Assignment-5 Part-B

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.

Ans:- TSS (Total Sum of Squares)- TSS is the sum of squared deviations of each data point from the mean of the data. It represents the total amount of variability in the dependent variable. TSS can be decomposed into ESS and RSS using the formula TSS = ESS + RSS.

ESS (Explained Sum of Square):- The sum of squares due to regression (SSR) or explained sum of squares (ESS) is the sum of the differences between the predicted value and the mean of the dependent variable. In other words, it describes how well our line fits the data.

RSS (Residual Sum of Squares):- The residual sum of squares (RSS) measures the level of variance in the error term, or residuals, of a regression model.

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

Ans:- Regularization in machine learning prevents overfitting by reducing model complexity. It helps ensure that the model generalizes well to new data. Here are common regularization techniques:

1. L1 (Lasso): Encourages sparsity by penalizing absolute values of coefficients.
2. L2 (Ridge): Reduces large coefficients with squared penalties.
3. Dropout: Randomly deactivates neurons during training to enhance robustness.
4. Early Stopping: Stops training once performance on validation data declines

4. What is Gini–impurity index?

Ans- The Gini impurity index measures the level of impurity or disorder in a dataset, commonly used in decision trees. A lower Gini impurity indicates more homogeneous or pure data, while a higher Gini Implies more diversity or disorder.

Formula: - $G=1-\sum i=1cpi2G=1-\sum_{i=1c}p_{i2}$, where cc is the number of classes, and $pipi$ is the probability of a sample belonging to class i.

Interpretation: -
1. Gini impurity of 0 means all samplws are in one class (perfect purity).
2. Gini impurity close to 1 indicates high diversity among classes.

Usage indecision Trees: -
1. Decision trees aim to split data to minimize Gini impurity, leading to more distinct, effective splits.

5. Are unregularized decision-trees prone to overfitting? If yes, why?

Ans- Yes, unregularized decision trees can overfit because they grow too complex. Here's

1. Too Many Splits: Decision trees can split until every detail is captured, including noise, making them too specific to the training data.

2. Lack of Limits: Without rules to control their size or depth, trees can become huge, learning patterns that don't apply to new data.

3. Small Subsets: Deep trees create tiny groups, leading to decisions based on limited information, which can lead to high variability.

To prevent overfitting, you can regularize decision tree by:

1. Setting a Max Depth: Restricts how deep the tree can go.

2. Minimum Samples for Split: Ensures there's enough data to justify a split.

3. Pruning: Cuts off less useful branches to simplify the tree.

6. What is an ensemble technique in machine learning?

Ans- Ensemble techniques in machine learning use multiple models together to improve accuracy and reduce errors. By combining several models, ensembles generally produce more reliable results. Here are some common ensemble methods:

1. Bagging (Bootstrap Aggregating): - Builds multiple models by training each on a different subset of the data, then combines their predictions, often by averaging or majority voting. Random Forest is a typical example.

2. Boosting: - Creates models sequentially, where each new model corrects errors made by the previous ones. The final prediction is a weighted combination of all models. Examples include AdaBoost and Gradient Boosting.

3. Stacking: - Combines predictions from several base models using a "meta-model" to get the final result.

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

Ans- The difference between Bagging and Boosting Techniques.

	Bagging	**Boosting**
Basic Concept	Combines multiple models trained on different subsets of data.	Train models sequentially, focusing on the error made by the previous model.

Objective	To reduce variance by averaging out individual model error.	Reduces both bias and variance by correcting misclassifications of the previous model.
Data Sampling	Use Bootstrap to create subsets of the data.	Re-weights the data based on the error from the previous model, making the next models focus on misclassified instances.
Model Weight	Each model serves equal weight in the final decision.	Models are weighted based on accuracy, i.e., better-accuracy models will have a higher weight.
Error Handling	Each model has an equal error rate.	It gives more weight to instances with higher error, making subsequent model focus on them.
Overfitting	Less prone to overfitting due to average mechanism.	Generally not prone to overfitting, but it can be if the number of the model or the iteration is high.
Performance	Improves accuracy by reducing variance.	Achieves higher accuracy by reducing both bias and variance.
Common Algorithms	Random Forest	AdaBoost, XGBoost, Gradient Boosting Mechanism
Use Cases	Best for high variance, and low bias models.	Effective when the model needs to be adaptive to errors, suitable for both bias and variance errors.

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

Ans- Out-of-bag (OOB) error, also called out-of-bag estimate, is a method of measuring the prediction error of random forests, boosted decision trees, and other machine learning models utilizing bootstrap aggregating (bagging). Bagging uses subsampling with replacement to create training samples for the model to learn from.

9. What is K-fold cross-validation?

Ans:- Cross-validation is a resampling technique used to validate machine learning models against a limited sample of data. In this article we will talk about K-fold Cross-validation and its advantages and disadvantages. It's working is shown with python program.

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

Ans- These are used to specify the learning capacity and complexity of the model. Some of the hyper parameters are used for the optimization of the models, such as Batch size, learning rate, etc., and some are specific to the models, such as Number of Hidden layers, etc.

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?

Ans- R-squared is a commonly used measure of goodness of fit in regression, indicating the proportion of variance explained by the model. However, Adjusted R-squared is often better for multiple regression as it accounts for the number of predictors. RSS provides an absolute measure of fit but is not as interpretable or comparable across different models.

In the context of regression analysis, both R-squared (R) and Residual Sum of Squares (RSS) are used as measures of goodness of fit for a model. R^2, the coefficient of determination, indicates the proportion of variance in the dependent variable that can be explained by the regression equation. It is commonly used because it is relatively easy to interpret; the closer to 1.0, the better the model explains the variance in the data.

RSS, on the other hand, is the sum of the squares of the differences between the observed values and the values predicted by the model. Minimizing RSS is the objective of the least-squares criteria. However, while RSS provides an absolute measure, it isn't comparable across models with different sample sizes or numbers of predictors.

Thus, in the context of multiple regression, Adjusted R-squared is often a better measure than both R-squared and RSS because it accounts for the number of predictors in the model, fostering comparison between models with different numbers of independent variables without being biased towards models with more variables.

The standard error of the regression,derived from the square root of the residual mean square, is an additional helpful statistic as it offers an average distance that the observed values fall from the regression line.

11. What issues can occur if we have a large learning rate in Gradient Descent?
Ans- Gradient descent is an optimization algorithm used in machine learning to minimize the cost function by iteratively adjusting parameters in the direction of the negative gradient, aiming to find the optimal set of parameters.

The cost function represents the discrepancy between the predicted output of the model and the actual output. The goal of gradient descent is to find the set of parameters that minimizes this discrepancy and improves the model's performance.
The algorithm operates by calculating the gradient of the cost function, which indicates the direction and magnitude of steepest ascent. However, since the objective is to minimize the cost function, gradient descent moves in the opposite direction of the gradient, known as the negative gradient direction.
Gradient descent can be applied to various machine learning algorithms, including linear regression, logistic regression, neural networks, and support vector machines. It provides a general framework for optimizing models by iteratively refining their parameters based on the cost function.
The learning rate is an important hyperparameter that greatly affects the performance of gradient descent. It determines how quickly or slowly our model learns, and it plays an important role in controlling both convergence and divergence of the algorithm. When the learning rate is too large, gradient descent can suffer from divergence. This means that weights increase exponentially, resulting in exploding gradients which can cause problems such as instabilities and overly high loss values. On the other hand, if the learning rate is too small, then gradient descent can suffer from slow convergence or even stagnation—which

means it may not reach a local minimum at all unless many iterations are performed on large datasets.

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

Ans- Logistic Regression has traditionally been used as a linear classifier, i.e. when the classes can be separated in the feature space by linear boundaries. That can be remedied however if we happen to have a better idea as to the shape of the decision boundary…

Logistic regression is known and used as a linear classifier. It is used to come up with a hyperplane in feature space to separate observations that belong to a class from all the other observations that do not belong to that class. The decision boundary is thus linear. Robust and efficient implementations are readily available (e.g. scikit-learn) to use logistic regression as a linear classifier.

13. Differentiate between Adaboost and Gradient Boosting.

Ans- AdaBoost:- AdaBoost or Adaptive Boosting is the first Boosting ensemble model. The method automatically adjusts its parameters to the data based on the actual performance in the current iteration. Meaning, both the weights for re-weighting the data and the weights for the final aggregation are re-computed iteratively.

In practice, this boosting technique is used with simple classification trees or stumps as base-learners, which resulted in improved performance compared to the classification by one tree or other single base-learner.
Gradient Boost is a robust machine learning algorithm made up of Gradient descent and Boosting. The word ‘gradient’ implies that you can have two or more derivatives of the same function. Gradient Boosting has three main components: additive model, loss function and a weak learner.

The technique yields a direct interpretation of boosting methods from the perspective of numerical optimisation in a function space and generalises them by allowing optimisation of an arbitrary loss function.
The Comparison

1)Loss Function:
2)Flexibility
3)Benefits
4)Shortcomings
5)Wrapping
Logistic regression has traditionally been used to come up with a hyperplane that separates the feature space into classes. But if we suspect that the decision boundary is nonlinear we may get better results by attempting some nonlinear functional forms for the logit function. Solving for the model parameters can be more challenging but the optimization modules in scipy can help.

14. What is bias-variance trade off in machine learning?
Ans- Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.

Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.

If our model is too simple and has very few parameters then it may have high bias and low variance. On the other hand if our model has large number of parameters then it's going to have high variance and low bias. So we need to find the right/good balance without overfitting and underfitting the data.

This tradeoff in complexity is why there is a tradeoff between bias and variance. An algorithm can't be more complex and less complex at the same time.

Total Error

To build a good model, we need to find a good balance between bias and variance such that it minimizes the total error.

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

Ans- RBFs are similar to MLPs with three layers (input, middle or “hidden” layer, and output). Also like MLPs, RBFs can model any nonlinear function easily. The major difference between the two networks is that an RBF does not input raw input data but rather it passes a distance measure from the inputs to the hidden layer. This distance is measured from some center value in the range of the variable (sometimes the mean) to a given input value in terms of a Gaussian function.

These distances are transformed into similarities that become the data features worked with in a succeeding regression step. This nonlinear function can permit the mapping operation to capture many nonlinear patterns in the input data.

The processing of RBFs (like any neural network) is iterative. The weights associated with the hidden nodes are adjusted following some strategy (like back propagation). If a large enough RBF is run through enough iterations, it can approximate almost any function almost perfectly; that is, it is theoretically a universal approximator. The problem with RBF processing (like with the MLP) is the tendency to overtrain the model.