Assignment 02 part 01 by Mohsin Hilal Qadri

Q. **NO.01** 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:

R-squared (R^2) :

R-squared tells us how much of the variation in our dependent variable (the thing we're trying to predict) is explained by your independent variables (the things we're using to make predictions).

Range: It goes from 0 to 1.

- **0**: model doesn't explain any of the variation.
- **1**: model explains all of the variation.
- **How to use it**: The closer R-squared is to 1, the better our model fits the data. It's like a percentage: an R-squared of 0.8 means 80% of the variation is explained by your model.

Residual Sum of Squares (RSS) :

RSS is the total of the squared differences between the actual values and the values predicted by your model.

Range: It can be any non-negative number, starting from 0 and going up.

- **0**: Perfect fit (your predictions are exactly right).
- **Larger numbers**: Worse fit (your predictions are further off).

 How to use it: The smaller the RSS, the better your model fits the data. But RSS depends on the scale of your data, so it's not as straightforward to compare between different datasets.

Which is Better for Measuring Goodness of Fit?

 R-squared: Easier to understand and compare. It gives a clear, standardized measure of how well your model is doing.

 RSS: Useful in specific contexts but harder to interpret because its value depends on the scale of the data.

Conclusion:

 R-squared is generally a better measure of goodness of fit because it is easier to understand and compare. It gives a percentage-like figure that tells you how well your model explains the data.

 RSS is more detailed but less intuitive. It's good for understanding the exact differences between predicted and actual values, but its usefulness is limited by its dependency on the data's scale.

Q.NO.02 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:

Total Sum of Squares (TSS):

TSS measures the total variance in the dependent variable. It quantifies the total variation of the observed data points from their mean.

Formula:

$$TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

Where y_i is the observed value, and $\bar{y}$ is the mean of the observed values.

Explained Sum of Squares (ESS) :

ESS measures the variation explained by the regression model. It quantifies how much of the total variation in the dependent variable is explained by the independent variables.

Formula:

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

Where $\hat{y}_i$ is the predicted value by the model, and $\bar{y}$ is the mean of the observed values.

Residual Sum of Squares (RSS) :

RSS measures the variation that is not explained by the model. It quantifies the discrepancies between the observed values and the predicted values.

Formula:

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

where $\hat{y}_i$ is the predicted value by the model, and $\bar{y}$ is the mean of the observed values.

Relationship Between TSS, ESS, and RSS

TSS=ESS+RSS

This means the total variation (TSS) is equal to the explained variation (ESS) plus the unexplained variation (RSS).

Q.no 03 What is the need of regularization in machine learning?

Answer: Regularization in machine learning helps to prevent overfitting, which is when a model performs well on training data but poorly on new, unseen data. By adding a penalty for more complex models, regularization encourages the model to be simpler and more generalizable. This leads to better performance on test data. Techniques like L1 (Lasso) and L2 (Ridge) regularization are commonly used to achieve this balance.

Q. no 04: What is Gini–impurity index?

Answer: The Gini impurity index is a metric used in decision trees to measure the impurity or disorder of a dataset. It helps to determine the best way to split the data at each node in the tree. The lower the Gini impurity, the more homogeneous (pure) the data is in terms of class labels.

Formulla:

Gini $= 1 - \sum_{i=1}^{c} p_i^2$

where:

- C is the number of classes,
- p_i is the proportion of instances in the node belonging to class i.

Use in Decision Trees :

During the construction of a decision tree, the Gini impurity is used to evaluate the quality of splits. The goal is to choose splits that minimize the Gini impurity in the child nodes, thereby making them as pure as possible. This process helps in creating a decision tree that can classify new data points accurately.

Q.no.05 : Are unregularized decision-trees prone to overfitting? If yes, why?

Answer : Yes, unregularized decision trees often overfit the data. This happens because they keep splitting the data until each leaf has very few data points, capturing all the details and noise in the training data. This makes the tree too specific to the training data, leading to poor performance on new data. Regularization methods like limiting the tree depth or pruning help avoid this by simplifying the tree.

Q. no 06 : What is an ensemble technique in machine learning?

Answer : An ensemble technique in machine learning combines multiple models to produce better predictions than any single model could achieve alone. It works by aggregating the predictions of different models to improve accuracy and robustness. Ensemble methods are widely used in practice because they reduce overfitting and increase predictive performance by leveraging the strengths of diverse models.

Benefits

- **Improved Accuracy**: Combining multiple models can lead to better performance than any single model.
- **Reduced Overfitting**: Ensemble methods can mitigate the risk of overfitting, particularly with complex models.

Q. no 07 : What is the difference between Bagging and Boosting techniques?

Answer:

Bagging (Bootstrap Aggregating)

- **Method**:
  - Uses multiple parallel models (often the same type) trained on different subsets of data.
  - Each model is trained independently.
  - Final prediction is averaged (regression) or voted (classification) from all models.
- **Example**: Random Forest

Boosting

- **Method**:
  - Sequentially trains models, where each new model corrects errors made by the previous ones.
  - Models are weighted based on their performance.
  - Final prediction is a weighted sum of predictions from all models.
- **Examples**: AdaBoost, Gradient Boosting Machines (GBM)

Conclusion

- **Bagging** reduces variance by averaging predictions from multiple models.
- **Boosting** reduces bias by sequentially improving model predictions based on previous mistakes.

Q. no. 08 : What is out-of-bag error in random forests?

Answer:

The out-of-bag (OOB) error in Random Forests is an estimate of how well the model will generalize to new, unseen data. It's calculated by using the data points that were not included in the bootstrap sample used to train each tree. Since these out-of-bag samples were not used in training, they act as a validation set. The OOB error is the average error across all trees in the forest, providing a convenient way to assess the model's performance without needing a separate validation set.

Q. no. 09: What is K-fold cross-validation?

Answer:

K-fold cross-validation is a method used to evaluate how well a machine learning model will generalize to new data. Here's how it works:

- **Splitting**: The data is split into K subsets (folds).
- **Training and Validation**: The model is trained on K-1 folds and validated on the remaining fold.
- **Repeat**: This process is repeated K times, with each fold used once as the validation set.
- **Average Performance**: The performance results from each fold are averaged to get a final performance estimate.

K-fold cross-validation gives a more reliable measure of the model's performance compared to a single train-test split.

Q. no. 10 : What is hyper parameter tuning in machine learning and why it is done?

Answer:

Hyperparameter tuning in machine learning is the process of finding the best settings for the parameters that are set before the learning process begins. These settings greatly impact the performance of the model. Tuning is done to optimize the model's performance, prevent overfitting, and improve its ability to generalize to new data.

Why Hyperparameter Tuning is Done

1. **Optimize Model Performance**: Different hyperparameter settings can result in vastly different model performances. Tuning helps find the set of hyperparameters that maximize model performance metrics such as accuracy, F1-score, or AUC.
2. **Prevent Overfitting**: Proper hyperparameter tuning can prevent overfitting (when the model performs well on training data but poorly on new data) by controlling model complexity.
3. **Improve Generalization**: Tuning helps improve the model's ability to generalize to new, unseen data by finding the optimal balance between bias and variance.
4. **Explore Model Capabilities**: It allows exploration of different combinations of hyperparameters to understand the behavior and capabilities of the model.
5. **Requirement of Different Datasets**: Since different datasets can require different hyperparameters.

Q.no. 11: What issues can occur if we have a large learning rate in Gradient Descent?

Answer:

If the learning rate in Gradient Descent is set too high, several problems can arise:

1. **Divergence**:
   - The updates to the model parameters become too large, causing the optimization process to overshoot the optimal point. This leads to the parameters oscillating and failing to converge to the minimum of the loss function.
2. **Difficulty in Convergence**:
   - A large learning rate may prevent the algorithm from finding the minimum of the loss function. The updates might repeatedly miss the minimum, preventing the model from converging.
3. **Oscillations**:
   - The large updates can cause the model parameters to oscillate around the optimal solution instead of settling at it. This oscillatory behavior prevents the model from stabilizing and converging.
4. **Instability**:
   - Training becomes unstable as the updates are too drastic, making it challenging to consistently improve the model parameters over iterations. This instability can lead to unpredictable performance.
5. **Overfitting**:
   - A high learning rate may cause the model to overfit the training data, as it learns too quickly and adjusts to noise in the data rather than the underlying patterns. This results in poor generalization to new, unseen data.
   -

Q. no. 12: Can we use Logistic Regression for classification of Non-Linear Data? If not, why?

Answer:

No, Logistic Regression is not suitable for the classification of non-linear data. It assumes a linear relationship between the features and the log-odds of the target variable. Therefore, it may not capture complex, non-linear patterns in the data, leading to poor performance in such cases. For non-linear data, it's better to use models like Support Vector Machines with non-linear kernels, Decision Trees, Random Forests, Gradient Boosting Machines, or Neural Networks. These models can handle non-linear relationships and construct non-linear decision boundaries effectively.

Q. no. 13: Differentiate between Adaboost and Gradient Boosting

Answer:

Given below is the difference between differentiation between Adaboost and Gradient Boosting:

Adaboost (Adaptive Boosting)

- **Sequential Training**: Adaboost trains multiple weak learners sequentially.
- **Weighted Samples**: It assigns higher weights to misclassified samples so that subsequent learners focus more on correcting these mistakes.
- **Equal Weight**: All weak learners have equal weight in the final prediction.
- **Base Learners**: Typically uses decision trees with a depth of 1 (decision stumps).

Gradient Boosting

- **Sequential Training**: Gradient Boosting also trains multiple weak learners sequentially.
- **Gradient Descent**: It uses gradient descent technique to minimize the loss function by adding weak learners.
- **Different Weights.

Q. no 14 : What is bias-variance trade off in machine learning?

Answer:

The bias-variance trade-off in machine learning refers to the delicate balance between two types of errors that affect model performance:

Bias

- **Definition**: Bias measures how much the average prediction of a model differs from the true value.
- **High Bias**: Occurs when a model is too simple and fails to capture the underlying patterns in the data (underfitting).

Variance

- **Definition**: Variance measures how much the predictions of a model vary for different training sets.
- **High Variance**: Occurs when a model is too complex and learns not only the underlying patterns but also noise in the training data (overfitting).

Trade-off

- **Objective**: The goal is to find a model that achieves a good balance between bias and variance to generalize well to new, unseen data.
- **Challenge**: Increasing model complexity reduces bias but increases variance, and vice versa.
- **Optimal Model**: The optimal model complexity is found by tuning hyperparameters and using regularization techniques to balance bias and variance.

Q. no 15: Give short description each of Linear, RBF, Polynomial kernels used in SVM.

Answer:

Linear Kernel

- **Description**: Computes the dot product between two vectors.
- **Usage**: Effective for linearly separable data or large-scale datasets.

RBF (Radial Basis Function) Kernel

- **Description**: Measures similarity based on the distance between samples in a high-dimensional space.
- **Usage**: Suitable for capturing complex relationships in non-linear data.

Polynomial Kernel

- **Description**: Computes similarity by mapping data into higher-dimensional space using a polynomial function.
- **Parameters**: Includes degree ddd and optional coefficient rrr.
- **Usage**: Effective for data with non-linear relationships that can be approximated by polynomials.