MACHINE LEARNING

ASSIGNMENT - 5

Q1 to Q15 are subjective answer type questions, Answer them briefly.

Q. 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. Both R-squared and Residual Sum of Squares (RSS) are measures of goodness of fit in regression analysis, but they capture different aspects of the model's performance.

R-squared (also known as the coefficient of determination) measures the proportion of variation in the dependent variable that is explained by the independent variables in the model. In other words, it indicates how well the model fits the data, with values ranging from 0 to 1. Higher R-squared values indicate a better fit, as they mean that a larger proportion of the variation in the dependent variable is explained by the independent variables in the model.

On the other hand, RSS measures the total sum of squared differences between the actual values of the dependent variable and the predicted values by the model. It represents the amount of unexplained variation in the data, and lower RSS values indicate a better fit, as they mean that the model is able to explain more of the variation in the data.

In investing, R-squared is generally explaind as the percentage of a fund or security's movements that can be explained by movements in a benchmark index.

An R-squared of 100% means that all movements of a security or dependent variable are completely explained by movements in the index or independent variable.

If R-squared values between 100-70% it means strogly correlated.When below 70% it means correlation going to break.

Q. 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. TSS :- The Total SS (TSS or SST) tells you how much variation there is in the **dependent variable.**

Total SS = Σ(Yi – mean of Y)^2.

Note: Sigma (Σ) is a mathematical term for summation or "adding up." It's telling you to add up all the possible results from the rest of the equation.

Sum of squares is a measure of how a data set varies around a central number (like the mean). You might realize by the phrase that you're summing (*adding up*) squares—but squares of what? You'll sometimes see this formula:

$$y = Y - \overline{Y}$$

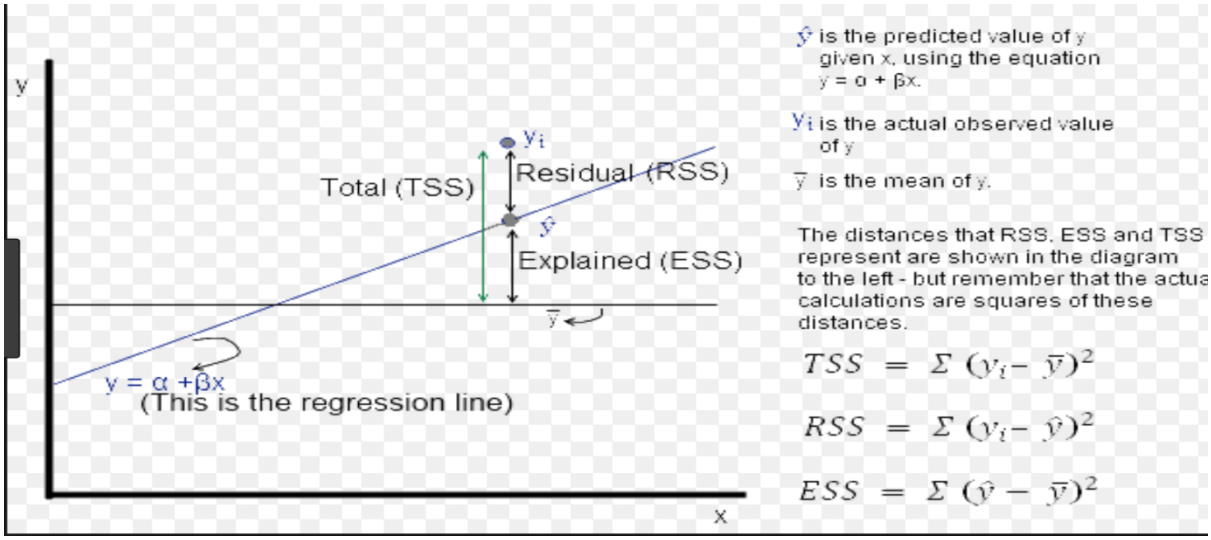
It is a measure of the total variability of the dataset.

$$\Sigma(y - \overline{y})^2 = \Sigma(\hat{y} - \overline{y})^2 + \Sigma(y - \hat{y})^2$$

ESS:-

The Explained SS tells us how much of the variation in the dependent variable your model explained.

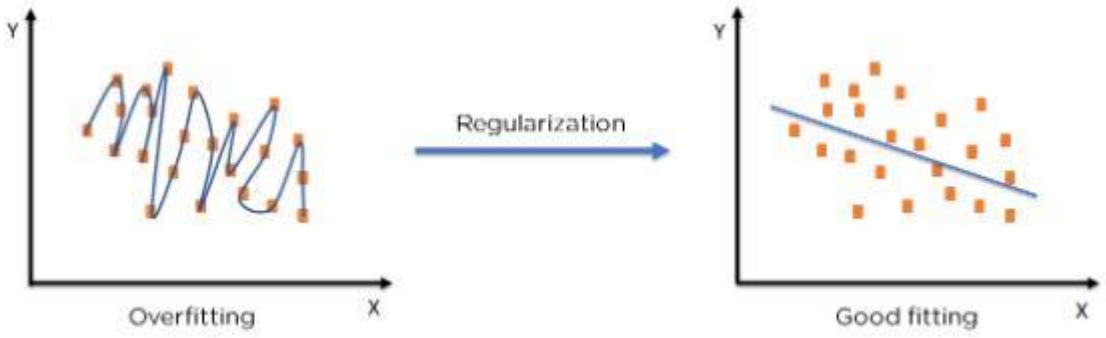
Explained SS = Σ(Y-Hat – mean of Y)^2.



RSS:-In statistics, the residual sum of squares (RSS), also known as the sum of squared residuals (SSR) or the sum of squared errors of prediction (SSE), is the sum of the squares of residuals (deviations of predicted from actual empirical values of data).

Q. 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 underfitting.



Using Regularization, we can fit our machine learning model appropriately on a given test set and hence reduce the errors in it.

Regularization is a technique used to reduce the errors by fitting the function appropriately on the given training set and avoid overfitting.
The commonly used regularization techniques are :

1. L1 regularization
2. L2 regularization
3. Dropout regularization

A regression model which uses **L1 Regularization** technique is called **LASSO(Least Absolute Shrinkage and Selection Operator)** regression.
A regression model that uses **L2 regularization** technique is called **Ridge regression**.
Lasso Regression adds **"absolute value of magnitude"** of coefficient as penalty term to the loss function(L).

Q. 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. The internal working of Gini

impurity is also somewhat similar to the working of entropy in the Decision Tree. In the Decision Tree algorithm, both are used for building the tree by splitting as per the appropriate features but there is quite a difference in the computation of both methods. Gini Impurity of features after splitting can be calculated by using this formula.

Formula: $Gini=1-(p1,p2,p3,p4,p5.....pn)^2$

where:p=is the probability of an object being classified to a particular class.

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.

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

ANS.Yes,Decision trees are prone to overfitting, especially when a tree is particularly deep. This is due to the amount of specificity we look at leading to smaller sample of events that meet the previous assumptions. This small sample could lead to unsound conclusions. This is a disadvantage of Decision tree.

Q. 6. What is an ensemble technique in machine learning?

ANS. Ensemble learning is the process by which multiple models, such as classifiers or experts, are strategically generated and combined to solve a particular computational intelligence problem. Ensemble learning is primarily used to improve the (classification, prediction, function approximation, etc.)

There are many ensemble techniques available but we will discuss about the below two most widely used methods:

1. Bagging

2. Boosting

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

ANS. Differences Between Bagging and Boosting

Sr.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 is built independently.	New models are influenced by the performance of previously built models.
4.	Each model receives equal weight.	Models are weighted according to their performance.
5.	Bagging tries to solve the over-fitting problem.	Boosting tries to reduce bias.
6.	In this base classifiers are trained parallelly.	In this base classifiers are trained sequentially.
7.	Example: The Random forest model uses Bagging.	Example: The AdaBoost uses Boosting techniques

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

ANS. The out-of-bag error is the average error for each predicted outcome calculated using predictions from the trees that do not contain that data point in their respective bootstrap sample. This way, the Random Forest model is constantly being validated while being trained. Let us consider the $^{j\text{th}}$ decision tree DT_j that has been fitted on a subset of the sample data. For every training observation or sample $z_i=(x\ i\ \ yi)$ not in the sample subset of DT_j where xi is the set of features and yi is the target, we use DT_j to predict the outcome oi for xi. The error can easily be computed as $|o_i - y_i|$.
The out-of-bag error is thus the average value of this error across all decision trees.

If there are N rows in the training data set. Then, the probability of not picking a row in a random draw is (N-1)/N

Using sampling-with-replacement the probability of not picking N rows in random draws is $((N-1) / N)^N$

Therefore, about 36.8 % of total training data are available as OOB sample for each DT and hence it can be used for evaluating or validating the random forest model.

Q. 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.

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.

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
4. Summarize the skill of the model using the sample of model evaluation scores
Importantly, each observation in the data sample is assigned to an individual group and stays in that group for the duration of the procedure. This means that each sample is given the opportunity to be used in the hold out set 1 time and used to train the model k-1 times.

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

ANS. Hyperparameter tuning consists of finding a set of optimal hyperparameter values for a learning algorithm while applying this optimized algorithm to any data set. That combination of hyperparameters maximizes the model's performance, minimizing a predefined loss function to produce better results with fewer errors.

A Machine Learning model is defined as a mathematical model with a number of parameters that need to be learned from the data. By training a model with existing data, we are able to fit the model parameters.
However, there is another kind of parameter, known as ***Hyperparameters***, that cannot be directly learned from the regular training process. They are usually fixed before the actual training process begins. These parameters express important properties of the model such as its complexity or how fast it should learn.
Some examples of model hyperparameters include:

1. The penalty in Logistic Regression Classifier i.e. L1 or L2 regularization
2. The learning rate for training a neural network.
3. The C and sigma hyperparameters for support vector machines.
4. The k in k-nearest neighbors.

Models can have many hyperparameters and finding the best combination of parameters can be treated as a search problem. The two best strategies for Hyperparameter tuning are:

- GridSearchCV
- RandomizedSearchCV

GridSearchCV

In GridSearchCV approach, the machine learning model is evaluated for a range of hyperparameter values. This approach is called GridSearchCV, because it searches for the best set of hyperparameters from a grid of hyperparameters values.

RandomizedSearchCV

RandomizedSearchCV solves the drawbacks of GridSearchCV, as it goes through only a fixed number of hyperparameter settings. It moves within the grid in a random fashion to find the best set of hyperparameters. This approach reduces unnecessary computation.

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

ANS. The learning rate can seen as step size, η. As such, gradient descent is taking successive steps in the direction of the minimum. 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 osculations around the minimum or in some cases to outright divergence.

In order for Gradient Descent to work, we must set the learning rate to an appropriate value. This parameter **determines how fast or slow we will move towards the optimal weights**. If the learning rate is very large we will skip the optimal solution. If it is too small we will need too many iterations to converge to the best values. So using a good learning rate is crucial.

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

ANS. 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.

Q. 13. Differentiate between Adaboost and Gradient Boosting.

ANS.

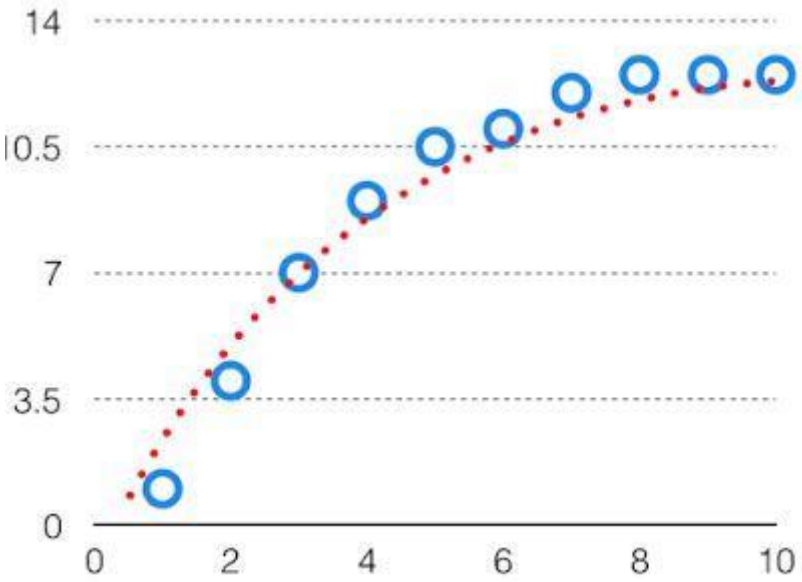
Gradient Boosting	Adaboost
It identifies complex observations by huge residuals calculated in prior iterations	The shift is made by up-weighting the observations that are miscalculated prior
The classifiers are weighted precisely and their prediction capacity is constrained to learning rate and increasing accuracy	Every classifier has different weight assumptions to its final prediction that depend on the performance.
The trees with week learners are constructed using a greedy algorithm based on split	The trees are called decision stumps.

points and purity scores. The trees are grown deeper with eight to thirty-two terminal nodes.	
It develops a tree with help of previous classifier residuals by capturing variances in data. The final prediction depends on the maximum vote of the week learners and is weighted by its accuracy.	It gives values to classifiers by observing determined variance with data. Here all the week learners possess equal weight and it is usually fixed as the rate for learning which is too minimum in magnitude.
Gradient boosting cut down the error components to provide clear explanations and its concepts are easier to adapt and understand	The exponential loss provides maximum weights for the samples which are fitted in worse conditions.
This method trains the learners and depends on reducing the loss functions of that week learner by training the residues of the model	Its focus on training the prior miscalculated observations and it alters the distribution of the dataset to enhance the weight on sample values which are hard for classification

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

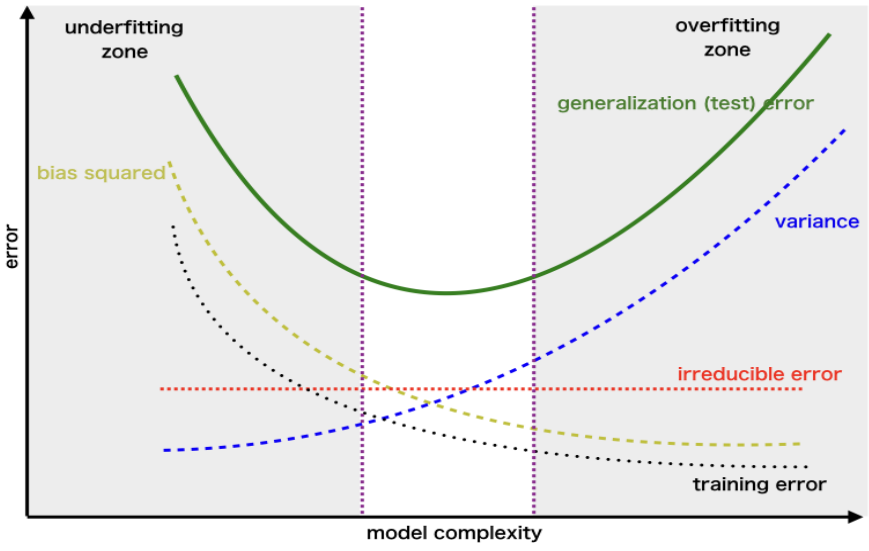
ANS. f the algorithm is too simple (hypothesis with linear eq.) then it may be on high bias and low variance condition and thus is error-prone. If algorithms fit too complex (hypothesis with high degree eq.) then it may be on high variance and low bias. In the latter condition, the new entries will not perform well. Well, there is something between both of these conditions, known as 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. For the graph, the perfect tradeoff will be like.



The best fit will be given by hypothesis on the tradeoff point.

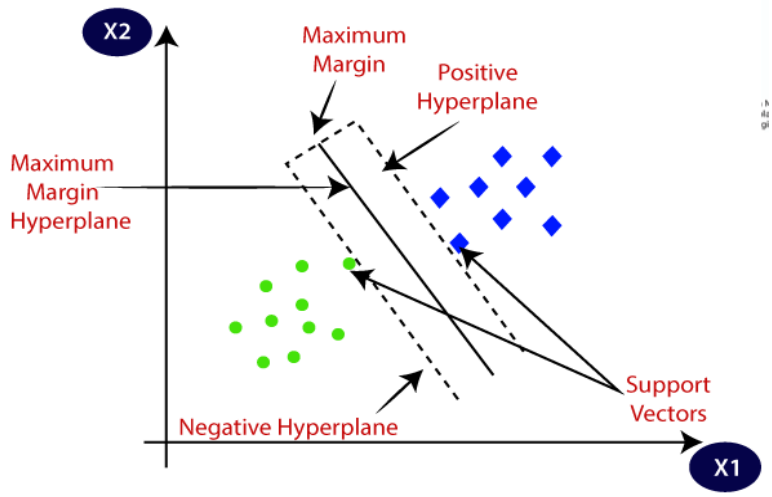
The error to complexity graph to show trade-off is given as –



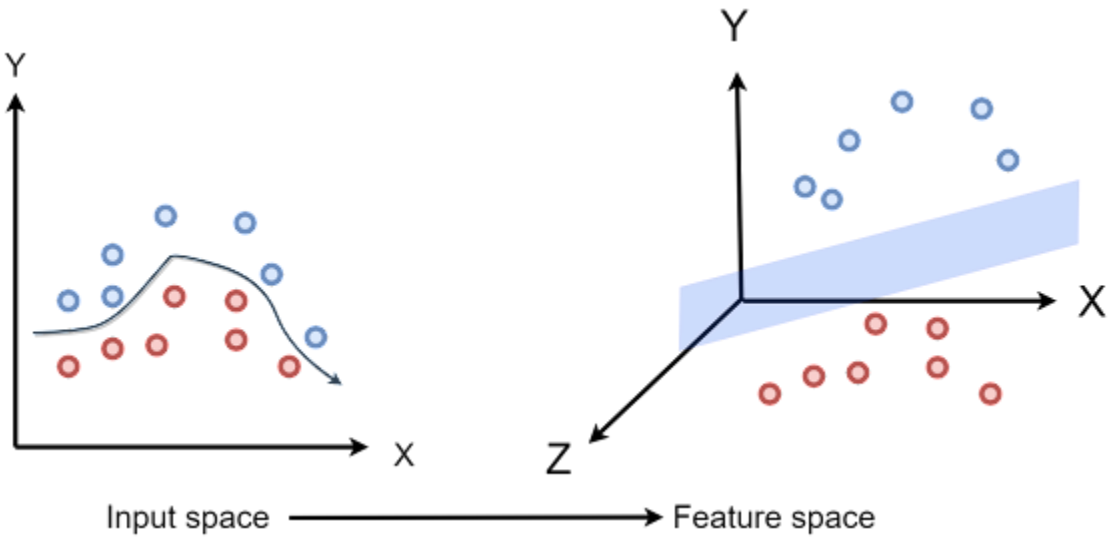
This is referred to as the best point chosen for the training of the algorithm which gives low error in training as well as testing data.

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

ANS. Linear SVM: Linear SVM is **used for linearly separable data**, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.



Unlike linear or polynomial kernels, **RBF is more complex and efficient at the same time** that it can combine multiple polynomial kernels multiple times of different degrees to project the non-linearly separable data into higher dimensional space so that it can be separable using a hyperplane.



In machine learning, the polynomial kernel is a kernel function commonly used with support vector machines (SVMs) and other kernelized models, that represents the similarity of vectors (training samples) in a feature space over polynomials of the original variables, allowing learning of non-linear models.