MACHINE LEARNING

ASSIGNMENT - 5

Q1 to Q15 are subjective answer type questions, Answer them briefly.

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?

R-squared (Coefficient of Determination) and **Residual Sum of Squares (RSS)** both measure the goodness of fit in a regression model, but they do so differently and serve different purposes. **R-squared** is generally the better measure of the goodness of fit in regression models due to its interpretability and ability to compare models, especially across different datasets.

R-squared represents the proportion of the variance in the dependent variable that is explained by the independent variables in the model. It explains how much of our data is being explained by our model and It ranges from 0 to 1. The closer the value of R-square to 1, the better the model fits the data.

RSS measures the sum of the squared differences between the observed and predicted values in the model (i.e., the total squared error). A smaller RSS means a better fit of the model because it indicates that the residuals (errors) are smaller.

R-squared allows for the comparison of models on different datasets and with different scales, and is easily interpretable while RSS cannot

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.

TSS measures the total variance in the observed data, which is the sum of the squared differences between each observed value and the mean of the observed values.

$TSS= \sum^{n}_{i=1}(yi-\bar{y})^2$

yi= is the actual observed value.

$\bar{y}$ = the mean of the observed values.

n =is the number of observations.

ESS is the sum of the squared differences between the predicted values and the mean of the observed values.

$ESS= \sum^{n}_{i=1}(\hat{y}i-\bar{y})^2$

$\hat{y}$= Predicted value

RSS measures the variation in the dependent variable that is **not explained** by the regression model. It is the sum of the squared differences between the observed values and the predicted values (the residuals or errors).

RSS= $^{n}\sum_{i=1}(y\hat{}i-y^{-})^2$

TSS=ESS+RSS

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

When using regression model to train some data, there is a good chance that the mode will overfit the given training data set. Regularization helps sort this overfitting problem i.e it prevent overfitting by discouraging overly complex models and it encourages better generalization, ensuring good performance on both training and unseen data. Without regularization, a machine learning model is likely to memorize the training data rather than learning general patterns, leading to poor performance on new data.

4. What is Gini–impurity index?

Gini Impurity Index is a metric used in decision tree algorithms, specifically for classification tasks, to evaluate how "pure" a node is. The Gini impurity index is used to split the nodes in decision trees, where lower Gini impurity values indicate better splits. The goal of decision trees is to split the dataset in such a way that each resulting node contains mostly instances of a single class (meaning; the node is "pure").

$$Gini\ Impurity = \sum_{i=1}^{c} Pi^2$$

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

Yes, unregularized decision trees are prone to overfitting because they can grow too deep and create very specific splits that fit the training data perfectly, including noise. Regularization techniques help control the complexity of decision trees and improve their ability to generalize to new data, reducing overfitting.

6. What is an ensemble technique in machine learning?

Ensemble technique in machine learning combine the predictions of multiple models to improve the overall performance of the model. They reduce the risk of overfitting, improve accuracy, and enhance the generalizability of the model by leveraging the strengths of individual models. Common ensemble methods include **bagging**, **boosting**, **stacking**, and **voting**

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

The difference between Bagging and boosting techniques are

- Training data subsets are drawn randomly with replacement from the entire training data while Boosting New subsets contains the component that were misclassified by previous models
- Bagging attempts to tackle the overfitting issue while boosting tries to reduce bias
- Every model receives an eqal weight for bagging while models are weighed by their performance, objective to decrease bias no variance.
- Every models built independently for bagging while models are affected by the performance of the previously developed models in boosing.

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

The **out-of-bag (OOB) error** is a useful and efficient method for validating the performance of a random forest model. It leverages the bootstrapping process to estimate the error without requiring a separate validation set, making it an effective tool for measuring the model's ability to generalize to unseen data.

Its benefits includes;

- Efficient data usage by validating data that were not seen by individual trees leading to a more robust performance evaluation
- OOB error provides an internal, unbiased estimate of model performance without the need to set aside a validation set. This allows you to use the entire training data set for both training and validation.
- OOB error is often a good approximation of the model's **test error**, i.e it provides a reliable estimate of how the model will generalize to new, unseen data.

9. What is K-fold cross-validation?

Is a technique in machine learning used to evaluate the performance of a model in a way that is both reliable and efficient. It involves dividing the dataset into **K** equally sized (or nearly equal) **folds**, or subsets, and then using each fold as a validation set while the remaining **K-1** folds are used for training. This process is repeated **K times**, with each fold used exactly once as the validation set. The final result is the average performance of the model across all K trials.

Pictorial illustration example: if Cross Validation is 5

Training set

Test set

Iteration

1	**2**	**3**	**4**	**5**
TEST	Training set	Training set	Training set	Training set
Training set	TEST	Training set	Training set	Training set

Training set	Training set	TEST	Training set	Training set
Training set	Training set	Training set	TEST	TEST
Training set	Training set	Training set	Training set	Training set

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

Hyperparameter tuning is o the process of selecting the best set of hyperparameters for a machine learning model to improve its performance on unseen data. **Hyperparameters** are settings or configurations that are external to the model and cannot be learned directly from the training data, unlike model parameters (e.g., weights in neural networks). They control the learning process and affect how the model is trained. Example include;The number of neighbors in K-nearest neighbors (KNN), the learning rate in neural networks and the depth of trees in decision trees etc.

1. It is done to help to prevent overfitting, where the model learns too much detail from the training data, making it perform poorly on new, unseen data.
2. It also controls learning process, parameters like learning rate, momentum, and batch size in optimization algorithms can control how quickly or slowly a model learns. If these values are too large or too small, the model may fail to converge or take too long to train.

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

When using Gradient Descent to optimize a machine learning model, the learning rate is a key hyperparameter that determines the step size for updating the model's parameters during each iteration. A large learning rate in Gradient Descent can cause the following:

1. **Overshooting the Minimum**: A large learning rate can cause the model to take excessively large steps in the direction of the gradient. As a result, instead of gradually converging toward the optimal minimum (the lowest point in the loss function), the updates may overshoot the target. The model can "jump" over the minimum, oscillating back and forth without ever converging.

- **Effect**: The model fails to converge, and the loss function does not decrease as expected.

2. **Divergence**:If the learning rate is extremely large, the model may diverge completely. This means that the updates become so large that the model moves further and further away from the optimal solution, causing the loss function to increase exponentially with each iteration.

- **Effect**: The loss function can grow without bound, and training fails entirely.

3. **Instability and Oscillations**: with a large learning rate, the parameter updates can cause large swings in the opposite direction of the gradient. This leads to instability in

the learning process, where the loss function fluctuates wildly instead of decreasing steadily.

- **Effect**: The training process becomes unstable, with the model parameters oscillating around the minimum but never settling down.

4. **Suboptimal Convergence**: Even if the model manages to converge with a large learning rate, the final solution might not be optimal. The model might stop near a local minimum but miss the global minimum or better solutions.

- **Effect**: The model achieves suboptimal performance, as it converges to a point that isn't the true minimum of the loss function.
- **Loss of Generalization**: Large learning rates can cause the model to "overreact" to the training data, resulting in updates that are too aggressive. This can harm the model's ability to generalize to new, unseen data, as it may learn noisy or irrelevant features from the training set.
- **Effect**: The model may overfit to the training data and underperform on the test set, compromising its generalization.

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

No. Logistic regression is a **linear model**, meaning that it is designed to model linear relationships between the input features and the target variable.

13. Differentiate between Adaboost and Gradient Boosting.

AdaBoost (Adaptive Boosting) and **Gradient Boosting** are both popular ensemble techniques in machine learning that aim to improve model performance by combining the predictions of several weak learners (typically decision trees). However, the way they achieve this is different.

AdaBoost:	**Gradient Boosting**:
It focuses on improving model performance by adjusting the weights of misclassified instances after each weak learner is added. The next weak learner focuses more on these harder-to-classify examples, making it adaptive to errors.	It builds the model by sequentially adding weak learners that correct the residual errors (or gradients) of the previous learners. It optimizes the model by minimizing a differentiable loss function using gradient descent.
After each weak learner (often a decision stump) is trained, the instances that were incorrectly classified are given higher weights, and the instances that were classified correctly are given lower weights.	It sequentially adds weak learners to the model, but instead of adjusting sample weights, it fits the next learner to the residual errors of the previous learner.

The final prediction is a weighted majority vote (for classification) or weighted sum (for regression) of all the weak learners.	The final prediction is the sum of predictions from all the weak learners.
It gives more importance (higher weights) to misclassified instances so that future learners focus on correcting those errors.	It fits each new weak learner to the residuals (errors) of the previous learner, directly minimizing the errors at each step.

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

Bias-variance tradeoff is a concept in machine learning that refers to the balance between two sources of error that affect a model's performance: **bias** and **variance**. Understanding this tradeoff is crucial for building models that generalize well to new, unseen data.

Bias is the error introduced by approximating a real-world problem, which may be highly complex, by a simplified model. High bias; Indicates that the model is too simplistic and does not capture the underlying patterns in the data well. This can lead to **underfitting**.

Variance refers to the model's sensitivity to small fluctuations or noise in the training data. A model with high variance fits the training data very well but may fail to generalize to new data. When there is high variance, the model captures the noise or random fluctuations in the training data, leading to **overfitting.**

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

Linear Kernel: is the simplest kernel, which does not map the data into a higher dimension. It simply computes the dot product between two input vectors. This is used when the data is linearly separable. It is effective when the data is linearly separable or nearly so, and it's computationally efficient.

RBF (Radial Basis Function) Kernel: Also known as the Gaussian Kernel, the RBF kernel maps the data into an infinite-dimensional space. It's useful for non-linear problems by measuring the similarity between points based on their distance. It is commonly used in practice when there is no clear prior knowledge about the data's linearity. Effective for non-linear classification tasks.

Polynomial Kernel: This kernel maps the data into a higher-dimensional space using polynomial functions of the input data. It allows fitting non-linear data in a polynomial space. It is suitable for data with non-linear boundaries. Higher degrees of polynomials increase the complexity, allowing more flexibility but can also risk overfitting.