MACHINE LEARNING ASSIGNMENT

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 (R^2) and Residual Sum of Square (RSS) are both measure used to assess the goodness of fit of a Regression model, but they serve different purposes.

i. **R-Squared(R^2)-**
   - **Purposes** – R^2 represents the proportion of the variance in the dependent variable that is explained by the Independent Variable in the model.
   - **Interpretation** – It ranges from 0 to 1 indicates that the model does not explain any variance, and 1 indicates that the model explains all the variance.
   - **Formula – R^2 = 1 - SSR/SST**
     Where, SSR is the Sum of Squared Residuals and SST is the Total Sum of Squares.
   - **Explanation** – A higher R^2 value indicates a better fit as it implies that a larger proportion of the variability in the dependent variable is accounted for by the independent variables.

ii. **Residual Sum of Squares (RSS)-**
   - **Purposes** – RSS measures the total squared difference between the observed values and the values predicted by the model.
   - **Interpretation** – Lower value of RSS indicates a better fit because it means that the model's prediction s are closer to the actual observations.
   - **Formula - RSS=∑i=1n(yi−y^i)^2**
     Where y_i is the observed value, $y^{\wedge}_i$ is the predicted value and n is the number of data points.
   - **Explanation** – Minimising the RSS is a goal in Regression analysis. A smaller RSS implies that the model is doing a better job of fitting the truth.

In a simple terms – R^2 is generally considered a better measures of goodness of fit in Regression compared of RSS

R^2 also known as the co – efficient of determination represents the proportion variation in the dependent variable that is explained by the independent variable in the model. I ranges 0 to 1 where 1 indicates a better fit of the model to the data, suggesting that a larger portion of the variability in the dependent variable is accounted for by thwe independent Variable. R^2 is a preferred because it gives a percentage indicating how well the independent variable explain the variability in the dependent variable, making it easier to interpret and compare models.

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 - In regression analysis, Total Sum of Squares (TSS), Explained Sum of Squares (ESS), and Residual Sum of Squares (RSS) are important concepts that help evaluate the goodness of fit a regression models. These are defined as fellow: -

i. **Total Sum of Squares(TSS) –**
   - TSS represents the total variability in dependent variable (y).
   - It is the sum of the squared difference between each and observed dependent variable and the mean of the dependent variable.
   - TSS = $\sum(yi\text{-}y^{-})^2$, where yi is each observed dependent variable value and y^{-} is the mean of dependent variable.

ii. **Explained Sum of Squares(ESS) –**
   - ESS measures the variability in the dependent variable that is explained by the independent variable in the regression model.
   - It is the Sum of the Squared difference between the predicted values (y^) and the mean of the dependent variable.
   - ESS=$\sum(y^{\wedge}\text{-}\ y^{-})^2$,
     Where $y^{\wedge}_{i}$ is each predicted value from the regression model, and y^{-}_{i} is the mean of the dependent variable.

iii. **Residual Sum of Squares(RSS) –**
   - RSS measure the variability in the dependent that is not explained by the independent variables.
   - It is the sum of the squared difference between each observed dependent variable value and its corresponding predicted value.
   - RSS = $\Sigma(y_i - y^{\wedge}_{i})^2$ where y_i is each observed dependent variable value, and $y^{\wedge}_{i}$ is the predicted value from the Regression model

The relation between three metrics can be expressed by the following equation –

TSS = ESS + RSS

This equation states that the total variability in the dependent variable (TSS) can be decomposed into the explained variability (ESS) and the unexplained variability (RSS) by the regression model. The R^2 (co –efficient of determination) is then calculated as the Ration of ESS to RSS

R^2 = ESS/RSS

R^2 ranges from 0 to 1 where 1 indicates a perfect fit and higher values suggest a better fit of the Regression model to the data.

3. **What is the need of Regularization in Machine Learning?**

Ans. – Regularization refers to techniques that are used to calibrate Machine Learning models in order to minimize the adjusted loss function and prevent overfitting or under fitting. Using Regularization, we can fit our Machine Learning model appropriately on a given test set and have reduce the error in it.

- **What are Overfitting & Under fitting?**

  To train our Machine Learning model, we give it some data to learn from. The process of plotting a series of data points and drawing the best fit line to understand the relationship between the variable is called Data fitting.
  Our model is the best fit when it can find all necessary patterns in our Data and avoid the random data points and unnecessary pattern called Noise.

There are two main types of Regularization techniques –

i. **Lasso Regression (L1) –** Lasso model Regression will omit various co –efficient values to zero so that it can reduce down the error and comes up with a problem of existing Under fitting or Overfitting.
   Lasso (alpha = 0.00001)

ii. **Ridge Regression (L2) –** Ridge Regression will try to reduce down the variation of the co – efficient values which are into the Negative or Positive.
   Ridge (alpha = 0.0001)

iii. **Elastic Net –** It is a combination of both Lasso & Ridge and it will control all the co – efficient values for the minimum reduction as well as omitting of the co – efficient values. So it'll be settling down all these co – efficient on its own to make the prediction high and just to reduce the error.

4. What is Gini – Impurity Index?

Ans. – The Gini – Impurity Index is a measures used in Decision Tree algorithm, particularly in the context of binary classification tasks to evaluate the impurity or disorder of a set of data points. It is a criterion for splitting nodes in a Decision Tree.

Gini (D) = 1 - $\Sigma^{c}_{i=1}(p_i)^2$

Where,

- D is the set of data points in a node
- C is the number of classes in the classification
- p_i is the probability of randomly choosing an element of class i from the set D.

The Gini – Impurity ranges from 0 to 1 where,

- 0 indicates perfect purity (all element belong to the same class)
- 1 indicates maximum impurity(elements are evenly distributed across all classes)

In Decision tree Algorithms, when selecting a feature to split a node the one that minimize the weighted average of Gini Impurity for the resulting child nodes is chosen.
The idea is to reduce impurity at each split, ultimately leading to pure leaves where all data points belong to a single class.
Compared to other impurity measures like entropy, the Gini Impurity is computation less intensive and in practice it often leads to similar result in terms of tree Structure and performance. Decision Tree, guided by impurity measures like Gini impurity, recursively split the data into subsets based on the selected features until a stopping criterion is met, creating a tree structure for making prediction on new data.

5. Are Unregularized Decision – Trees prone to overfitting? If yes? Why?

Ans. – Yes, Unregularized decision – trees are prone to overfitting and there are several reasons why this occurs: -

i. **High Variance –**

- Decision Tree can be very flexible and have the capability to fit the training data extremely well, capturing intricate details and noise in the dataset.
- Without any constraints a decision tree can grow deep, creating a complex structure that perfectly fits the training data but may not generalize well to new unseen data.

ii. **Capturing noise –**

- Decision tree have the tendency to capture noise and outliers present in the training data, especially when the tree is allowed to grow without any limitations.
- Noise or random fluctuations in the training data may not be representative of the underlying pattern in the true relationship between features and the target variable.

iii. **Overly Complex Trees –**

- Unregularized decision trees may become overly complex, with many branches and leaves, leading to a model that is too specific to the training data.
- The complexity of the trees increases the likelihood of fitting the noise in the data rather than capturing the true underlying patterns.

iv. **Minimum Split Size –**

- Setting a minimum number of data points required to perform a split helps control the granularity of t5he tree. Small splits may capture noise, so enforcing a minimum split size can prevent overfitting.

v. **Maximum Depth –**

- Limiting the maximum depth of the tree restricts its growth. This can be an effective way to prevent the tree from becoming overly complex and overfitting the training data.

6. What is an Ensemble Technique in Machine Learning?

Ans. – Ensemble Technique are those that whenever we do have larger dataset, we do make the chance of dataset or subject of a big dataset and these subsets are learned by individual models and these individual models output can be mean accuracy.

These number of subsets could be grown at any larger number here.

Ensemble Means – Assembling (Assemble of different subsets into one unit, different models into one unit)

Ensemble Techniques divided into two types: -

i. Homogeneous
ii. Heterogeneous

- **Homogeneous** methods divided into two types –
  i. Bagging (parallel Technique)
  ii. Boosting (Sequential Techniques)

Bagging (Parallel Tech.) – That often considers homogeneous weak learners, learn them independently from each other in parallel and combines them following some kind of deterministing averaging process.

In Bagging Techniques there is the use of Random Forest Classifier, Extra Tree.

Boosting (Sequential Tech.) – The dataset is being learned in sequential order by the different models. These models are basically DTC (Decision Tree Classifier) and at the end by learning from end models here all these dataset. So here we'll be getting the Final Accuracy or Output.

In Boosting Tech. there is the use of Gradient Boosting, ADA Boost.

- **Heterogeneous –**

  Voting Classifier works for Heterogeneous model means it can take KNN, DTC, SVC, etc. together and make them as one and then it given the dataset inside the Voting Classifier. Voting Classifier divided into two types –

  i. Hard Voting (Majority Case)
  ii. Soft Voting (Probability)

7. What is the difference between Bagging & Boosting Techniques?

Ans. – Bagging and Boosting are both Ensemble Techniques, but they differ in how they build and combine individual models.

- **Bagging (Parallel) –**

  Idea – Create multiple instances of the same model, each trained on a different subset of the training data.

  Training Process – Randomly sample subsets of the data with replacement and train a model on each subset independently.

  Model Independence – The models are trained independently of each other.

  Combining Predictions – Combine the predictions of individual model by averaging (for Regression) or taking a majority vote (for Classification).

  Example – Random Forest and Extra tree is a popular algorithm that uses Bagging, specially applied to Decision Trees.

- **Boosting (Sequential)** –

  Idea – Sequentially train a series of weak learners (models that perform slightly better than random chance) and give more weight to misclassified instances in subsequent iteration.

  Training Process – Train each model in sequences, adjusting the weights of instances to focus on previously misclassified ones.

  Model Dependence – Each model depends on the success of the previous ones: They are trained in a series, trying to correct made by earlier models

  Example – Ada Boost, Gradient Boosting are popular Boosting Techniques.

In Summary – Bagging creates diverse model independently by training on different subsets of the data and combines them by averaging or majority voting.
Boosting builds a series of models sequentially with each model focusing on correcting errors made by the previous ones and combines them by assigning different weight to their predictions.
Both Bagging & Boosting aim to improve the overall performance and robustness of a model by leveraging the strength of multiple models.

8. **What is out – of – Bag error in Random Forest?**

Ans. – In a Random Forest algorithm the out – of – Bag (OOB) error is an estimate of the model's performance on unseen data, and it is calculated without the need for a seprate validation set.

i. **Bootstrap Sampling** - In Random Forest, each decision tree is trained on a random bootstrap sample of the original dataset. This means that some data points are included in the training set multiple times, while other may not be included at all.
ii. **OOB Instances** – For each tree in the forest there is a set of data points that were not included in its training sample. These data points are referred to as the OOB instances for that trees.
iii. **Calculating OOB Error** – After training the Random Forest, each tree can be evaluated on its corresponding OOB Instances. The predictions made on these instances are then compared to the tree labels to calculate the error of that specific tree.

The OOB error is particularly useful for tuning Hyperparameter, assessing model performance, and gaining insights into how well Random Forest is likely to generalize to new unseen data.

9. **What is K – Fold Cross-Validation?**

Ans. – K- Fold Cross-Validation is one of the type of cross – validation along with K –Fold there is 2 more types of Cross – Validation. They are –

i. Hoblont Cross – validation
ii. Loo Cross – Validation(Leave One – Out CV)

'K' means - how many number of segments we have to do for the dataset. So in a very general terms. Suppose I'm taking 0 – 300 rows are with me and K = 3 so entire dataset will be divided into 3 chunks.

In K – Fold CV, the original dataset is equally partitioned into K subparts or field. Out of the K – Folds or groups for each iteration. One group is selected as validation data and the remaining (K = 1) groups are selected as training data. The process is repeated for K – times until each group is treated as validation and remaining as training data.

```
from sklearn.model_selection import KFold
kfold=KFold(5)
score = cross_val_score(svc,x,y,cv=kfold)
print(score)
print(score.mean())
print(score.std())
```

10. What is hyper parameter tuning in Machine Learning and why it is done?

Ans. – Hyperparameter tuning also known as hyperparameter optimization, is the process of finding the optimal set of hyperparameter from a Machine Learning model. Hyperparameter are configuration settings external to the model that cannot be learned from the training data but significantly impact the model's performance.
Examples – It include learning Rate, Regularization strength, the number of layers in a neutral network and the number of trees in a Random Forest.

We can make all the testing of different parametric value automatically by the help of Hyperparameter tuning with the help of Grid Search CV for this we need to make the model.

```
Model = svc()
Dict1={kernel:['rbf','poly','linear']}
gv=GridSearchCV(estimator=model,parameter=Dict1)
gv.fit (x,y)
```

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

Ans. – Gradient Descent is an iterative optimization algorithm for finding the local minimum function. Gradient Descent is simply used to find the values of a function's parameters (co – efficient) that minimize a cost function as far as possible.

Gradient Descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. In machine Learning we use gradient descent to update the parameter of our model. Parameter refers to co – efficient in linear Regression and weight in neutrals network.

A Gradient measures how much the output of a function changes if you can change the input little bit.

Large Learning Rate can occur issue: -

1. **Overshooting the Minimum:**
   - A large learning rate can cause the algorithm to take excessively large steps during each iteration. This may lead to overshooting the minimum of the cost function and oscillating around the optimal point without converging.
2. **Divergence:**
   - If the learning rate is too high, the updates to the model parameters may become so large that the optimization process diverges. The parameters may oscillate or move away from the optimal values instead of converging.
3. **Instability and Unstable Solutions:**
   - Large learning rates can result in unstable and unpredictable behaviour. The optimization process may exhibit erratic movements, making it difficult to find a stable solution.

Its common practice to perform a hyperparameter tuning process, including trying different learning rates and monitoring the optimization process. Techniques like learning rate schedules or adaptive learning rate methods dynamically adjust the learning rate during training, which can be more effective than using a fixed, large learning rate.

In summary, a large learning rate in Gradient Descent can lead to convergence problems, overshooting, and instability. It's crucial to choose an appropriate learning rate to ensure stable and efficient optimization of the model parameters.

12. 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 log-odds of the binary outcome. In its basic form, Logistic Regression is designed for linearly separable data, and it may not perform well when faced with non-linear relationships between features and the target variable.

If the decision boundary in the data is non-linear, Logistic Regression might struggle to capture the complexity of the relationship. In situations where the classes are not linearly separable, Logistic Regression may misclassify instances or provide suboptimal results.

However, there are ways to extend Logistic Regression to handle non-linear data:

1. **Feature Engineering:**
    - One approach is to engineer new features by transforming the existing ones. For instance, adding polynomial features or interaction terms can help capture non-linear relationships.
2. **Kernelized Logistic Regression:**
    - Another technique involves using kernelized versions of Logistic Regression, similar to what is done in Support Vector Machines (SVMs). By applying a kernel trick, Logistic Regression can be adapted to work in higher-dimensional feature spaces, potentially capturing non-linear decision boundaries.
3. **Ensemble Methods:**
    - Ensemble methods, such as Random Forests or Gradient Boosting, are capable of handling non-linear relationships naturally. These algorithms build multiple weak learners and combine their predictions, allowing them to capture complex patterns in the data.

While it is possible to make Logistic Regression more flexible and suitable for non-linear data through the aforementioned techniques, in some cases, other algorithms specifically designed for non-linear relationships (e.g., decision trees, support vector machines with non-linear kernels, neural networks) may be more appropriate and efficient.

In summary, while Logistic Regression is a powerful algorithm for linearly separable data, it might not be the best choice for highly non-linear datasets. Feature engineering and extensions of the algorithm can be applied to handle non-linear relationships, but other methods may be more suitable for complex, non-linear classification problems.

13. Differentiate between Adaboost and Gradient Boosting.

Ans. - Adaboost (Adaptive Boosting) and Gradient Boosting are both ensemble techniques used for boosting in machine learning, but they differ in certain aspects. Here are the key differences between Adaboost and Gradient Boosting:

1. **Objective:**
    - **Adaboost:** The primary objective of Adaboost is to boost the performance of weak learners (usually decision trees) by assigning weights to data points and adjusting them at each iteration based on misclassification errors. It combines multiple weak learners to create a strong learner.
    - **Gradient Boosting:** Gradient Boosting, on the other hand, aims to minimize the residual errors of the model at each iteration. It builds a series of weak learners sequentially, with each one focusing on the errors made by the previous models.
2. **Weighting of Data Points:**
    - **Adaboost:** Adaboost assigns weights to data points, and the weights are adjusted based on the misclassification errors of the weak learners. Instances that are misclassified receive higher weights in subsequent iterations.
    - **Gradient Boosting:** In Gradient Boosting, the focus is on minimizing the residuals (errors) of the model. Each new weak learner is trained to fit the residuals of the combined model from the previous iterations.
3. **Model Fitting:**
    - **Adaboost:** Adaboost fits a series of weak learners (often shallow decision trees) sequentially. Each new weak learner corrects the errors made by the previous ones.
    - **Gradient Boosting:** Gradient Boosting also fits a series of weak learners sequentially, but each one is trained to minimize the gradient of the loss function with respect to the model's prediction. Common weak learners are decision trees.
4. **Combining Weak Learners:**
    - **Adaboost:** The weak learners in Adaboost are combined through a weighted sum, where the weight of each learner is determined by its performance in classifying instances. More accurate learners receive higher weights.
    - **Gradient Boosting:** The weak learners in Gradient Boosting are combined by summing their predictions, each multiplied by a learning rate. The learning rate controls the contribution of each weak learner to the final model.

5. **Learning Rate:**
    - **Adaboost:** Adaboost does not typically use a learning rate. Instead, it assigns different weights to weak learners based on their performance.

- **Gradient Boosting:** Gradient Boosting uses a learning rate to control the step size during optimization. Lower learning rates can improve the stability of the algorithm.

6. **Handling Outliers:**
    - **Adaboost:** Adaboost can be sensitive to outliers because it assigns higher weights to misclassified instances. Outliers may receive increasing emphasis in subsequent iterations.
    - **Gradient Boosting:** Gradient Boosting tends to be more robust to outliers due to the nature of minimizing residuals.

Both Adaboost and Gradient Boosting are powerful algorithms and have their strengths in different scenarios. The choice between them depends on the characteristics of the data and the problem at hand.

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

Ans. – Certain times when we train a model that model start taking self – assumption values. Generally model will be trying to make the best fit line.
In this way, where it is learning maximum data points so best fit line it is going to make it but certain time model does not take those data points.
Model is just saying bypassing the data
When data point bypassing the model is called Underfitting
And certain time the model takes all the data learns all the data this situation is known as an overfit.
In other words,
A Bias occurs when an algorithm has limited flexibility to learn from data. Such model pay very little attention to the training data and oversimplify the model therefore the validation error or prediction error and training error follow similar trends. Such models always lead to a high error on training and test data. High Bias causes underfitting in our model.
Variance defines the algorithm's sensitivity to specific sets of data. A model with a high variance pays a lot of attention to training data does not generalize therefore the validation error or prediction error are far apart from each other. Such model usually perform very well on training data but have high error rates on test data. High Variance causes overfitting in our model.
An Optimal model is one in which the model is sensitive to the pattern in our model, but at the same time can generalize to new data. This happens when Bias and Variance are both optimal. We call this Bias – Variance trade-off and we can achieve it in over or under fitted models by using Regression.

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

Ans. - Support Vector Machines (SVMs) are powerful supervised learning models used for classification and regression tasks. Kernels in SVMs are functions that enable the algorithm to operate in a higher-dimensional space without explicitly calculating the new feature vectors. Here are short descriptions of three commonly used kernels in SVMs:

1. **Linear Kernel:**
   - **Description:** The linear kernel is the simplest and most basic kernel. It computes the dot product between two feature vectors in the input space, effectively implementing a linear decision boundary.
   - **Use Case:** Suitable for linearly separable datasets where classes can be divided by a straight line or hyperplane.
2. **RBF (Radial Basis Function) Kernel:**
   - **Description:** The RBF kernel, also known as the Gaussian kernel, maps input data into a high-dimensional space using a radial basis function. It measures the similarity between data points in the transformed space based on the Euclidean distance.
   - **Use Case:** Effective for capturing non-linear relationships and handling complex decision boundaries. It is widely used and can adapt to various data distributions.
3. **Polynomial Kernel:**
   - **Description:** The polynomial kernel raises the dot product of two vectors to a specified power, introducing non-linearity into the decision function. The degree parameter controls the order of the polynomial.
   - **Use Case:** Suitable for problems where a non-linear decision boundary is required, and the degree parameter influences the complexity of the decision function. Higher degrees allow the model to capture more intricate patterns but may increase the risk of overfitting.

In summary:

- **Linear Kernel:** Suitable for linearly separable data with a simple decision boundary.
- **RBF (Gaussian) Kernel:** Effective for capturing complex, non-linear relationships and handling various data distributions.
- **Polynomial Kernel:** Useful for introducing non-linearity into the decision function, with the degree parameter controlling the complexity of the model.