INSTITUTE NAME:BESANT TECHNOLOGIES

PROJECT NAME:MACHINE LEARNING INTERVIEW QUESTIONS

SUBMITTED TO:ML TRAINER

SUBMITTED BY:SHREE PUNITHA R

CO-ORDINATOR NAME:DURIKA M

DATE OF SUBMISSION:19/11/2025

1. Linear Regression

1. What are the assumptions of Linear Regression?

Linear regression assumes linearity between variables, independence of errors, homoscedasticity, and normally distributed residuals. It also assumes no multicollinearity among predictors. Violation of these assumptions reduces model reliability.

2. What is multicollinearity?

Multicollinearity occurs when independent variables are highly correlated with each other. This makes coefficient estimates unstable and increases variance. It is detected using VIF or correlation matrix.

3. What is the cost function in Linear Regression?

The cost function used is Mean Squared Error (MSE). It measures the average squared difference between actual and predicted values. Gradient descent tries to minimize this cost.

4. What is R^2 and Adjusted R^2?

R^2 measures how much variance in the target is explained by the model. Adjusted R^2 corrects for adding unnecessary features. It is more reliable when comparing models with different numbers of features.

5. What is overfitting in Linear Regression?

Overfitting happens when the model learns noise along with the pattern. It occurs in high-dimensional data. Regularization methods like L1 and L2 help reduce overfitting.

6. Difference between L1 and L2 regularization?

L1 (Lasso) performs feature selection by shrinking some weights to zero. L2 (Ridge) shrinks all weights but doesn’t make them zero. L1 works better when many features are irrelevant.

7. What is gradient descent?

Gradient descent is an optimization algorithm to minimize the cost function. It updates weights in the direction of steepest descent. Learning rate controls the step size.

8. What are residuals?

Residuals are the difference between actual and predicted values. They help evaluate model performance. A random residual pattern indicates a good model.

9. Why do outliers affect Linear Regression?

Linear regression uses squared error, so outliers cause large penalties. This shifts the regression line unnecessarily. Robust regression methods can reduce impact.

10. Why is feature scaling not required?

Linear regression coefficients adjust automatically to feature scale. But if using gradient descent, scaling improves convergence. Regularized regression also works better with scaled features.

2.Logistic Regression

1. Why use Logistic Regression for classification?

Logistic regression maps outputs to probabilities using the sigmoid function. This makes it suitable for binary classification. It also provides interpretable coefficients.

2. What is the sigmoid function?

The sigmoid maps values between 0 and 1. It outputs probability of belonging to a class. The decision threshold is typically 0.5.

3. What is log loss?

Log loss penalizes incorrect high-confidence predictions heavily. It measures the uncertainty of probability estimates. Lower log loss indicates better performance.

4. What is Maximum Likelihood Estimation (MLE)?

MLE finds parameters that maximize probability of observing given data. It is used instead of least squares. It fits the best decision boundary for classification.

5. What is the decision boundary?

It is the line or surface dividing classes. Derived from model coefficients. For logistic regression, it is linear unless polynomial features are used.

6. How to handle imbalanced classes?

Use class weights, resampling, SMOTE, or adjusting the threshold. Metrics like F1-score and ROC-AUC are more reliable. Logistic regression is sensitive to class imbalance.

7. Why not use Linear Regression for classification?

Linear regression gives unbounded outputs beyond 0–1. It also assumes Gaussian errors, unsuitable for classification. Logistic regression models probability directly.

8. What are odds and odds ratio?

Odds describe likelihood of event occurring vs. not occurring. Logistic coefficients represent log-odds. Exponentiating coefficients gives odds ratios.

9. What is regularization in Logistic Regression?

Regularization prevents overfitting by penalizing large weights. L1 makes the model sparse while L2 distributes penalty across features. It helps with high-dimensional data.

10. Difference between Logistic and Linear Regression?

Linear regression predicts continuous values; logistic predicts probabilities. Logistic uses sigmoid and log loss. Linear regression uses MSE and assumes normality.

3. Decision Tree

1. How does a decision tree split nodes?

It uses metrics like Gini impurity, entropy, or MSE. The best split maximizes information gain. Each node divides data into pure subsets.

2. What is entropy?

Entropy measures randomness or impurity in the data. Higher entropy means more disorder. Decision trees aim to reduce entropy with splits.

3. What is information gain?

Information gain is reduction in impurity after splitting. Higher gain means better feature for splitting. It guides the structure of tree.

4. Why do trees overfit?

Trees keep splitting until data is perfectly classified. This leads to capturing noise. Pruning helps reduce overfitting.

5. What is pruning?

Pruning removes unnecessary branches. It improves generalization by simplifying the model. Can be done pre- or post-training.

6. Can decision trees handle missing values?

Yes, using surrogate splits or ignoring missing values during calculation. Some libraries automatically handle missing data. Trees are flexible with mixed data types.

7. What is Gini impurity?

Gini measures the probability of incorrectly classifying a random sample. Lower Gini indicates purer nodes. It is computationally faster than entropy.

8. Advantage of decision trees?

They are easy to visualize and interpret. Require no feature scaling or normalization. Work well with both numerical and categorical data.

9. What is max_depth?

It limits the height of the tree. Prevents model from overfitting. Often tuned using cross-validation.

10. Difference between CART and ID3?

CART uses Gini and supports regression. ID3 uses entropy and supports only classification. CART trees are binary; ID3 can produce multiway splits.

4. Random Forest

1. How does Random Forest work?

Random Forest builds multiple decision trees using bootstrapped samples and random subsets of features. Each tree gives a prediction, and the final output is voted or averaged. This ensemble approach reduces variance and improves stability.

2. What is bagging?

Bagging (Bootstrap Aggregation) trains multiple models on different randomly sampled datasets. Each dataset is formed using sampling with replacement. The results are averaged to reduce variance.

3. What is Out-of-Bag (OOB) error?

OOB samples are data points not included in a tree’s bootstrapped sample. OOB error is computed using these samples, making it a built-in validation method. It helps estimate model performance without separate validation data.

4. Why does Random Forest reduce overfitting?

It averages predictions from many uncorrelated trees. This reduces the chance that noise in one tree affects the final output. Ensemble diversity makes the model robust.

5. What is feature randomness?

At each split, only a random subset of features is considered. This ensures trees are less correlated. It improves generalization and prevents overfitting.

6. What is feature importance?

It measures how much each feature contributes to reducing impurity across all trees. Features used in important splits receive higher scores. Helps in model interpretation and feature selection.

7. Can Random Forest handle missing data?

Yes, by using surrogate splits or ignoring missing values in impurity calculations. Some implementations like XGBoost also handle missing data internally. This makes RF robust to incomplete datasets.

8. What are the limitations of Random Forest?

It can be slow on large datasets with many trees. It requires more memory compared to a single tree. Interpretability is lower than simple models.

9. What is the role of number of trees?

More trees generally improve accuracy but increase computation. After a point, performance gain becomes minimal. Usually 100–500 trees are sufficient.

10. Difference between Random Forest and Gradient Boosting?

RF uses bagging and builds trees independently in parallel. Boosting builds trees sequentially to correct previous errors. Boosting reduces bias while RF reduces variance.

5. Support Vector Machine (SVM)

1. What is the main objective of SVM?

SVM aims to find the hyperplane that maximizes the margin between classes. The points closest to this boundary are support vectors. A larger margin improves generalization.

2. What are support vectors?

Support vectors are data points closest to the separating hyperplane. They determine the margin width. Removing them changes the model significantly.

3. What is the kernel trick?

Kernel trick maps data to a higher-dimensional space without explicit computation. This allows SVM to solve non-linear problems efficiently. Common kernels include RBF, polynomial, and sigmoid.

4. What is the role of C parameter?

C controls trade-off between margin maximization and classification error. High C tries to classify all points correctly (low bias). Low C allows errors but increases margin (high bias).

5. What is gamma in RBF kernel?

Gamma defines how far the influence of a single training example extends. High gamma → tight boundaries → risk of overfitting. Low gamma → smoother boundaries.

6. Hard margin vs soft margin SVM?

Hard margin does not allow misclassification, suitable for noise-free data. Soft margin allows some misclassifications to improve generalization. Soft margin is used in real-world noisy datasets.

7. What are the advantages of SVM?

Works well in high-dimensional spaces and complex boundaries. Effective when number of features is large. Robust to overfitting if parameters are well tuned.

8. What are limitations of SVM?

Slow for large datasets; training complexity is high. Sensitive to parameter tuning and feature scaling. Does not work well when classes overlap significantly.

9. Can SVM handle multi-class classification?

Yes, using one-vs-one or one-vs-rest strategies. Libraries like sklearn automatically implement this. SVM is inherently binary but can be extended.

10. Why does SVM need feature scaling?

SVM relies on distance-based calculations. Features with larger magnitudes can dominate others. Scaling ensures balanced influence across features.

<u>6. K-Nearest Neighbors (KNN)</u>

1. How does KNN work?

KNN classifies a point based on majority class among its k nearest neighbors. Distance metrics determine closeness. It is simple and non-parametric.

2. Why is KNN called a lazy learner?

KNN has no training phase; it stores all data. Prediction is slow because distances are computed at prediction time. Useful for small datasets.

3. How do you choose K?

Small K → noisy, overfits. Large K → smooth boundary, may underfit. Cross-validation helps find optimal K.

4. Why is feature scaling important in KNN?

KNN relies on distance metrics. Unscaled features distort distances and harm accuracy. Normalize or standardize data before KNN.

5. What is the curse of dimensionality?

As dimensions increase, data becomes sparse. Distances become less meaningful. This reduces KNN performance significantly.

6. Which distance metrics are used?

Euclidean, Manhattan, Minkowski for numeric data. Hamming distance for categorical data. Choice depends on data type.

7. Can KNN be used for regression?

Yes, by averaging the values of nearest neighbors. Works well for smooth datasets. But sensitive to outliers.

8. What are weighted KNN methods?

Closer neighbors get higher weights. Helps improve accuracy when distant neighbors are less relevant. Reduces bias from uneven densities.

9. What are the drawbacks of KNN?

Prediction is slow for large datasets. Sensitive to noise and irrelevant features. Memory requirements are high.

10. How to speed up KNN?

Use KD-trees, ball trees, or approximate nearest neighbor algorithms. Dimensionality reduction (PCA) also helps. Helps scale KNN to larger datasets.

7. Naive Bayes

1. What is Naive Bayes classifier?

Naive Bayes applies Bayes' theorem with the assumption of feature independence. It calculates the probability of each class given the input. Works well for high-dimensional data.

2. Why is it called "naive"?

It assumes features are independent given the class. This is rarely true but works surprisingly well. Simplifies computation drastically.

3. What types of Naive Bayes exist?

Gaussian NB for continuous data. Multinomial NB for word counts. Bernoulli NB for binary features.

4. What is Laplace smoothing?

Adds +1 to all counts to avoid zero probability. Helps when a feature-class combination does not appear in training data. Improves model robustness.

5. When does Naive Bayes perform poorly?

When features are highly correlated. The independence assumption fails. Also struggles with complex decision boundaries.

6. Why NB is used in text classification?

Text features (word frequencies) are mostly independent. NB is fast and requires small training data. Performs well in spam detection and sentiment analysis.

7. How does NB handle continuous features?

Gaussian NB assumes features follow a normal distribution. Probability is computed using Gaussian density function. Works well even if distribution is not perfectly normal.

8. What is the posterior probability?

It is the probability of a class given the observed data. Calculated using Bayes' theorem. NB picks the class with highest posterior.

9. What is conditional independence?

Features are assumed independent when class label is known. This simplifies joint probability calculation. It's the core assumption in Naive Bayes.

10. Advantages of Naive Bayes?

Fast training, low memory, works well with high-dimensional data. Handles text and categorical data efficiently. Robust even with limited training samples.

8. K-Means Clustering

1. What is K-Means clustering?

K-Means partitions data into K clusters based on similarity. It assigns each point to the nearest centroid. Centroids update iteratively until convergence.

2. How to choose the value of K?

Use the elbow method, silhouette score, or gap statistic. These methods measure how well clusters are separated. K should balance compactness and separation.

3. What is the K-Means algorithm?

Initialize centroids → assign points to nearest centroid → update centroids → repeat. Process continues until centroids stop moving. It optimizes cluster compactness.

4. Limitations of K-Means?

Sensitive to initialization and outliers. Works poorly with non-spherical or uneven-sized clusters. Requires specifying K beforehand.

5. What is K-Means++?

Improved initialization technique. Selects initial centroids far apart to reduce randomness. Leads to faster convergence and better clusters.

6. Why is scaling necessary?

Distance metrics are affected by feature magnitudes. Scaling ensures fair contribution of each feature. Without scaling, clusters become biased.

7. What is within-cluster sum of squares (WCSS)?

WCSS measures compactness of each cluster. K-Means tries to minimize total WCSS. Lower WCSS means tighter clusters.

8. Can K-Means work with categorical data?

No, because it relies on numerical distance metrics. K-Modes or K-Prototypes are used instead. They handle categorical or mixed data.

9. What is convergence in K-Means?

Occurs when centroids no longer change significantly. Means algorithm has stabilized. Typically achieved in few iterations.

10. When does K-Means fail?

With varying cluster densities, non-convex shapes, or outliers. It assumes spherical clusters. Choosing wrong K also degrades performance.

9.<u>DBSCAN (Density-Based Spatial Clustering of Applications with Noise)</u>

1. What is DBSCAN?

DBSCAN is a density-based clustering algorithm that groups points closely packed together.
It identifies clusters based on the density of points and labels low-density points as noise.
It does not require specifying the number of clusters beforehand.

2. What are the main parameters in DBSCAN?

DBSCAN uses two parameters: **eps (ε)**, the neighborhood radius, and **minPts**, the minimum number of points required to form a dense region.
These two together determine how clusters and noise are identified.
Choosing the right values is crucial for good results.

3. What are core points, border points, and noise points?

A **core point** has at least minPts within eps distance.
A **border point** has fewer than minPts neighbors but lies within the neighborhood of a core point.
A **noise point** does not satisfy either condition and is considered an outlier.

4. How does DBSCAN differ from K-Means?

DBSCAN does not require the number of clusters (K) and can find arbitrarily shaped clusters.
It handles noise naturally, unlike K-Means which forces all points into clusters.
It also performs better on datasets with varying densities.

5. What are the advantages of DBSCAN?

DBSCAN detects clusters of any shape, unlike K-Means which works best with spherical clusters.
It identifies noise/outliers explicitly, improving cluster quality.
It is effective for spatial and geographical datasets.

6. What are the limitations of DBSCAN?

DBSCAN struggles when clusters have varying densities because a single eps value might not work.
It can be computationally expensive for high-dimensional data.
Choosing eps and minPts correctly can be challenging.

7. How do you choose eps (ε) in DBSCAN?

A common method is using the **k-distance graph** (usually k = minPts).
You plot the distances and look for the "elbow" point, which indicates a suitable eps.
This helps ensure meaningful dense regions are detected.

8. Why is DBSCAN good at detecting outliers?

Points that do not belong to any dense region are automatically labeled as noise.
It doesn't force all points into clusters, unlike centroid-based methods.
Thus, outlier detection comes as a natural result of the algorithm.

9. Can DBSCAN handle high-dimensional data? Why or why not?

DBSCAN struggles in high dimensions due to the **curse of dimensionality**, where distance metrics lose meaning.
Density becomes hard to measure as points appear uniformly distant.
Algorithms like HDBSCAN or dimensionality reduction methods are preferred first.

10. What is HDBSCAN and how is it related?

HDBSCAN (Hierarchical DBSCAN) extends DBSCAN to handle varying densities.
It builds a hierarchy of clusters and extracts stable clusters from it.
It eliminates the need for choosing eps manually and improves robustness.

10. Gradient Boosting / XGBoost

1. What is boosting?

Boosting builds models sequentially, each correcting errors of the previous one. Weak learners are combined into a strong learner. It reduces bias and improves accuracy.

2. How does Gradient Boosting work?

Each new tree fits the residual errors of the previous model. The final prediction is the sum of all trees scaled by learning rate. It iteratively improves performance.

3. Why is learning rate important?

Learning rate controls contribution of each tree. Small rate improves accuracy but requires more trees. Large rate risks overfitting.

4. What is the role of tree depth?

Depth controls complexity of each tree. Shallow trees reduce overfitting and make boosting stable. Deep trees may memorize noise.

5. What is regularization in XGBoost?

XGBoost uses L1 & L2 penalties on leaf weights. This reduces overfitting and improves generalization. Its regularization is stronger than other boosting methods.

6. What is early stopping?

Training stops when validation loss no longer improves. Prevents overfitting and saves time. Common in boosting and neural networks.

7. Why is XGBoost fast?

Uses parallel tree building, optimized memory, and pruning. Also supports distributed computing. These features give significant speed advantages.

8. What is shrinkage?

Shrinkage scales each tree’s output by learning rate. Slows down boosting to improve accuracy. Prevents overfitting.

9. What is feature importance in XGBoost?

Calculated from gain, cover, or frequency of splits. Shows how much each feature contributes to model accuracy. Useful for feature selection.

10. Limitations of Gradient Boosting?

Sensitive to hyperparameters and noise. Can overfit without proper tuning. Training is slower than bagging methods like Random Forest.