







**Ans: Classification problems.( Logestic regression used is when the dependent variable is categorical, especially binary classification like “yes/no”, “spam/not spam”, etc.)**













**Ans: d) The independent variables should be categorical**





**Ans: False**





**Ans: Log Loss or Binary Cross-Entropy**















**Ans: b) Sigmoid**





**Ans: L1 regularization(Lasso) : Adds the absolute values of the coefficients as a penalty term to the loss function.**

**It encourages sparsity, meaning it can reduce some coefficients exactly to zero, effectively performing feature selection.**

**L2 regularization (Ridge) : Adds the squared values of the coefficients as a penalty term. It tends to shrink coefficients smoothly towards zero but does not make them exactly zero, so it retains all features but reduces their impact.**















**Ans: b) Precision--Recall**





**Ans: The output of a Logistic Regression model before applying the sigmoid function is the linear combination of the input features and the model weights, also known as the logit or raw score.**

**Mathematically, it's given by:**

**z=wTx+bz = \mathbf{w}^T \mathbf{x} + bz=wTx+b**

**Where:**

* **w\mathbf{w}w = weight vector**
* **x\mathbf{x}x = input feature vector**
* **b bb = bias (intercept) term**

**This value zzz is then passed through the sigmoid function to get a probability output**







Ans: **1. Logistic Regression**

* **Purpose: Used for classification problems.**
* **Target Variable: Categorical, typically binary (e.g., 0 or 1, True or False, Yes or No).**
* **Output: A probability between 0 and 1, which is then typically converted into a class label.**
* **Equation:**

**P(Y=1∣X)=11+e−(β0+β1X1+…+βnXn)P(Y=1|X) = \frac{1}{1 + e^{-(\beta\_0 + \beta\_1X\_1 + \ldots + \beta\_nX\_n)}}P(Y=1∣X)=1+e−(β0​+β1​X1​+…+βn​Xn​)1​**

* **Use case: Spam detection, disease diagnosis (yes/no), etc.**

**2. Linear Target Variable (Linear Regression)**

* **Purpose: Used for regression problems.**
* **Target Variable: Continuous numeric values (e.g., price, temperature, weight).**
* **Output: A real number (any value on the number line).**
* **Equation:**

**Y=β0+β1X1+…+βnXn+εY = \beta\_0 + \beta\_1X\_1 + \ldots + \beta\_nX\_n + \varepsilonY=β0​+β1​X1​+…+βn​Xn​+ε**

* **Use case: Predicting house prices, sales amounts, etc.**













**Ans: b) Regularization(L1 or L2 penalty)**















**Ans: a) Supervised learning**





**Ans: False**













**Ans: b) Euclidean distance**





**Ans: Small k (e.g., k = 1, 3)**

* **Pros:**
  + **Captures local patterns well**
  + **Can fit the training data very closely**
* **Cons:**
  + **High variance: sensitive to noise or outliers**
  + **Overfitting risk: may model noise instead of general patterns**

**🔹 Large k (e.g., k = 15, 50)**

* **Pros:**
  + **Smoother decision boundary: averages over more neighbors**
  + **Less sensitive to noise: more robust to outliers**
* **Cons:**
  + **High bias: may oversimplify the model**
  + **Underfitting risk: might miss important local structures**













**Ans: Classification only**





**Ans: Step-by-Step Process:**

**1. Choose the number of neighbors (k)**

* **Decide how many neighbors to consider (e.g., k = 3 or k = 5).**

**2. Measure distance**

* **Compute the distance between the new data point and every point in the training set.**
* **Most common distance: Euclidean distance**

**d(x,y)=∑i=1n(xi−yi)2d(x, y) = \sqrt{\sum\_{i=1}^n (x\_i - y\_i)^2}d(x,y)=i=1∑n​(xi​−yi​)2​**

**3. Find the k nearest neighbors**

* **Sort the training points by distance.**
* **Select the k closest points (smallest distances).**

**4. Majority vote (Classification)**

* **Look at the labels of these k nearest neighbors.**
* **Vote: The class with the most occurrences among the neighbors is the predicted class.**

**5. Assign the class**

* **The new data point is assigned the majority class from its neighbors.**













**Ans: c) KNN is computationally expensive at prediction time**





**Ans: Pros of KNN**

* **Simple and intuitive**
* **No need for model training**
* **Naturally supports multi-class classification**
* **Can model complex decision boundaries**

**Cons of KNN**

* **Slow at prediction time (especially with large datasets)**
* **Requires careful feature scaling (sensitive to distances)**
* **Memory-intensive**
* **Poor performance with irrelevant/noisy features**

**Pros of Logistic Regression**

* **Fast and efficient during prediction**
* **Very interpretable (useful in fields like medicine, finance)**
* **Works well for linearly separable data**
* **Naturally outputs probabilities**

**Cons of Logistic Regression**

* **Assumes linear relationship between features and log-odds**
* **Poor performance with complex or non-linear patterns**
* **Sensitive to multicollinearity between features**





**Ans : What happens when k = 1?**

* **The model assigns the class of the single nearest training data point to the new input.**
* **It becomes a very flexible classifier, perfectly memorizing the training data.**

**Pros of k = 1:**

* **Zero training error: It will classify all training data correctly.**
* **Very responsive to local patterns in the data.**

**Why it’s not always a good idea:**

1. **Overfitting**
   * **k = 1 leads to very high variance.**
   * **Even small noise or outliers in the data can cause incorrect predictions.**
2. **Poor generalization**
   * **The model memorizes the training data instead of learning general trends.**
   * **New, slightly different examples may be misclassified.**
3. **Sensitive to noise**
   * **One mislabeled training point can lead to a wrong prediction for nearby test points.**















**Ans: a)Using PCA for dimensionality reduction**



|  |
| --- |
|  |