Assignment

Machine Learning

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?

Ans: R-squared is generally a better measure of the goodness of fit for a regression model than the residual sum of squares (RSS).

R-squared, denoted as R^2, is a statistical measure that represents the proportion of the variance for the dependent variable that's explained by the independent variables in the model. It is dimensionless and ranges from 0 to 1, where a value closer to 1 indicates a better fit. R^2 is calculated using the formula:

$R^2 = 1 - \frac{RSS}{TSS}$

where RSS is the residual sum of squares, and TSS is the total sum of squares. TSS represents the total variance in the dependent variable.

The RSS, on the other hand, is the sum of the squared differences between the observed actual outcomes and the outcomes predicted by the regression model. It is calculated as:

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

where y_i is the actual value and $\hat{y}_i$ is the predicted value from the model for the i-th observation.

The reason why R^2 is often preferred over RSS as a measure of goodness of fit is due to its standardized nature:

1. Scalability: R^2 is scale-invariant, meaning it does not change if the scale of the data changes, whereas RSS is affected by the scale of the dependent variable. This makes R^2 a better choice when comparing models fitted on different scales.

2. Interpretability: R^2 has an intuitive interpretation as the proportion of variance explained, which is easier to understand than the sum of squared residuals. An R^2 of 0.75 means that 75% of the variance in the dependent variable is explained by the model, which is a straightforward interpretation.

3. Benchmarking: R^2 provides a clear benchmark. An R^2 of 0 indicates that the model explains none of the variability in the response data around its mean, while an R^2 of 1 indicates that the model explains all the variability.

4. Adjustment for model complexity: Adjusted R^2 takes into account the number of predictors in the model, which helps in assessing whether the addition of a new predictor really improves the model or is just adding complexity without significantly improving the fit.

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: The total sum of squares (TSS) is the sum of squared differences between the observed dependent variables and the overall mean. Think of it as the dispersion of the observed variables around the mean—similar to the variance in descriptive statistics 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.

The residual sum of squares (RSS) measures the level of variance in the error term, or residuals, of a regression model. The smaller the residual sum of squares, the better your model fits your data; the greater the residual sum of squares, the poorer your model fits your data.

TSS = ESS + RSS, where TSS is Total Sum of Squares, ESS is Explained Sum of Squares and RSS is Residual Sum of Squares. The aim of Regression Analysis is explain the variation of dependent variable Y.

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

Ans: Regularization is a critical technique in machine learning to reduce overfitting, enhance model generalization, and manage model complexity. Several regularization techniques are used across different types of models.

1. Lasso Regularization – L1 Regularization
2. Ridge Regularization – L2 Regularization
3. Elastic Net Regularization – L1 and L2 Regularization

4. What is Gini–impurity index?

Ans: Gini Impurity is a measurement used to build Decision Trees to determine how the features of a dataset should split nodes to form the tree. More precisely, the Gini Impurity of a dataset is a number between 0-0.5, which indicates the likelihood of new, random data being misclassified if it were given a random class label according to the class distribution in the dataset.

For example, say you want to build a classifier that determines if someone will default on their credit card. You have some labeled data with features, such as bins for age, income, credit rating, and whether or not each person is a student. To find the best feature for the first split of the tree – the root node – you could calculate how poorly

each feature divided the data into the correct class, default ("yes") or didn't default ("no"). This calculation would measure the **impurity** of the split, and the feature with the lowest impurity would determine the best feature for splitting the current node. This process would continue for each subsequent node using the remaining features.

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

Ans: Yes, Unregularized decision-trees to overfitting. Unregularized Decision trees, by their very nature, are prone to overfitting, especially when they are deep. Overfitting occurs when a model captures noise or fluctuations in the training data that do not represent the underlying data distribution. In the context of decision trees, overfitting can mean creating too many branches based on outliers or anomalies in the training data.

Overfitting happens when any learning processing overly optimizes training set error at the cost test error. While it's possible for training and testing to perform equality well in cross validation, it could be as the result of the data being very close in characteristics, which may not be a huge problem. In the case of decision trees they can learn a training set to a point of high granularity that makes them easily overfit. Allowing a decision tree to split to a granular degree, is the behaviour of this model that makes it prone to learning every point extremely well — to the point of perfect classification — i.e: overfitting.

Regularization techniques add constraints to the learning algorithm, reducing its freedom and, hence, its capacity to overfit. For decision trees, regularization is achieved by controlling the tree's depth and complexity.

6. What is an ensemble technique in machine learning?

Ans: Ensemble techniques in machine learning function much like seeking advice from multiple sources before making a significant decision, such as purchasing a car. Just as you wouldn't rely solely on one opinion, ensemble models combine predictions from multiple base models to enhance overall performance. One popular method, majority voting, aggregates predictions to select the class label by majority. This tutorial explores ensemble learning concepts, including bootstrap sampling to train models on different subsets, the role of predictors in building diverse models, and practical implementation in Python using scikit-learn.

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

Ans: **Bagging** and **Boosting** are two types of **Ensemble Learning**. These two decrease the variance of a single estimate as they combine several estimates from different models. So the result may be a model with higher stability. Let's understand these two terms in a glimpse.

1. **Bagging**: It is a homogeneous weak learners' model that learns from each other independently in parallel and combines them for determining the model average.

2. **Boosting**: It is also a homogeneous weak learners' model but works differently from Bagging. In this model, learners learn sequentially and adaptively to improve model predictions of a learning algorithm.

Differences Between Bagging and Boosting:

S.NO	Bagging	Boosting
1.	The simplest way of combining predictions that belong to the same type.	A way of combining predictions that belong to the different types.
2.	Aim to decrease variance, not bias.	Aim to decrease bias, not variance.
3.	Each model receives equal weight.	Models are weighted according to their performance.
4.	Each model is built independently.	New models are influenced by the performance of previously built models.
5.	Different training data subsets are selected using row sampling with replacement and random sampling methods from the entire training dataset.	Every new subset contains the elements that were misclassified by previous models.
6.	Bagging tries to solve the over-fitting problem.	Boosting tries to reduce bias.
7.	If the classifier is unstable (high variance), then apply bagging.	If the classifier is stable and simple (high bias) the apply boosting.
8.	In this base classifiers are trained parallelly.	In this base classifiers are trained sequentially.
9	Example: The Random forest model uses Bagging.	Example: The AdaBoost uses Boosting techniques

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

Ans: OOB (out-of-bag) errors are an estimate of the performance of a random forest classifier or regressor on unseen data. In scikit-learn, the OOB error can be obtained using the oob_score_ attribute of the random forest classifier or regressor.

The OOB error is computed using the samples that were not included in the training of the individual trees. This is different from the error computed using the usual training and validation sets, which are used to tune the hyperparameters of the random forest.

The OOB error can be useful for evaluating the performance of the random forest on unseen data. It is not always a reliable estimate of the generalization error of the model, but it can provide a useful indication of how well the model is performing.

9. What is K-fold cross-validation?

Ans: Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

If you have a machine learning model and some data, you want to tell if your model can fit. You can split your data into training and test set. Train your model with the training set and evaluate the result with test set. But you evaluated the model only once and you are not sure your good result is by luck or not. You want to evaluate the model multiple times so you can be more confident about the model design.

The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.

Note that k-fold cross-validation is to evaluate the model design, not a particular training. Because you re-trained the model of the same design with different training sets.

The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
    1. Take the group as a hold out or test data set
    2. Take the remaining groups as a training data set
    3. Fit a model on the training set and evaluate it on the test set
    4. Retain the evaluation score and discard the model

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

Ans: Hyper parameter tuning is the process of selecting the optimal values for a machine learning model's hyperparameters. Hyperparameters are settings that control the learning process of the model, such as the learning rate, the number of neurons in a neural network, or the kernel size in a support vector machine. The goal of hyperparameter tuning is to find the values that lead to the best performance on a given task.

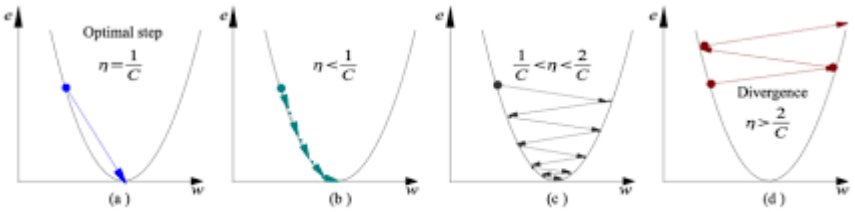
Why it is done:

- Improved model performance
- Reduced overfitting and underfitting
- Enhanced model generalizability
- Optimized resource utilization
- Improved model interpretability

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

Ans: If the learning rate is too high, the algorithm may overshoot the minimum, and if it is too low, the algorithm may take too long to converge. Overfitting: Gradient descent can overfit the training data if the model is too complex or the learning rate is too high.

- If the learning rate is too high, we might OVERSHOOT the minima and keep bouncing without reaching the minima
- If the learning rate is too small, the training might turn out to be too long



- The learning rate is optimal, and the model converges to the minimum.
- The learning rate is too small. It takes more time but converges to the minimum.
- The learning rate is higher than the optimal value. It overshoots but converges ($1/C < \eta < 2/C$).
- The learning rate is very large. It overshoots and diverges, moves away from the minima, and performance decreases in learning.

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

Ans: We Can not use Logistics Regression for Classification of Non-Linear Data. 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 hyper*plane* 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: Gradient boosting produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the ensemble in a stage-wise fashion like other boosting methods do, but it generalizes them by allowing optimization of an arbitrary differentiable loss function.

AdaBoost, short for Adaptive Boosting, is also a boosting technique that combines multiple weak classifiers into a strong one. The key difference is that AdaBoost focuses on training instances that are hard to classify by assigning them higher weights.

Differentiate between Adaboost and Gradient Boosting:

The most significant difference is that gradient boosting minimizes a loss function like MSE or log loss while AdaBoost focuses on instances with high error by adjusting their sample weights adaptively.

Gradient boosting models apply shrinkage to avoid overfitting which AdaBoost does not do. Gradient boosting also performs subsampling of the training instances while AdaBoost uses all instances to train every weak learner.

Overall gradient boosting is more robust to outliers and noise since it equally considers all training instances when optimizing the loss function. AdaBoost is faster but more impacted by dirty data since it fixates on hard examples.

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

Ans: If the algorithm is too simple (hypothesis with linear equation) then it may be on high bias and low variance condition and thus is error-prone. If algorithms fit too complex (hypothesis with high degree equation) then it may be on high variance and low bias. In the latter condition, the new entries will not perform well. Well, there is something between both of these conditions, known as a Trade-off or Bias Variance Trade-off. 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.

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

Ans: Linear SVM: When the data is perfectly linearly separable only then we can use Linear SVM. Perfectly linearly separable means that the data points can be classified into 2 classes by using a single straight line(if 2D)
It is a non-parametric model that works well with non-linear and high-dimensional data. RBF SVM works by mapping the input data into a higher-dimensional feature space, where the classes can be separated by a hyperplane
Polynomial KernelIt is a nonlinear kernel function that employs polynomial functions to transfer the input data into a higher-dimensional feature space. Where x and y are the input feature vectors, c is a constant term, and d is the degree of the polynomial, K(x, y) = (x. y + c)d.