MCQs

1. Using a goodness of fit,we can assess whether a set of obtained frequencies differ from a set of frequencies.

a) Mean

b) Actual

c) Predicted

d) Expected

Answer- d) Expected

2. Chisquare is used to analyse

a) Score

b) Rank

c) Frequencies

d) All of these

Answer - c) Frequencies

3. What is the mean of a Chi Square distribution with 6 degrees of freedom?

a) 4

b) 12

c) 6

d) 8

Answer- C) 6

4. Which of these distributions is used for a goodness of fit testing?

a) Normal distribution

b) Chisqared distribution

c) Gamma distribution

d) Poission distribution

Answer- b) Chi-square distribution

5. Which of the following distributions is Continuous

a) Binomial Distribution

b) Hypergeometric Distribution

c) F Distribution

d) Poisson Distribution

Answer - c) F Distribution

6. A statement made about a population for testing purpose is called?

a) Statistic

b) Hypothesis

c) Level of Significance

d) TestStatistic

Answer - b) Hypothesis

7. If the assumed hypothesis is tested for rejection considering it to be true is called?

a) Null Hypothesis

b) Statistical Hypothesis

c) Simple Hypothesis

d) Composite Hypothesis

Answer - a) Null Hypothesis

8. If the Critical region is evenly distributed then the test is referred as?

a) Two tailed

b) One tailed

c) Three tailed

d) Zero tailed

Answer- a) Two tailed

9. Alternative Hypothesis is also called as?

a) Composite hypothesis

b) Research Hypothesis

c) Simple Hypothesis

d) Null Hypothesis

Answer - b) Research Hypothesis

10. In a Binomial Distribution, if ‘n’ is the number of trials and ‘p’ is the probability of success, then the mean value is given by

a) np

b) n

Answer- a) np

Subjective

1. **R-squared or Residual Sum of Squares (RSS) which one of these two is a better measure of goodness of fit model in regression and why?**

   Answer-

   In regression analysis, R-squared (coefficient of determination) is typically thought to be a superior indicator of goodness of fit than residual sum of squares (RSS). Here's why.

   Interpretability: R-squared calculates the proportion of variation in the dependent variable that is predictable from the independent variables. It runs from 0 to 1, with 1 representing a perfect match. This allows for easy interpretation and comparison among models. RSS, on the other hand, is merely the sum of squared residuals and does not serve as a direct measure of fit quality on its own.

   Normalised Measure: R-squared is normalised to the total variation in the dependent variable, providing a relative measure of the regression model's ability to explain data variance. This normalisation enables comparison of different datasets and models. RSS, which is an absolute measure, does not have this advantage.

   Use in Model Comparison:R-squared is frequently used to compare different models. A higher R-squared value indicates that the model has stronger explanatory power. It is useful in determining whether extra variables provide explanatory value to the model. RSS alone cannot permit this comparison without extra normalisation or scaling.

   Direct Relationship to Fit:R-squared represents the proportion of variance explained by the model. As R-squared increases, the model's predictions approach the actual values, indicating a better fit. RSS, while useful in computations such as the F-statistic for significance testing, does not provide as much information about the model's overall fit.

   In conclusion, R-squared is chosen over RSS as a measure of goodness of fit in regression because it gives a normalised, interpretable measure of how well the model explains the variation in the dependant variable. It enables easy comparison of models and datasets, and its value immediately displays the fraction of variability explained by the model's predictors.

2. **What are TSS (Total Sum of Squares), ESS (Explained Sum of Squares) and RSS (Residual Sum of Squares) in regression. Also mention the equation relating these three metrics with each other.**

Answer - In regression analysis, TSS (Total Sum of Squares), ESS (Explained Sum of Squares), and RSS (Residual Sum of Squares) are important terms for understanding the variability in the dependent variable (Y) and the regression model's contributions.

1) Total Sum of Squares (TSS):

Definition: TSS measures the total variability in the response variable Y around its mean Y^-.

Formula: $TSS=\sum_{i=1}^{n} (Y_i - \bar{Y})^2$

where Y_i are the observed values of the response variable Y, $\bar{Y}$ is the mean of Y, and n is the number of observations.

2. The Residual Sum of Squares (RSS):

Definition: RSS measures the variability in the response variable that is not explained by the regression model, i.e., the residual variation.

Formula: $RSS=\sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$

where Y_i are the observed values of Y and $\hat{Y}_i$ are the predicted values from the regression model.

3. Explained Sum of Squares (ESS):

Definition: ESS quantifies the variability in the response variable that is explained by the regression model.

Formula: $ESS=\sum_{i=1}^{n} (\hat{Y}_i - \bar{Y})^2$

where $\hat{Y}_i$ are the predicted values of Y from the regression model.

In summary, TSS, ESS, and RSS are important metrics in regression analysis because they assist quantify total variance, variance explained by the model, and residual variance, respectively. Understanding these measures is critical for determining the goodness of fit and predictive power of regression models.

3. What is the need of regularization in machine learning?

Answer- Regularisation in machine learning is required for numerous reasons:

1. Preventing Overfitting: One of the key goals of regularisation is to avoid overfitting. Overfitting happens when a model learns the noise and minutiae of the training data too well, resulting in poor generalisation to new, untested data. Regularisation strategies penalise the model's objective function, discouraging it from overfitting the training data and enhancing its capacity to generalise.

2. Handling Collinearity: In datasets with strongly linked predictor variables (multi collinearity), ordinary least squares (OLS) regression can yield unstable and incorrect coefficient estimates. Regularisation approaches such as Ridge Regression (L2 regularisation) and Elastic Net (a combination of L1 and L2 regularisation) can reduce multi collinearity by decreasing the coefficients of correlated variables.

3. Feature Selection: Regularisation approaches, such as L1 regularisation (Lasso), have the property of reducing the coefficients of less significant features to zero. This results in automatic feature selection, in which the model only considers the most relevant predictors. This simplifies the model and enhances its interpretability.

4. Controlling Model Complexity: Regularisation allows us to reduce a model's complexity by penalising coefficient size. Models with big coefficients are more complex and prone to overfitting. Regularisation promotes simpler models that generalise better to fresh data by penalising these coefficients (by L1 or L2 regularisation).

5. Improving Model Stability: Regularisation approaches can boost the numerical stability of the optimisation process during model training. They can assist keep the model parameters from growing too high, which could lead to numerical overflow or instability in the calculations.

6. Improving Model Performance: Regularisation strategies typically result in higher model performance metrics on fresh, previously unknown data. Regularised models, which reduce

overfitting and focus on relevant features, often have fewer prediction errors and higher accuracy than non-regularized models.

In conclusion, regularisation is critical in machine learning for improving model generalisation, handling multi collinearity, performing feature selection, controlling model complexity, improving stability during training, and eventually improving overall model performance on unseen data. It is an essential tool for developing strong and successful machine learning models across multiple domains and applications.

3. **What is Gini–impurity index?**
   Answer - "The Gini impurity index is a metric used in decision tree algorithms and ensemble methods such as random forests to assess the impurity or homogeneity of a collection of instances based on their class labels.

   Definition:

   The Gini impurity of a set of instances is computed by summing the probability \(p(i) \) of an example being classified to each class \(i \) and the chance \(1 - p(i) \) of a random sample being mistakenly classified.

   Formula:

   For a node t that contains Nt_samples, the Gini impurity G(t) is calculated as:

   $G(t)=1-\sum_{i=1}^{c} p(i|t)^2$

   where: -
   C is the number of classes.
   p(i|t) is the proportion of samples in node t that belong to class i.

   Interpretation:

   - Gini impurity runs from 0 to 0.5 (or 0 to 1 depending on the formulation), with 0 indicating that the set is totally pure (all samples belong to the same class) and higher values indicating greater impurity (more mixed class labels in the set).

   - A Gini impurity of zero indicates that all samples in the set belong to the same class, meaning absolute purity.

   - A Gini impurity of 0.5 (or 0.25, depending on the formulation) indicates that the set is uniformly distributed across all classes, signifying the highest impurity.

Usage in decision trees:

In decision tree methods, the Gini impurity index is utilised at each node to identify the optimum feature-threshold split. The split that produces the lowest Gini impurity after the split is chosen as the optimal split. This process continues recursively until the tree is fully grown or a stopping criterion is met.
Advantages:

- Easy to grasp and computationally efficient. - Effective for decision tree and ensemble approaches, such as random forests.

- Encourages splits, resulting in nodes with more uniform class distributions.

Limitations:

- May be biased towards selecting qualities with a large number of classes.
- Sensitive to class imbalance (where one class outnumbers others).
- Other impurity measurements, such as entropy (information gain), may be favoured in some situations.

In conclusion, the Gini impurity index is a measure of impurity or uncertainty used in decision tree algorithms to aid the process of picking splits that result in more homogeneous groups of data based on class labels."

4. **Are regularized decision-trees prone to overfitting? If yes, why?**
   Answer –
   "Yes, regularized decision trees are prone to overfitting, which arises for a variety of reasons:

   1. High Variance: Decision trees can learn exceedingly complicated decision boundaries that completely divide the training data. A decision tree can grow unconstrained (regularised) until it perfectly classifies every example in the training set. This results in a model that over fits the training data, collecting both noise and outliers as well as underlying trends. As a result, such a model may have difficulty generalising to new, previously unknown data.

   2. Capturing Noise: Decision trees can divide based on any attribute at any level of the tree, allowing them to capture noise in training data. This noise can include random oscillations or outliers that do not reflect the genuine underlying relationships in the data. Regularized trees are not penalised for splitting to collect such noise, resulting in overfitting.

   3. Complexity: Un regularized decision trees can become quite deep and complicated, particularly when there are several features or when the tree is allowed to grow until it perfectly classifies all training samples. A deep tree with multiple splits can simulate complex interactions between variables, which may be unique to the training data and will not generalise to new data.

4. Sensitive to Small alterations: Because decision trees can divide based on any characteristic at any time, they are extremely sensitive to minor alterations in training data. A minor change in the training data could result in an entirely different tree topology if no constraints are applied. This sensitivity may worsen overfitting.

5. No Smoothing method: Unlike some other machine learning models (such as linear regression with regularisation), decision trees lack a built-in method for smoothing or simplifying the model. Regularisation strategies like as pruning (removing nodes from the tree that do not provide significant improvement) and restricting the tree's maximum depth aid in complexity control and overfitting reduction.

To address these difficulties and avoid overfitting, numerous regularisation approaches can be applied to decision trees:
- Pruning: Post-pruning or cost-complexity pruning entails developing the entire tree and then eliminating nodes that do not significantly improve the model's performance metrics.
- Minimal Samples per Leaf: Specifying the minimal number of samples necessary at a leaf node can prevent the model from doing overly specific splits that collect noise.
- Maximum Depth: Limiting the tree's depth allows you to control its complexity and keep it from getting too deep.
- Minimum Impurity Decrease: Requiring a minimum improvement in impurity (such as Gini impurity or entropy) before splitting helps to avoid constructing splits that do not significantly improve the model's performance.
To summarise, un regularized decision trees can over fit training data because of their capacity to collect noise and absence of model complexity limitations. Regularisation techniques are critical for strengthening decision trees' generalisation capacity and making them more resilient to data fluctuations."

6. What is an ensemble technique in machine learning?

Answer –

"An ensemble methodology in machine learning is a strategy that combines numerous individual models (also known as base models or weak learners) to create a more powerful predictive model. The goal behind ensemble methods is to use the collective wisdom of several models to improve overall performance when compared to any one model in the ensemble.

Key concepts:

1. Base Models (Weak Learners)

- Ensemble approaches often begin with independent base models trained on the same dataset. These foundation models might be homogeneous (e.g., decision trees) or heterogeneous (e.g., combining decision trees and neural networks).

2. Aggregation Strategy: - Once the foundation models have been trained, ensemble methods aggregate their predictions to provide a final forecast. The manner in which predictions are pooled depends on the specific ensemble method:

- Voting: In classification tasks, predictions from many models are integrated using either majority voting (hard voting) or averaging projected probabilities (soft voting).

- Averaging: In regression tasks, predictions are usually averaged across models to get the final prediction.

- Weighted Averaging: Some ensemble approaches weight forecasts according to each base model's confidence or performance.

3. Types of Ensemble Methods:

- Bagging (Bootstrap Aggregating): This method includes training numerous instances of the same base model on different subsets of the training data (sampled using replacement) and then averaging the results. Bagging is exemplified by Random Forests, which train decision trees using bootstrapped data sets.

- Boosting: Boosting methods train base models in a sequential manner, with each successive model focusing on improving the predecessor's performance by paying greater attention to previously misclassified samples. Examples include AdaBoost (Adaptive Boosting), Gradient Boosting Machines (GBM), and XGBoost.

- Stacking: Stacking (also known as Stacked Generalisation) combines multiple base models by training a meta-model to determine the optimum way to integrate the base models' predictions. The meta-model is trained using the predictions (meta-features) produced by the base models on a validation set.

4. Advantages:

- Improved Performance: By lowering variance, bias, or both, ensemble approaches frequently outperform individual models in terms of accuracy or prediction.

- Robustness: Because ensembles incorporate several models' perspectives, they are more resilient to noisy data and outliers.

- Versatility: They can be used for a variety of machine learning tasks (classification, regression, and clustering), as well as diverse base models.

5. Considerations: - Computational Cost: Ensemble approaches may be more computationally expensive than single models, especially for large datasets or complicated base models.

- Interpretability: Some ensemble methods, including Random Forests, provide insights into feature importance, but others, such as stacking with complex models, may compromise interpretability.

In conclusion, ensemble approaches use the capability of several models to improve predicted accuracy and robustness in machine learning tasks. They are widely employed in a variety of fields and have become an essential component of modern machine learning procedures."

7. What is the difference between Bagging and Boosting techniques?

Answer -

Bagging (Bootstrap Aggregating) and Boosting are two ensemble machine learning approaches that improve the performance of individual models. However, they vary in their method to training and combining basic models:

Bagging (Bootstrap Aggregation):

1. Approach: - Bagging trains multiple instances of the same underlying model (e.g., decision trees) on distinct subsets of training data.
- Each subset (known as a bootstrap sample) is drawn at random from the original training data, with replacement.

2. Training Process: - Base models are trained in parallel, independent of each other.
- Random sample with replacement causes each model to see a slightly different subset of the data.
- The models have no dependencies; they are learned and predicted independently.

3. Combining Predictions: In bagging, predictions from all base models are often averaged (for regression) or majority-voted (for classification) to provide the final prediction.
- Bagging reduces variance by averaging predictions from many models trained on distinct subsets of data, resulting in better overall generalisation.

4. Examples - **Random Forest**: A bagging-based ensemble method that trains many decision trees on diverse bootstrap samples of data and averages their predictions.

Boosting:

1. Approach: - Boosting includes successively training numerous base models (e.g., weak learners like shallow decision trees), with each succeeding model correcting the faults of its predecessor.
- Models are trained sequentially, with each model focusing on cases predicted erroneously by preceding models.

2. Training Process: Base models are trained iteratively. Initially, all data points have equal weight.
- The weights of erroneously predicted instances rise with each iteration, causing subsequent models

to pay more attention to these instances.

3. Combining Predictions: Predictions from all base models are combined using a weighted sum or voting mechanism, with better performing models receiving larger weights.
- Boosting tries to eliminate bias by focusing sequentially on the most difficult-to-classify samples, hence increasing the model's prediction performance.

4. instances: - AdaBoost (Adaptive Boosting): A traditional boosting technique where each weak learner is trained sequentially, and each subsequent model focuses on instances that were misclassified by earlier models.
- Gradient Boosting Machines (GBM): Another common boosting technique that develops models sequentially by optimising a loss function in a gradient descent-like way to reduce prediction errors.

Differences Summarised:

- Training: Bagging trains base models independently and concurrently, whereas Boosting trains models sequentially, with each succeeding model learning from the errors of its predecessor.

- Dependency: Bagging models are independent of one another, but Boosting models are dependent because each successive model improves on the performance of the prior one.
- Focus: Bagging reduces variance by aggregating predictions from numerous models, whereas boosting reduces bias and improves performance by focusing on difficult-to-classify instances.

8. What is out-of-bag error in random forests?

Answer - The out-of-bag (OOB) error in Random Forests is a method for estimating model performance without requiring a separate validation set or cross-validation. This is how it works.

Random Forest is an ensemble learning method that mixes decision trees trained on several subsets of data. Each tree in the Random Forest is trained on a bootstrap sample of the original dataset (sampling with replacement), which implies that some data points may be repeated while others are excluded.

Out-of-Bag Error Calculation: - Approximately one-third of the data points are excluded from each individual decision tree in the Random Forest during training.
- These missing data points are known as out-of-bag (OOB) samples for that specific tree.
- The OOB samples serve as a validation set for each tree in the forest because they were not utilised in training that tree.

Steps for Assessing OOB Error:
1. Prediction for OOB Samples: - The OOB samples from each tree in the Random Forest are utilised to produce predictions based on the tree's learned rules.

2. Aggregation of Predictions: - Because each OOB sample was only omitted from one tree's training set, it is utilised to evaluate its predictions.
- Predictions for OOB data from all trees in the forest are combined (averaged for regression or majority-voted for classification).

3. Compute OOB Error: - The OOB error is calculated by averaging the prediction error (e.g., mean squared error for regression or misclassification rate for classification) over all trees in the forest.

Advantages of OOB Error: - No Need for Cross-Validation: OOB error gives an impartial measure of Random Forest's performance without requiring a separate validation set or cross-validation.
- Efficient: Every data point is used for training or validation (OOB estimate) across the ensemble of trees, allowing it to make the most of the data.

Usefulness: - OOB error can aid in model selection (e.g., hyper parameter tuning) and evaluating the generalisation performance of the Random Forest model. - It provides a simple approach to determine how well the model will perform on unknown data.

In summary, the out-of-bag error in Random Forests is an estimate of prediction error generated by assessing the model on data points that were not included in the training of each individual tree in the ensemble. It is an effective and efficient method for assessing model performance that does not require additional validation sets or cross-validation methods.

9. What is K-fold cross-validation?

Answer- K-fold cross-validation is a machine learning approach used to evaluate the performance of a prediction model. It is a resampling process that divides the original dataset into K equal-sized folds (or subsets). The model is trained and assessed K times, with each fold serving as a validation set and the remaining K-1 folds as the training set.

Steps for K-fold Cross-validation:

1. Partitioning the Dataset: - The original dataset is randomly divided into K equal-sized subgroups (folds). Each subset is referred to as a fold, indicated by \(D_1, D_2,..., D_K).

2) Iterative Training and Validation:
- For each iteration (i = 1, 2,..., K):
- Set fold \(D_i \) as the validation (or test) set.
- Add the remaining \(K-1 \) folds (D_1, D_2,..., D_{i-1}, D_{i+1},..., D_K) to the training set.
- Train the model on the training data.
- Evaluate the model on the validation set (calculate performance metrics like accuracy, error rate, etc.).
3. Performance Estimation: - After K iterations, K separate performance estimates are obtained (one for each fold used as validation).
- The final performance estimate is usually the average of the K performance measures gathered throughout each iteration.

Benefits of K-fold Cross-validation:

More trustworthy Performance Estimate: K-fold cross-validation, which uses several data splits, provides a more trustworthy estimate of model performance than a simple train/test split.
Maximises Data Usage: Every data point is used exactly once in both training and validation, making the best use of the available data.
Decreases Variance: By averaging K performance estimates, K-fold cross-validation reduces the variance of the estimated performance metric when compared to a single train/test split.

Selecting the Value of K:

Typical Values: K is commonly set to 5 or 10, however the value might vary depending on the size of the dataset and available computational capabilities.
Trade-Offs: Larger K values incur a larger computational cost (since the model is trained and evaluated K times), but provide a more accurate estimate of model performance.

Variants of K-fold Cross-validation:

Stratified K-fold Cross-Validation: Ensures that each fold maintains the percentage of samples for each class, which is useful for imbalanced datasets.

Repeated K-fold Cross-Validation: This method involves repeating the K-fold process with different random data splits, which is important for producing more robust performance estimations.
In conclusion, K-fold cross-validation is a popular machine learning technique for assessing model performance by systematically splitting data into training and validation sets. It provides a more robust assessment of model performance than a conventional train/test split, particularly when the dataset is limited in size or the performance estimate must be accurate.

10. What is hyper parameter tuning in machine learning and why it is done?

Answer-

Hyper parameter tuning in machine learning is the process of determining the best hyper parameters for a model. Hyper parameters are parameters that are defined before the learning process begins and control various parts of the learning algorithm. They are not learned directly from the data, but rather are determined by prior understanding of the problem, heuristics, or experiments.

The Importance of hyper parameter Tuning

1. Optimising Model Performance: Hyper parameters have a major impact on the performance of machine learning models. Depending on the task, selecting the appropriate settings can result in increased accuracy, precision, recall, or other performance indicators.

2. Generalisation: Well-tuned hyper parameters can reduce overfitting and under fitting, resulting in models that are more generalizable to new, previously unknown data. Overfitting happens when a model does well on training data but badly on test data, whereas under fitting happens when a model is too simplistic to capture the underlying patterns in the data.

3. Algorithm Behaviour: Hyper parameters can influence the behaviour and convergence of learning algorithms. In gradient descent, for example, the learning rate influences how quickly and consistently the model learns from the data.

Common Hyper parameters:

Learning Rate: Controls the step size used in optimisation strategies such as gradient descent.
The **Number of Hidden Layers and Units in Neural Networks** determines the neural network's complexity and capability.
Regularisation Parameters: Determine the level of regularisation used to penalise big model weights and prevent overfitting (for example, lambda in Ridge Regression).
Kernel Parameters in Support Vector Machines (SVMs) include the kernel type and kernel coefficients.
Tree-Specific Parameters in Decision Trees and Random Forests: Examples include a tree's maximum depth, minimum sample size per leaf, and the number of trees in a forest.

Techniques for hyper parameter tuning:

1. Manual Search: Choose hyper parameters based on domain expertise and intuition.

2. Grid Search: Systematically evaluate a variety of hyper parameter settings using a grid of values, then choose the optimum combination based on cross-validation results.

3. Random Search: Select hyper parameter values at random from a given distribution. It is frequently more efficient than grid search in high-dimensional hyper parameter spaces.

4. Bayesian Optimisation: A sequential model-based optimisation technique that uses Bayesian inference to choose which hyper parameter values to assess next based on previous results.

5. Automated hyper parameter Tuning Tools: Many libraries and platforms (e.g., scikit-learn's 'GridSearchCV', 'RandomizedSearchCV', or TensorFlow's 'KerasTuner') offer automated tools to help with hyper parameter tuning.

Challenges:

Computational Cost: Evaluating many hyper parameter combinations can be computationally intensive, particularly for complicated models or big datasets.

Curse of Dimensionality: The amount of hyper parameters and their possible values can create a high-dimensional search space, making it difficult to locate the best combination effectively.

In conclusion, hyper parameter tuning is critical in machine learning for optimising model performance, improving generalisation, and ensuring that the chosen model is appropriate for the specific dataset and situation at hand. It entails selecting the optimal set of hyper parameters via methodical experimentation and evaluation of various configurations.

11. What issues can occur if we have a large learning rate in Gradient Descent?

Answer - When a high learning rate is utilised in Gradient Descent, various challenges might arise, most notably the stability and convergence of the optimisation process. Here are the major issues:

1. Overshooting the Minimum: - High learning rate leads to substantial parameter modifications in each iteration. This can cause the algorithm to overshoot the minimum of the loss function and oscillate around the ideal position. Instead of smoothly converging, the parameters may repeatedly hop back and forth across the minimum, failing to settle on the ideal values.

2. Divergence: In extreme circumstances, a high learning rate might lead parameters to diverge, moving away from ideal values instead of converging. This occurs because the steps taken are so vast that the loss function's values increase rather than decrease.

3. Instability in Training: High learning rates might cause instability in the training process. The changes may differ significantly from one iteration to the next, making it difficult for the model to converge on a solution. As a result, the training process may yield inconsistent or unreliable outcomes.

4. Difficulty in determining optimal values: With a high learning rate, the gradient descent technique may fail to identify the true minimum of the loss function. It may settle at a suboptimal or fluctuating zone near the minimum rather than correctly determining the ideal parameter values.

5. Poor Generalization: Models with high learning rates are more prone to over fit training data. This is because they quickly modify to fit the training examples too closely, capturing noise and specifics from the training data rather than learning general patterns that apply well to fresh, unseen data.

6. Computational Challenges: - High learning rates can raise computational expenses. While larger steps may result in faster convergence in principle, in practice, instability and the necessity for more frequent adjustments (due to oscillations or overshooting) may necessitate more iterations, cancelling

out any possible time savings.

Addressing Large Learning Rate Issues:

To solve the challenges produced by a high learning rate, numerous solutions might be used:

Learning Rate Scheduling: Use techniques such as learning rate decay or adaptive learning rates (such as Adam's optimizer) to dynamically alter the learning rate based on training progress.

Gradient Clipping: Limit the size of gradients during training to prevent too large updates. This helps to stabilise training without slowing the learning rate too much.

Grid Search or Random Search**: Experiment with different learning rates systematically to discover the best value using cross-validation or validation set performance evaluation.

Normalised Input Data: Ensure that input features are correctly normalised (e.g., scaled to zero mean and unit variance) to help stabilise gradient descent and limit sensitivity to high learning rates.

In summary, while a high learning rate helps speed up gradient descent, it also increases the risk of instability, oscillation, and failure to converge. The learning rate must be carefully selected and tuned to provide steady and effective training of machine learning models.

12. Can we use Logistic Regression for classification of Non-Linear Data? If not, why?

Answer-

Logistic Regression is a linear classifier, which means it assumes a linear relationship between the features and the target variable's log odds. This assumption limits its capacity to model nonlinear interactions in data. Here's why Logistic Regression may not be appropriate for categorization of nonlinear data:

1. Linear Decision Boundary: Logistic Regression models use a linear decision boundary in feature space. This means it can only distinguish between classes using a straight line (in 2D) or a hyperplane (in higher dimensions).
- If the connection between the characteristics and the target variable is non-linear, Logistic Regression may struggle to capture and describe this complexity appropriately.

2. Under fitting: When the underlying decision boundary is non-linear, Logistic Regression may under fit data. It may struggle to fit the training data correctly, resulting in poor performance on both training and test datasets.
Under fitting arises when Logistic Regression fails to capture the intricate patterns or interactions found in nonlinear data distributions.

3. Limited Expressiveness: Logistic Regression models struggle to capture complicated relationships and interactions between features. They are fundamentally limited to linear transformations of the input information.

-Logistic Regression cannot directly capture non-linear relationships, such as those involving polynomial features or variable interactions, without first modifying the input features (which might be time-consuming and ineffective).

4. Alternative Models for Non-Linear Data: - Flexible models, such as:
Support Vector Machines (SVMs) using non-linear kernels (e.g., polynomial kernel, Gaussian RBF kernel) can successfully represent non-linear decision boundaries.
Decision Trees and Ensemble Methods (e.g., Random Forests, Gradient Boosting Machines) can deal with non-linear interactions by dividing the feature space into regions and combining weak learners.
Neural Networks (particularly deep neural networks) are highly adaptable and capable of learning complex non-linear mappings from inputs to outputs, making them ideal for a variety of classification problems.

In conclusion, Logistic Regression is not suitable for classifying non-linear data due to its linear assumption and limited ability to capture complicated relationships. For situations in which the relationship between features and target variables is non-linear, it is recommended to investigate alternative machine learning models that can successfully accommodate and model such complexity.

13. Differentiate between Adaboost and Gradient Boosting.

Answer - Adaboost (Adaptive Boosting) and Gradient Boosting are two ensemble learning strategies that are used to improve the performance of machine learning models, particularly while boosting. Despite having the same goal of integrating numerous weak learners into a strong learner, they take different approaches and update the model in each iteration.

Adaboost (Adaptive Boosting)

1.Sequential Training: Adaboost trains weak learners (such as decision trees or stump models) on frequently changed data.
- Each succeeding weak learner concentrates on examples that prior learners erroneously labelled.

2. Weighted Data Points: - Adaboost applies higher weights to cases misclassified in prior iterations. This focus on misclassified examples allows the model to gradually improve by addressing its errors.

3. Final Prediction: - We combine the forecasts of all weak learners and weight them based on their individual accuracy throughout training.
- Typically, more accurate weak learners contribute more to the overall forecast.

4. Example: - Adaboost algorithms include Random Forests.

14. What is bias-variance trade off in machine learning?

Answer –

The bias-variance trade-off is a key concept in machine learning that describes the relationship between a model's capacity to capture underlying patterns in data (bias) and its ability to adapt to new, unknown data (variance). This is a full explanation:

Bias:

Definition: Bias is the error caused by approximating a real-world problem with a simplified model. It determines how far the predictions deviate from the genuine values across different training datasets.

High Bias: A model with a high bias will oversimplify the situation. It may consistently overlook important correlations between characteristics and the target variable (underfitting).

Low Bias: A model with low bias accurately detects the underlying patterns in training data. It works nicely on both the training and test datasets.

Variance:

Definition: Variance assesses the model's sensitivity to tiny fluctuations in the training data. It estimates how much the predictions change between training datasets.

High Variance: A model with a high variance is extremely susceptible to noise or random variations in the training data. It detects random noise rather than genuine underlying patterns (overfitting).

minimal Variance: A model with minimal variance performs well when applied to new, previously unknown data. It does not alter significantly between training datasets and captures fundamental patterns rather than noise.

Trade-off:

Bias and Variance Trade-off: In machine learning, improving the bias typically increases the variance, and vice versa. There is a trade-off between bias and variance, as eliminating one frequently increases the other.

Optimal Model: The goal is to identify the best balance that reduces both bias and variance in order to obtain good predicting performance on new data.

Practical implications:

Under fitting: Models with high bias and low variance (under fitting) do not reflect the underlying patterns in the data. They perform poorly on both the training and testing datasets.

Overfitting: Models with low bias and high variance (overfitting) match the training data too closely, resulting in noise and unpredictable fluctuations. They perform well on training data but badly on new, unlabelled data.

Techniques for Managing Bias-Variance Trade-offs:

Regularisation: Add a penalty term to the objective function to discourage overly complex models. This helps to limit variance and avoid overfitting.

Cross-validation: Use techniques such as K-fold cross-validation to estimate the model's performance on previously unknown data and modify hyper parameters to achieve the best balance.

Ensemble Methods: Use numerous models (e.g., Bagging, Boosting) to reduce variance by averaging predictions or focusing on hard-to-predict occurrences.

- **Feature Selection**: Choose relevant features and decrease noise in the data to increase the model's capacity to detect meaningful patterns.

Conclusion:

Understanding and managing the bias-variance trade-off in machine learning is critical for developing models that generalise successfully to new data. It entails determining the appropriate level of model complexity and tuning parameters to strike a balance between under fitting and overfitting, hence enhancing the model's prediction performance and reliability.

15. Give short description each of Linear, RBF, Polynomial kernels used in SVM.

Answer –

Here's a brief summary of each type of kernel widely used in Support Vector Machines (SVMs).

1. Linear Kernel:

Description: The linear kernel is the most basic type of kernel function utilised in SVMs. The dot product between the input features is computed by $K(xi,xj)=xiT\cdot xj$

Usage: Suitable for linearly separable data with an intended decision boundary of a straight line in the input space.
Characteristics: When the relationship between features and target is roughly linear, the computation is fast and effective.

2. Radial Basis Function (RBF) Kernel:

Description: The RBF kernel calculates the similarity between data points using the Euclidean distance between them in the feature space. It is given by $K(xi,xj)=exp(-\gamma\|xi-xj\|2)$, where γ is a hyperparameter.

Usage: Versatile kernel for non-linear problems with complex decision boundaries.
Characteristics: Captures complicated relationships between features and targets by mapping them into a higher-dimensional space.

3. Polynomial Kernel:

Description: The polynomial kernel computes the similarity between data points as the polynomial of the dot product of the input features $K(xi,xj)=(\gamma xiT\cdot xj+r)d$, where γ is a scaling factor, rrr is an optional coefficient, and ddd is the degree of the polynomial.

Usage: Useful for instances when data is not linearly separable but can be separated using a higher-degree polynomial.
Characteristics: SVMs may describe non-linear decision boundaries by converting the input space to a higher-dimensional feature space.

Summary:

Linear Kernel: Simple and quick, ideal for linearly separable data.
RBF Kernel: Flexible and powerful, ideal for handling complicated and non-linear data.
Polynomial Kernel: Captures complex non-linear interactions by transforming input features with polynomials.

Choosing the proper kernel is determined by the individual characteristics of the data and the complexity of the decision boundary required for accurate classification or regression tasks in SVMs.