsible for feature extraction.
- Applies **learnable filters/kernels** that slide across the image.
- Each filter performs a **dot product** with local regions of the input.
- Produces **feature maps** representing detected patterns.

- Different filters detect different features like:
  - Edges
  - Corners
  - Textures
  - Shapes
- Example: A *vertical-edge* filter produces high values where vertical edges appear.

---

3. Activation Layer (ReLU)

- Adds **non-linearity** to the network.
- ReLU operation:

$$f(x) = \max(0, x)$$

- Negative values become **0**, while positive values remain unchanged.
- Helps the network learn complex and non-linear patterns.

---

4. Pooling Layer (Max Pooling / Average Pooling)

- **Reduces spatial dimensions** of feature maps (height & width).
- Lowers the number of parameters → faster & less memory.
- Makes features **translation-invariant** (small shifts don't affect detection).
- **Max Pooling (2×2)** picks the *maximum value* in each window.
- Helps retain the most important/strongest features.

---

5. Fully Connected (Dense) Layers

- After convolution + pooling, feature maps are **flattened** into a 1D vector.
- Each neuron connects to all neurons from the previous layer.
- Learns **high-level relationships** between extracted features.
- Final dense layer gives **class scores**.
- Uses **softmax** activation for multi-class classification to output probabilities.

B) What are different activation function used is NN?

1) Sigmoid Activation Function

Meaning

Sigmoid converts any real number into a value between **0 and 1**.
It behaves like a **probability mapping** and is mainly used when the output must represent a probability.

Formula

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

Where It Is Used

- Binary classification output layer
- Logistic regression
- Models where output must be between 0 and 1

Example

If input = 2

$$\sigma(2) \approx 0.88$$

Meaning the model predicts 88% probability.

---

2) Tanh (Hyperbolic Tangent)

Meaning

Tanh is similar to sigmoid but outputs values between **–1 and +1**.
It is **zero-centered**, making it suitable for hidden layers.

Formula

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

Where It Is Used

- In hidden layers of simple neural networks
- In problems where negative outputs are useful

- In networks requiring symmetric activation

Example

If input = 2

$$\tanh(2) \approx 0.964$$

---

3) ReLU (Rectified Linear Unit)

Meaning

ReLU outputs **0 for negative values** and **x for positive values**.
This makes the network learn faster and handle deep layers efficiently.

Formula

$$ReLU(x) = \max(0, x)$$

Where It Is Used

- Hidden layers of CNNs
- Deep learning architectures
- MLPs and GANs
- Almost all modern neural networks

Example

x = –5 → 0
x = 4 → 4

---

4) Leaky ReLU

Meaning

Leaky ReLU is a modification of ReLU that allows a **small negative value** when the input is negative.
This helps keep neurons active instead of becoming stuck.

Formula

$$LeakyReLU(x) = \begin{cases} x & x > 0 \\ 0.01x & x < 0 \end{cases}$$

Where It Is Used

- CNN layers
- Deep models where small negative gradients are needed
- Situations where ReLU sometimes outputs many zeros

Example

x = –8 → –0.08
x = 8 → 8

---

5) Softmax Activation Function

Meaning

Softmax converts a vector of scores into a **probability distribution**.
Each output represents the probability of belonging to a class, and all probabilities sum to **1**.

Formula

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_j \quad e^{x_j}}$$

Where It Is Used

- Multi-class classification output layer
  (e.g., classify images into cat/dog/bird)
- NLP text classification
- Multi-class logistic regression

Example

If output = [2, 1, 0],
Softmax ≈ [0.67, 0.24, 0.09]

C) Explain ANN with it's Architecture

An **Artificial Neural Network (ANN)** is a computational model inspired by the **human brain**.
It consists of interconnected units called **neurons** that work together to process information, learn patterns from data, and make predictions or decisions.

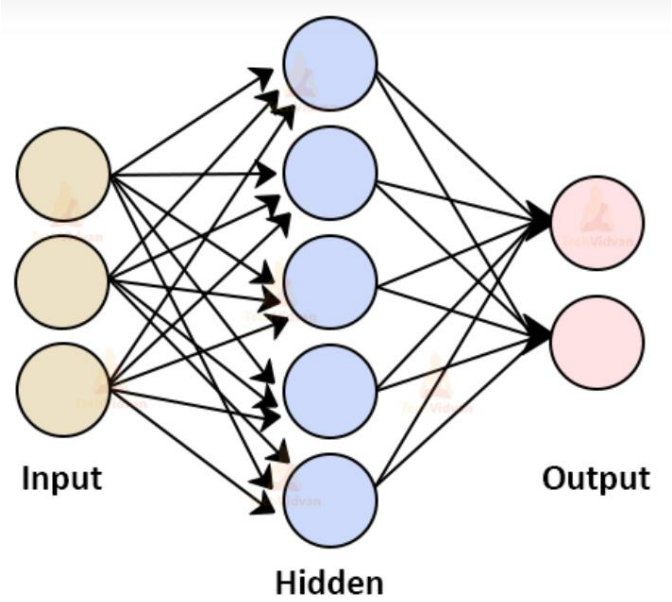


Fig: **Artificial Neural Network (ANN)**

1. Layers

a) Input Layer

- This is the first layer of ANN.
- It receives the raw input data (features).
- Each input neuron simply passes the information to the next layer.

b) Hidden Layer

- Performs all major computations on the input data.
- Contains neurons with **weights, biases, and activation functions**.
- Extracts patterns, relationships, and features from input.
- Can have one or multiple hidden layers depending on the complexity.

c) Output Layer

- Produces the final result of the neural network (classification, prediction, etc.).
- Uses suitable activation functions like Softmax, Sigmoid, etc.

---

2. Components of ANN

a) Weights

b) Bias

c) Activation Function

d) Cost (Loss) Function

3. Working of ANN (Step-by-Step)

Step 1: Input Feeding

- Raw data is fed into the **input layer**.

Step 2: Weight Assignment

- Each connection from input to hidden layer is assigned **random weights** initially.

Step 3: Adding Bias

- A bias value is added to each weighted input.

Step 4: Weighted Sum Calculation

$$\text{Weighted Sum} = (w_1x_1 + w_2x_2 + \ldots + b)$$

Step 5: Activation Function

- The weighted sum is passed through an activation function (ReLU, Sigmoid, etc.).
- Determines whether the neuron fires and what value it outputs.

Step 6: Forward Propagation

- The processed information moves from:
  Input Layer → Hidden Layer → Output Layer
- Gives the predicted output.

Step 7: Error Calculation

- Predicted output is compared with actual output.
- Error is calculated using the loss function.

Step 8: Backpropagation

- Error is sent backward through the network.
- Weights are adjusted to reduce the error.
- Uses gradient descent or other optimization algorithms.

Step 9: Repeating for Epochs

- The process of forward propagation + backpropagation is repeated multiple times (epochs).
- Gradually improves accuracy and reduces error.

Step 10: Final Prediction

- After training, final optimized weights are used to make accurate predictions.

BONUS QUESTIONS

1. Explain in brief types of ANN based on layers.
2. Write short note on Back propagation network.
3. Write short note on importance of Activation function used in Neural Network
4. What is Personalized recommendation? What is content based recommendation

Join Community by clicking below links

Website Link

For question papers and notes visit website

Click here

Telegram Channel

Click here

WhatsApp Channel

Click here