ASSIGNMENT 2-1

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?

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.

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

4. What is Gini–impurity index?

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

6. What is an ensemble technique in machine learning?

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

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

9. What is K-fold cross-validation?

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

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

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

13. Differentiate between Adaboost and Gradient Boosting.

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

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

1. R-squared vs. Residual Sum of Squares (RSS) for Goodness of Fit

R-squared is generally considered a better measure of goodness of fit compared to **Residual Sum of Squares (RSS)** because:

- **R-squared**:
  - **Normalized Measure**: Provides a proportion of the variance in the dependent variable that is explained by the independent variables. It ranges from 0 to 1, where higher values indicate a better fit.
  - **Comparison**: Allows comparison of models with different numbers of predictors and scales.
- **RSS**:
  - **Absolute Measure**: Represents the total squared errors between observed and predicted values. Lower RSS values indicate better fit, but it does not account for the number of predictors and can decrease with more predictors.

2. TSS, ESS, and RSS in Regression

- **TSS (Total Sum of Squares)**: Measures the total variance in the dependent variable.
  - **Formula**: TSS=∑(yi−y¯)2
  - Where yi is the observed value and y¯ of the observed values.
- **ESS (Explained Sum of Squares)**: Measures the variance explained by the regression model.
  - **Formula**: ESS=∑(y^i−y¯)2ESS = \sum (\hat{y}_i - \bar{y})^2ESS=∑(y^i−y¯)2
  - Where y^i\hat{y}_iy^i is the predicted value from the model.
- **RSS (Residual Sum of Squares)**: Measures the variance not explained by the model.
  - **Formula**: RSS=∑(yi−y^i)2RSS = \sum (y_i - \hat{y}_i)^2RSS=∑(yi−y^i)2
- **Relationship**: TSS=ESS+RSSTSS = ESS + RSSTSS=ESS+RSS This equation indicates that the total variance in the data is partitioned into explained and residual variances.

3. Need for Regularization in Machine Learning

Regularization is needed to:

- **Prevent Overfitting**: By adding a penalty for large coefficients, regularization helps prevent the model from fitting the noise in the training data.
- **Improve Generalization**: Helps the model generalize better to unseen data by discouraging overly complex models.
- **Stabilize Learning**: Reduces the variance of the model, leading to more stable and robust predictions.

4. Gini–Impurity Index

The **Gini impurity index** is a measure used in decision trees to evaluate the quality of a split. It quantifies the degree of impurity or disorder in a dataset:

- **Formula**: Gini=1−∑(pi)2Gini = 1 - \sum (p_i)^2Gini=1−∑(pi)2 Where pip_ipi is the proportion of samples in class iii.
- **Interpretation**: A lower Gini impurity indicates a better split, with more homogenous groups.

5. Unregularized Decision Trees and Overfitting

Yes, unregularized decision trees are prone to overfitting. This happens because:

- **High Variance**: They can create very complex models with deep trees that perfectly fit the training data but fail to generalize to new, unseen data.
- **Over-Splitting**: Unrestricted trees can split data into very small subsets, capturing noise rather than the underlying patterns.

6. Ensemble Technique in Machine Learning

An **ensemble technique** combines predictions from multiple models to improve performance and robustness. It leverages the strengths of individual models to make a final prediction, often leading to better generalization than any single model.

7. Difference Between Bagging and Boosting

- **Bagging (Bootstrap Aggregating)**:
  - **Approach**: Trains multiple models independently on different subsets of the data (bootstrap samples) and averages their predictions.
  - **Purpose**: Reduces variance and helps prevent overfitting.
  - **Example**: Random Forests.
- **Boosting**:
  - **Approach**: Trains models sequentially, with each new model focusing on correcting errors made by previous models.
  - **Purpose**: Reduces bias and improves the model's accuracy by combining weak learners into a strong learner.
  - **Example**: AdaBoost, Gradient Boosting.

8. Out-of-Bag Error in Random Forests

Out-of-bag (OOB) error is an estimate of the model's generalization error, calculated by evaluating each training sample on the trees for which it was not included in the bootstrap sample:

- **Calculation**: Each sample is used to test the model that was not trained on it, and the average error is computed.
- **Purpose**: Provides an unbiased estimate of the model's performance without needing a separate validation set.

9. K-fold Cross-Validation

K-fold cross-validation is a technique for evaluating model performance by dividing the dataset into kkk subsets (or folds). The model is trained kkk times, each time using k−1k-1k−1 folds for training and the remaining fold for testing:

- **Procedure**: The process is repeated kkk times with each fold serving as the test set once. The results are averaged to get the final performance metric.
- **Purpose**: Provides a better estimate of model performance and reduces variability in the performance measure.

10. Hyperparameter Tuning in Machine Learning

Hyperparameter tuning involves adjusting the settings of a model (e.g., learning rate, number of trees) to improve its performance. It is done to:

- **Optimize Model Performance**: Fine-tuning hyperparameters can lead to better fitting of the model to the data.
- **Avoid Overfitting/Underfitting**: Helps in finding the right balance between model complexity and generalization.

11. Issues with Large Learning Rate in Gradient Descent

A large learning rate in Gradient Descent can cause:

- **Divergence**: The model parameters may oscillate or diverge instead of converging to the optimal values.
- **Instability**: Can lead to a loss function that jumps around and fails to settle down to a minimum.

12. Logistic Regression for Non-Linear Data

Logistic Regression is not well-suited for classification of non-linear data unless transformed features or interactions are used. This is because:

- **Linear Decision Boundary**: Logistic Regression assumes a linear decision boundary between classes. For non-linear relationships, it may not capture the complex patterns in the data.

13. Difference Between AdaBoost and Gradient Boosting

- **AdaBoost**:
  - **Focus**: Adjusts the weights of incorrectly classified samples, emphasizing difficult cases in subsequent models.
  - **Learning**: Sequentially adds models to correct errors of previous models.
- **Gradient Boosting**:
  - **Focus**: Fits new models to the residuals (errors) of the combined predictions of previous models.
  - **Learning**: Optimizes a loss function through gradient descent, adding models to minimize the residual error.

14. Bias-Variance Trade-off in Machine Learning

The **bias-variance trade-off** is the balance between:

- **Bias**: Error due to overly simplistic models (underfitting), which may fail to capture the underlying patterns in the data.
- **Variance**: Error due to overly complex models (overfitting), which may capture noise as if it were a signal.

The goal is to find a model that minimizes both bias and variance to achieve optimal generalization.

15. Kernels Used in SVM

- **Linear Kernel**:
  - **Description**: Computes the dot product of input vectors in the original feature space.
  - **Formula**: K(x,x′)=x·x′K(x, x') = x \cdot x'K(x,x′)=x·x′
  - **Use**: Suitable for linearly separable data.
- **RBF (Radial Basis Function) Kernel**:
  - **Description**: Measures similarity based on the distance between vectors, using an exponential function.
  - **Formula**: K(x,x′)=exp⁡(−∥x−x′∥22σ2)K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)K(x,x′)=exp(−2σ2∥x−x′∥2)
  - **Use**: Effective for non-linear relationships.
- **Polynomial Kernel**:
  - **Description**: Computes the dot product raised to a power, allowing for polynomial decision boundaries.
  - **Formula**: K(x,x′)=(x·x′+c)dK(x, x') = (x \cdot x' + c)^dK(x,x′)=(x·x′+c)d
  - **Use**: Useful for capturing polynomial relationships between features.