MACHINE LEARNING- WORKSHEET 5

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-

1. **R-square:** R-square is a measure of variance for dependent variables. That is variance in the output that is explained by the small change in input.

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

   - The value of *R-square* is always between 0 (0%) and 1 (100%). The bigger the value better the fit.

2. **The residual sum of squares (RSS):** RSS is a statistical technique used to measure the amount of variance in a data set that is not explained by a regression model itself. Instead, it estimates the variance in the residuals, or error term.

   $$\boxed{RSS = \sum_{i}^{N} \left(y_i - y_{pred_i}\right)^2}$$

   - 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.
   - A value of zero means your model is a perfect fit.

Thus, we can interpret, R-squared is a better measure of goodness of fit model in regression models, as the variances are explained by the R2-Score, however, RSS represents the variances or errors that are not explained.

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-

Sum of squares (SS) is a statistical tool that is used to identify the dispersion of data as well as how well the data can fit the model in regression analysis. The sum of squares got its name because it is calculated by finding the sum of the squared differences.

Types of Sum of Squares

In regression analysis, the three main types of sum of squares are the Total Sum of Squares(TSS), Explained Sum of Squares(ESS), and Residual Sum of Squares(RSS).

1. Total sum of squares (TSS): The total sum of squares is a variation of the values of a dependent variable from the sample mean of the dependent variable. Essentially, the total sum of squares quantifies the total variation in a sample.

It can be determined using the following formula:

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

Where:
yi – the value in a sample
ȳ – the mean value of a sample

2. Explained Sum of Squares (ESS) (also known as the Sum of Squares due to Regression (SSR) or Model Sum of Squares): The Explained Sum of Squares describes how well a regression model represents the modelled data. A higher regression sum of squares indicates that the model does not fit the data well.

The formula for calculating the explained sum of squares is:

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

Where:
ŷi – the value estimated by the regression line
ȳ – the mean value of a sample

3. Residual sum of squares (RSS) (also known as the sum of squared errors(SSE) of prediction): The residual sum of squares essentially measures the variation of modelling errors. In other words, it depicts how the variation in the dependent variable in a regression model cannot be explained by the model. Generally, a lower residual sum of squares indicates that the regression model can better explain the data, while a higher residual sum of squares indicates that the model poorly explains the data.

The residual sum of squares can be found using the formula below:

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

Where:
yi – the observed value
ŷi – the value estimated by the regression line

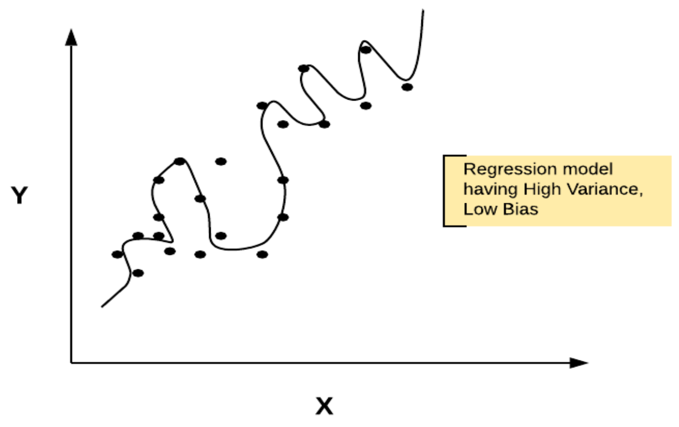
The relationship between the three types of sum of squares can be summarized by the following equation:

TSS = ESS + RSS

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

Ans-
Regularization is a technique used to reduce the errors by fitting the function appropriately on the given training set and avoid overfitting.

Overfitting happens when a machine learning model fits tightly to the training data and tries to learn all the details in the data; in this case, the model cannot generalize well to the unseen data. In other words, a high variance machine learning model captures all the details of the training data along with the existing noise in the data.



Regularization means restricting a model to avoid overfitting by shrinking the coefficient estimates to zero. When a model suffers from overfitting, we should control the model's complexity. Technically, regularization avoids overfitting by adding a penalty to the model's loss function:

Regularization = Loss Function + Penalty

There are three commonly used regularization techniques to control the complexity of machine learning models:

- L2 regularization (Lasso)
- L1 regularization (Ridge)
- Elastic Net

4. What is Gini–impurity index?

Ans-
Gini Index is one of the ways of splitting a decision tree. The Entropy and Information Gain method focuses on purity and impurity in a node. The Gini Index or Impurity measures the probability for a random instance being misclassified when chosen randomly. i.e., Gini Impurity tells us what is the probability of misclassifying an observation. The lower the Gini Index, the better the lower the likelihood of misclassification.

The degree of Gini Index varies between 0 and 1,

where,
'0' denotes that all elements belong to a certain class or there exists only one class (pure), and
'1' denotes that the elements are randomly distributed across various classes (impure).
A Gini Index of '0.5 'denotes equally distributed elements into some classes**.**
The formula for Gini Index:

$$Gini = 1 - \sum_{i=1}^{j} P(i)^2$$

Where:
j represents the no. of classes in the target variable
P(i) represents the ratio of Pass/Total no. of observations in node.

While building the decision tree, it is better to choose the attribute/feature with the least Gini index as the root node.

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

Ans-
Overfit condition arises when the model memorizes the noise of the training data and fails to capture important patterns. And therefore, we can say, yes, unregularized decision-trees prone to overfitting. Because, If the decision tree is allowed to train to its full strength, the model will overfit the training data.
In order to avoid overfitting, we have to put in a criterion to stop splitting the nodes beyond a point. This can be done by pruning the tree, limit max depth of trees, using ensemble techniques and hyperparameter tuning.

6. What is an ensemble technique in machine learning?

Ans-
An ensemble method is a technique which uses multiple independent similar or different models/weak learners to produce one optimal predictive model.
For e.g.: A random forest is an ensemble of multiple decision trees.

We can mention three major kinds of meta-algorithms that aims at combining weak learners:

- **Bagging:** that often considers homogeneous weak learners, learns them independently from each other in parallel and combines them following some kind of deterministic averaging process.

- **Boosting:** that often considers homogeneous weak learners, learns them sequentially in a very adaptative way (a base model depends on the previous ones) and combines them following a deterministic strategy.
- **Stacking:** that often considers heterogeneous weak learners, learns them in parallel and combines them by training a meta-model to output a prediction based on the different weak models' predictions.

Very roughly, we can say that bagging will mainly focus at getting an ensemble model with less variance than its components whereas boosting and stacking will mainly try to produce strong models less biased than their components (even if variance can also be reduced).

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

Ans-
Bagging:
1. In bagging methods, several instances of the same base model are trained in parallel on different bootstrap samples and then aggregated in some kind of "averaging" process.
2. Individual trees/models are independent of each other.
3. There is no concept of learning in bagging.
4. Learning rate is not used as hyper parameter in Bagging methods as the trees are independent of each other.
5. Helps reduce variance
6. Examples: Random Forest, Extra Tree

Boosting:
1. In boosting methods, several instances of the same base model are trained sequentially such that, at each iteration, the way to train the current weak learner depends on the previous weak learners and more especially on how they are performing on the data.
2. Individual trees are not independent of each other.
3. In boosting, each of the trees will learn from the mistakes of the previous tree and try to minimize the residual error as it keep moving forward.
4. Learning rate is used as hyper parameter in Boosting methods as each of the trees learns from previous iterations.
5. Helps reduce both bias and variance
6. Examples: Gradient Boosting, ADABoost, XGBooost

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.

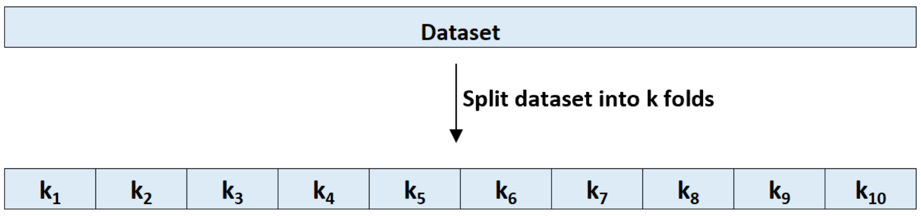
9. What is K-fold cross-validation?

Ans-
K-fold cross-validation is a technique used in machine learning to evaluate the performance of a model on unseen data. It involves dividing the available data into multiple folds(k) and using one of these folds as a validation set, and training the model on the remaining folds. This process is repeated multiple times, each time using a different fold as the validation set. Finally, the results from each validation step are averaged to produce a more robust estimate of the model's performance.

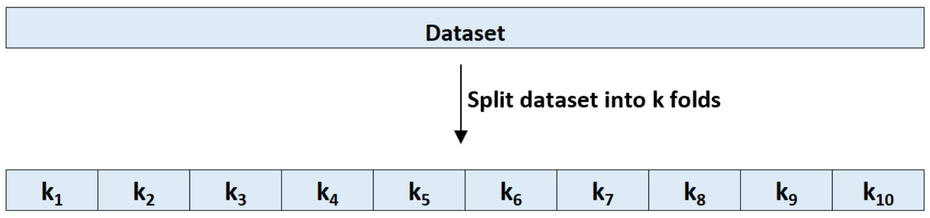
The main purpose of cross validation is to prevent overfitting, which occurs when a model is trained too well on the training data and performs poorly on new, unseen data. By evaluating the model on multiple validation sets, cross validation provides a more realistic estimate of the model's generalization performance, i.e., its ability to perform well on new, unseen data.

K-fold cross-validation uses the following approach to evaluate a model:

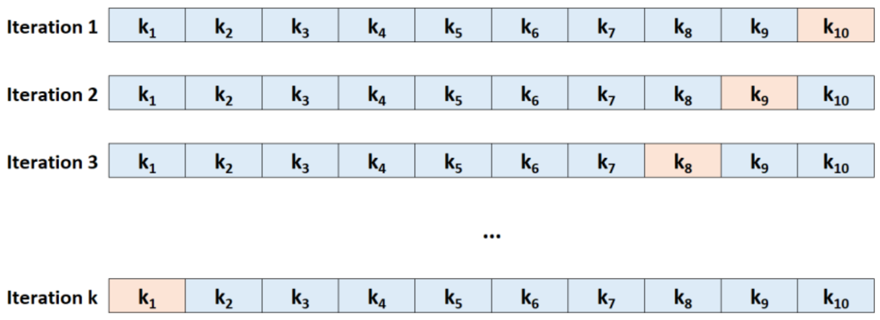
Step 1: Randomly divide a dataset into k groups, or "folds", of roughly equal size.



Step 2: Choose one of the folds to be the holdout set. Fit the model on the remaining k-1 folds. Calculate the test MSE on the observations in the fold that was held out.


Dataset
Split dataset into k folds
k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_8 k_9 k_{10}

Step 3: Repeat this process k times, using a different set each time as the holdout set.


Iteration 1
Iteration 2
Iteration 3
...
Iteration k
k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_8 k_9 k_{10}

Step 4: Calculate the overall test MSE to be the average of the k test MSE's.

- In practice, we typically choose to use between 5 and 10 folds.

10. What is hyper parameter tuning in machine learning and why it is done?
Ans-
Certain models can have many hyperparameters and finding the best combination of parameters can be treated as a search problem. Hyperparameter tuning is basically referred to as pulling the best parameters of the model.

Steps to perform hyper parameter tuning:
1. Select the right type of model.
2. Review the list of parameters of the model and build the HP space
3. Finding the methods for searching the hyperparameter space
4. Applying the cross-validation scheme approach
5. Assess the model score to evaluate the model

The two best strategies for Hyperparameter tuning are:
1. **GridSearchCV:** That exhaustively considers all parameter combinations
2. **RandomizedSearchCV**: That can sample a given number of candidates from a parameter space with a specified distribution.

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

Ans-
A learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck as it takes huge amount of time.

When the learning rate is too large, gradient descent can inadvertently increase the training error rather than decrease it. At extremes, a learning rate that is too large will result in weight updates that will be too large and the performance of the model (such as its loss on the training dataset) will oscillate over training epochs. Oscillating performance is said to be caused by weights that diverge.
The learning rate can seen as step size, η. If the step size η is too large, it can (plausibly) "jump over" the minima we are trying to reach, ie. we overshoot. This can lead to osculation's around the minimum or in some cases to outright divergence.

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

Ans-
Logistic regression is known and used as a linear classifier. It is used to come up with a hyperplane 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.

But in cases, where we suspect the decision boundary to be nonlinear, it may make sense to formulate logistic regression with a nonlinear model & evaluate how much better we can do.

13. Differentiate between Adaboost and Gradient Boosting.

Ans-
Adaboost:

1. AdaBoost or Adaptive Boosting method automatically adjusts its parameters to the data based on the actual performance in the current iteration. Meaning, both the weights for re-weighting the data and the weights for the final aggregation are re-computed iteratively.
2. In practice, this boosting technique is used with simple classification trees or stumps as base-learners, which resulted in improved performance compared to the classification by one tree or other single base-learner.
3. Each classifier has different weights assigned to the final prediction based on its performance.
4. AdaBoost is the first designed boosting algorithm with a particular loss function and is less flexible than Gradient Boosting.
5. The exponential loss provides maximum weights for the samples which are fitted in worse conditions.

Gradient Boosting:

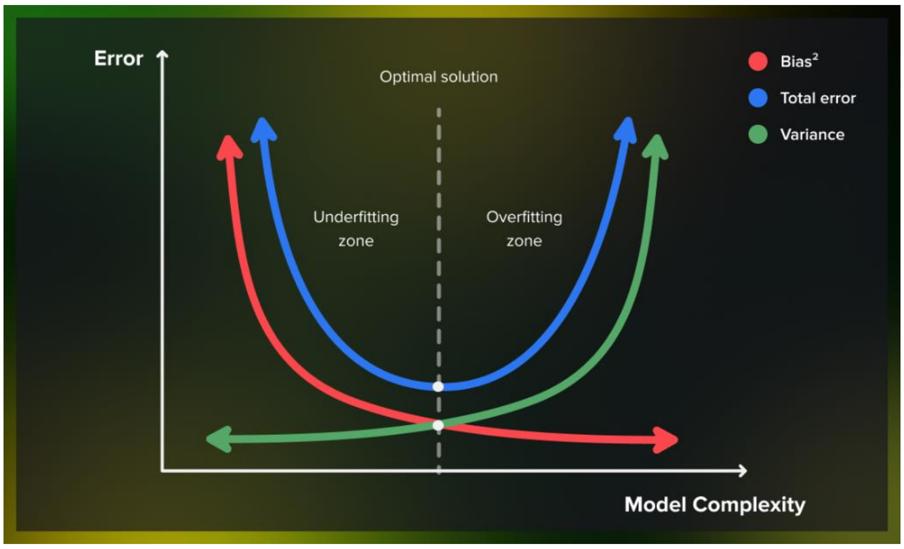
1. Gradient Boost is a robust machine learning algorithm made up of Gradient descent and Boosting. The word 'gradient' implies that you can have two or more derivatives of the same function. Gradient Boosting has three main components: additive model, loss function and a weak learner.
2. The trees with week learners are constructed using a greedy algorithm based on split points and purity scores. The trees are grown to a greater depth usually ranging from 8 to 32 terminal nodes.
3. All classifiers are weighed equally and their predictive capacity is restricted with learning rate to increase accuracy.
4. Gradient Boosting is a generic algorithm that assists in searching the approximate solutions to the additive modelling problem. This makes Gradient Boosting more flexible than AdaBoost.
5. Gradient boosting cut down the error components to provide clear explanations and its concepts are easier to adapt and understand.

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

Ans-

The bias-variance trade off refers to the trade-off that takes place when we choose to lower bias which typically increases variance, or lower variance which typically increases bias.

The following chart offers a way to visualize this trade-off:



We have to find the perfect balance between Bias and Variance. This just ensures that we capture the essential patterns in our model while ignoring the noise present it in. It helps optimize the error in our model and keep it as low as possible.

In practice, the most common way to minimize test MSE is to use cross-validation.

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

Ans-
A kernel is nothing but a measure of similarity between data points. SVM uses a technique called the kernel trick, where, the kernel functions are used to map the original dataset (linear/nonlinear) into a higher dimensional space with view to making it linear dataset. In other words, you can say that it converts non-separable problem to separable problems by adding more dimension to it.

The various kernel functions in SVM are:

1. **Linear Kernel:** Linear Kernel is used when the data is linearly separable, i.e., it can be separated using a single line. It is mostly used when there are a large number of features in a particular dataset. One of the examples where there are a lot of features, is Text Classification, as each alphabet is a new feature.
   - A linear kernel can be used as normal dot product of any two given observations. The product between two vectors is the sum of the multiplication of each pair of input values.

     K(x, xi) = sum(x * xi)

   - Advantages:

     Firstly, training Linear Kernel is faster than any other kernel.
     Secondly, while training, only the optimisation of the C regularization parameter is required. On the other hand, when training with other kernels, there is a need to optimise the gamma parameter which means that performing a Grid Search will usually take more time.
2. **Radial Basis Function Kernel:** RBF kernel is a function whose value depends on the distance from the origin or from some point. RBF can map an input space in infinite dimensional space. RBF is often used for Computer Vision

   K(x,xi) = exp(-gamma * sum((x – xi^2))

   Here, gamma is a parameter, which ranges from 0 to 1. A higher value of gamma will perfectly fit the training dataset, which causes over-fitting.

   Gamma=0.1 is considered to be a good default value. The value of gamma needs to be manually specified in the learning algorithm.

3. **Polynomial Kernel:** A polynomial kernel is a more generalized form of the linear kernel. The polynomial kernel can distinguish curved or nonlinear input space. It can be used for non-linear models and mainly used in natural Language Processing (NLP)

   K(x,xi) = 1 + sum(x * xi)^d

   Where d is the degree of the polynomial. d=1 is similar to the linear transformation.

   The degree needs to be manually specified in the learning algorithm.