Worksheet Assignment 5 – Machine Learning

Q1 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 and Residual Sum of Squares (RSS) are both measures of goodness of fit in regression analysis, but they capture different aspects of the model's performance.

Both R-squared and RSS have their own strengths and limitations. R-squared is a more commonly used measure of goodness of fit because it provides a clear interpretation of how much of the variation in the dependent variable is explained by the independent variables. It is also easy to compare the performance of different models based on their R-squared values.

However, R-squared has some limitations. For example, it may overestimate the fit of the model if the independent variables are highly correlated. Additionally, R-squared does not provide any information about the absolute level of fit of the model, which is where RSS comes in.

RSS provides a measure of the total unexplained variation in the dependent variable, which can be useful for assessing the magnitude of errors in the model. It can also be used to compare the performance of different models based on their absolute level of fit.

In summary, both R-squared and RSS are useful measures of goodness of fit in regression analysis. R-squared is more commonly used and provides a relative measure of fit, while RSS provides an absolute measure of fit and can be useful for assessing the magnitude of errors in the model. The choice of which measure to use depends on the specific context and goals of the analysis.

Q2 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), ESS (Explained Sum of Squares), and RSS (Residual Sum of Squares) are three important concepts in regression analysis that help to evaluate the goodness of fit of a regression model.

TSS (Total Sum of Squares) represents the total variation in the dependent variable (y) and is calculated as the sum of the squared differences between the actual values of y and the mean of y. It measures the total deviation of the response variable from its mean and is given by the formula:

$TSS = \Sigma(y_i - \bar{y})^2$

ESS (Explained Sum of Squares) represents the variation in the dependent variable that is explained by the regression model and is calculated as the sum of the squared differences between the predicted values of y ($\hat{y}$) and the mean of y. It measures the deviation of the predicted values of the response variable from its mean and is given by the formula:

$ESS = \Sigma(\hat{y}_i - \bar{y})^2$

RSS (Residual Sum of Squares) represents the variation in the dependent variable that is not explained by the regression model and is calculated as the sum of the squared differences between the actual values of y and the predicted values of y. It measures the deviation of the actual values of the response variable from the predicted values and is given by the formula:

$RSS = \Sigma(y_i - \hat{y}_i)^2$

The relationship between these three measures can be expressed by the following equation:

$TSS = ESS + RSS$

This equation states that the total variation in the dependent variable (TSS) can be decomposed into two components: the variation that is explained by the regression model (ESS) and the variation that is not explained by the model (RSS). The goal of regression analysis is to develop a model that explains as much of the total variation as possible, which is reflected in a high value of R-squared (the square of the correlation coefficient between y and x).

Q3 What is the need of regularization in machine learning?

Ans - Regularization is an important technique used in machine learning to prevent overfitting and improve the generalization performance of a model. Overfitting occurs when a model learns to fit the training data too closely, including the noise in the data, and performs poorly on new, unseen data. Regularization helps to control the complexity of a model and reduce the risk of overfitting by adding a penalty term to the loss function that the model is optimized for.

There are two commonly used regularization techniques in machine learning:

1. L1 regularization (also known as Lasso regularization) adds a penalty term proportional to the absolute value of the model's coefficients. This technique encourages the model to learn sparse representations by setting some of the coefficients to zero, effectively performing feature selection.
2. L2 regularization (also known as Ridge regularization) adds a penalty term proportional to the squared magnitude of the model's coefficients. This technique encourages the model to learn small, non-zero coefficients, effectively shrinking the coefficients towards zero.

Regularization can also help to stabilize the model's parameters, improve convergence during training, and reduce the sensitivity of the model to small changes in the input data.

Q4 What is Gini–impurity index?

Ans - The Gini impurity index is a measure of impurity or heterogeneity used in decision tree algorithms for classification tasks. The Gini impurity of a set of items (e.g., a node in a decision tree) is defined as the probability of misclassifying a random item from the set, given that the item is randomly labeled according to the class distribution in the set.
Mathematically, the Gini impurity index for a node with K classes is given by:
$G = 1 - \Sigma(k=1 \text{ to } K)\ pk^2$
where pk is the fraction of items in the node that belong to class k.
The Gini impurity index ranges from 0 to 1, with 0 indicating a perfectly pure node (all items belong to the same class) and 1 indicating a perfectly impure node (items are evenly distributed across all classes).
In decision tree algorithms, the Gini impurity index is used as a criterion to select the best split for a node. The algorithm considers all possible splits based on the values of the input features and selects the split that results in the largest reduction in Gini impurity between the parent node and the child nodes. This process is repeated recursively for each child node until a stopping criterion (e.g., maximum tree depth, minimum number of items per node) is reached.

Q5 Are unregularized decision-trees prone to overfitting? If yes, why?

Ans - Yes, unregularized decision trees are prone to overfitting. Overfitting occurs when the model becomes too complex and fits the training data too closely, resulting in poor generalization to new, unseen data.
Decision trees can be particularly prone to overfitting because they have the capacity to create very complex models that fit the training data well but may not generalize well to new data. Decision trees are constructed by recursively splitting the data based on the values of input features until the subsets at the leaves of the tree contain predominantly one class or the maximum depth of the tree is reached.
If the decision tree is allowed to grow without any regularization, it will continue splitting the data until each leaf contains only one data point, which results in a model that is perfectly tailored to the training data. However, this model is unlikely

to generalize well to new data because it has memorized the training data rather than learned the underlying patterns and relationships.
Regularization techniques, such as limiting the maximum depth of the tree, or pruning branches that do not contribute significantly to the overall performance of the model, can help prevent overfitting in decision trees.

Q6 What is an ensemble technique in machine learning?

Ans - Ensemble techniques in machine learning are methods that combine multiple models, often of different types or trained on different subsets of the data, to improve the overall performance and predictive accuracy of the model. The idea behind ensemble methods is to leverage the strengths of multiple models and mitigate the weaknesses of individual models.
The two most common types of ensemble techniques are:

1. Bagging stands for bootstrap aggregation, which involves creating multiple replicas of the training data set by randomly sampling with replacement. Then, a separate model is trained on each of these replicas, and the final prediction is made by taking the average or majority vote of the predictions made by all the models.

2. Boosting is a technique that iteratively trains a sequence of weak learners, each one focused on correcting the errors made by the previous model. The final prediction is made by taking a weighted sum of the predictions made by all the models.

Other ensemble techniques include stacking, where the predictions of multiple models are combined as inputs to a meta-model, and ensemble pruning, where models with high error rates are pruned from the ensemble.
Ensemble techniques are particularly effective when the individual models have low bias and high variance, as combining them can help reduce the variance and improve the overall performance of the model. Examples of ensemble methods in machine learning include Random Forests, Gradient Boosted Trees, and AdaBoost.

Q7 What is the difference between Bagging and Boosting techniques?

Ans Bagging and Boosting are two commonly used ensemble learning techniques in machine learning. They are used to improve the performance of weak learners by combining them in different ways.

Bagging stands for Bootstrap Aggregating. It involves training several models independently on different subsets of the training data, where each subset is created by random sampling with replacement. This leads to a diverse set of models, which are then combined by averaging their predictions or taking the majority vote. The idea behind bagging is to reduce the variance of the models by reducing their dependence on the training data.

Boosting, on the other hand, is a sequential learning technique that involves training a sequence of models, where each subsequent model tries to improve the performance of the previous model. In boosting, each model is trained on the same data set, but the weights of the training examples are adjusted to focus more on the examples that were misclassified by the previous model. The idea behind boosting is to reduce the bias of the models by focusing on the hard-to-classify examples.

Q8 What is out-of-bag error in random forests?

Ans - In a random forest algorithm, each tree in the forest is trained on a bootstrap sample of the original dataset, which means that each tree only sees a fraction of the data. This also means that some data points are not included in the sample for each tree. The remaining data points that are not used in the training of a particular tree are known as out-of-bag (OOB) samples.

The out-of-bag error is a way to estimate the performance of a random forest model without the need for cross-validation or a separate validation dataset. It is calculated as the error rate on the OOB samples for each tree in the

forest, and then averaged across all the trees. The out-of-bag error gives an estimate of how well the random forest model will generalize to new data.

Since each tree in the random forest is trained on a different bootstrap sample, the OOB samples for each tree are different. This means that the OOB error estimates are independent and can be used to identify which features are important for classification or regression tasks. By comparing the OOB error rates for different features, one can determine which features are most informative and should be included in the final model.

Q9 What is K-fold cross-validation?

Ans - K-fold cross-validation is a commonly used technique for evaluating the performance of a machine learning model. It is used to estimate the generalization performance of a model on new data, without the need for a separate validation set.

The basic idea behind K-fold cross-validation is to divide the data into K subsets or folds of equal size. The model is then trained on K-1 folds and validated on the remaining fold. This process is repeated K times, with each fold being used once as the validation set. The K validation results are then averaged to obtain a final performance estimate.

The benefits of K-fold cross-validation are that it provides a more reliable estimate of the model performance than a single train/test split, and it makes better use of the available data. It also allows for a more thorough assessment of the model's ability to generalize to new data.

One potential drawback of K-fold cross-validation is that it can be computationally expensive, especially when dealing with large datasets or complex models. However, it can still be a valuable tool for model selection and hyperparameter tuning, as it provides a more robust estimate of the model's performance compared to a single train/test split.

Q10 What is hyper parameter tuning in machine learning and why it is done?

Ans - Hyperparameter tuning in machine learning refers to the process of finding the optimal values of hyperparameters that are used to control the learning process of a machine learning algorithm.

Hyperparameters are parameters that cannot be learned from the data, but instead need to be set before the learning process begins. They control the behavior of the algorithm and can have a significant impact on its performance. Examples of hyperparameters include learning rate, regularization parameter, number of hidden layers, and number of trees in a random forest.

The process of hyperparameter tuning is done to improve the performance of the machine learning model. By finding the best combination of hyperparameters, the model can be optimized to achieve the highest possible accuracy or other performance metric. This is particularly important when dealing with complex models that have many hyperparameters, as the optimal combination may not be obvious or intuitive.

Q11 What issues can occur if we have a large learning rate in Gradient Descent?

Ans - If the learning rate in Gradient Descent is too large, it can lead to several issues, including:
Overshooting: With a large learning rate, the algorithm can overshoot the minimum point of the cost function, causing it to oscillate or diverge.
Instability: A large learning rate can cause the algorithm to be unstable and produce results that are highly sensitive to the initial starting point and the data.
Slow convergence: The algorithm may take longer to converge to the optimal solution or may not converge at all if the learning rate is too large.

Low accuracy: A large learning rate can lead to inaccurate results due to the algorithm's inability to converge to the optimal solution.

Q12 Can we use Logistic Regression for classification of Non-Linear Data? If not, why?

Ans - Logistic Regression is a linear classification algorithm, meaning it assumes a linear relationship between the input features and the output. Therefore, it may not be suitable for classification tasks involving non-linear data.
If the relationship between the input features and the output is non-linear, using a linear model such as logistic regression may result in poor performance. The model will not be able to capture the complex non-linear relationships in the data, leading to underfitting or high bias.
In such cases, non-linear classification algorithms such as decision trees, random forests, support vector machines (SVMs), or neural networks may be more suitable for the task.
However, in some cases, logistic regression can still be useful for classification tasks involving non-linear data. This can be achieved by using feature engineering techniques such as adding polynomial or interaction terms to the input features, transforming the features using non-linear functions, or using kernel tricks with logistic regression or SVMs. These techniques can help to transform the data into a higher-dimensional space where linear separation is possible.

Q13 Differentiate between Adaboost and Gradient Boosting ?

Ans - Key differences between Adaboost and Gradient Boosting:

Approach to Boosting: Adaboost focuses on adjusting the weights of misclassified samples during each iteration to improve the model's accuracy. Gradient Boosting, on the other hand, focuses on minimizing the residual error of the model at each iteration, by fitting new models to the residuals.

Weak Learners: Adaboost uses a simple base learner such as decision stumps, which are single-level decision trees, while Gradient Boosting can use a more complex base learner such as decision trees or neural networks.

Weighting of Samples: In Adaboost, each sample is assigned a weight based on its classification error, which is used to adjust the sample importance during training. In Gradient Boosting, the sample weights are not adjusted, and the algorithm focuses on fitting the model to the residuals of the previous iteration.

Learning Rate: Adaboost uses a fixed learning rate, which controls the contribution of each weak learner to the final model. Gradient Boosting uses a learning rate, which controls the step size of the optimization, and allows for more fine-grained control over the contribution of each weak learner.

Loss Function: Adaboost uses an exponential loss function, which is sensitive to misclassified samples and can be prone to outliers. Gradient Boosting can use a variety of loss functions such as squared error, absolute error, or log-likelihood, which can be more robust to outliers and better suited for the specific problem.

Q14 What is bias-variance trade off in machine learning?

Ans - Bias-variance trade-off is a fundamental concept in machine learning that refers to the balance between the ability of a model to fit the training data (bias) and its ability to generalize to new, unseen data (variance).

A model with high bias underfits the training data and has a low training error but a high test error, indicating poor generalization. On the other hand, a model with high variance overfits the training data and has a low training error but a high test error, indicating poor generalization.

The goal in machine learning is to find a model that has low bias and low variance, which leads to good generalization and good performance on both the training and test data. This can be achieved by selecting an appropriate model complexity, adjusting the regularization hyperparameters, increasing the size of the training data, or using ensemble methods such as bagging, boosting, or stacking.

Q15 Give short description each of Linear, RBF, Polynomial kernels used in SVM.

Ans – following are the kernel used in SVM

Linear Kernel: The linear kernel is the simplest kernel used in SVMs. It computes the dot product between two feature vectors and is used when the data is linearly separable. The linear kernel is useful when the number of features is large compared to the number of data points.

Radial Basis Function (RBF) Kernel: The RBF kernel is a popular kernel used in SVMs. It measures the similarity between two data points in a higher-dimensional space and is used when the data is not linearly separable. The RBF kernel has a tunable parameter called gamma, which controls the width of the kernel.

Polynomial Kernel: The polynomial kernel is used to handle non-linearly separable data by mapping the data into a higher-dimensional space using a polynomial function. The polynomial kernel has a tunable parameter called degree, which controls the degree of the polynomial function used to map the data.

Worksheet Assignment 5 -SQL

Answers

1. SELECT * FROM Movie;
2. SELECT title FROM Movie ORDER BY runtime DESC LIMIT 1;
3. SELECT title FROM Movie ORDER BY revenue DESC LIMIT 1;
4. SELECT title FROM Movie WHERE revenue/budget = (SELECT MAX(revenue/budget) FROM Movie);
5. SELECT m.title, p.name, g.gender, mc.character_name, mc.cast_order FROM Movie m JOIN Movie_Cast mc ON m.id = mc.movie_id JOIN Person p ON mc.person_id = p.id JOIN Gender g ON mc.gender_id = g.id;
6. SELECT c.name, COUNT(mc.movie_id) AS movies_produced FROM Country c JOIN Movie_Country mc ON c.id = mc.country_id GROUP BY c.name ORDER BY movies_produced DESC LIMIT 1;
7. SELECT id AS genre_id, name AS genre_name FROM Genre;
8. SELECT l.name, COUNT(ml.movie_id) AS movie_count FROM Language l JOIN Movie_Language ml ON l.id = ml.language_id GROUP BY l.name;
9. SELECT m.title, COUNT(DISTINCT mc.person_id) AS crew_members, COUNT(DISTINCT mc.character_name) AS cast_members FROM Movie m JOIN Movie_Cast mc ON m.id = mc.movie_id JOIN Person p ON mc.person_id = p.id GROUP BY m.title;
10. SELECT title FROM Movie ORDER BY popularity DESC LIMIT 10;
11. SELECT title, revenue FROM Movie ORDER BY revenue DESC LIMIT 1 OFFSET 2;
12. SELECT title FROM Movie WHERE status = "Rumoured";
13. SELECT m.title FROM Movie m JOIN Movie_Country mc ON m.id = mc.movie_id WHERE mc.country_id = (SELECT id FROM Country WHERE name = "United States of America") ORDER BY revenue DESC LIMIT 1;
14. SELECT mc.movie_id, pc.name FROM Movie_Company mc JOIN Production_Company pc ON mc.company_id = pc.id;

15. SELECT title FROM Movie ORDER BY budget DESC LIMIT 20;

Worksheet Assignment 5 – Statistics

1. **d) Expected**
2. **c) Frequencies**
3. **c) 6**
4. **b) Chi-squared distribution**
5. **C) F Distribution**
6. **b) Hypothesis**
7. **a) Null Hypothesis**
8. **a) Two tailed**
9. **b) Research Hypothesis**
10. **a) np**