NAME: Dipesh Ramesh Limaje

INTERNSHIP BATCH : 33

TOPIC: MACHINE LEARNING

SME : Mr. Shwetank Mishra

WORKSHEET NO : 5

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 is a statistical measure that represents the proportion of the variance in the dependent variable that is predictable from the independent variable(s) in a regression model. It ranges from 0 to 1, where 0 means that the model explains none of the variance in the dependent variable, and 1 means that the model explains all of the variance. A high R-squared value indicates that the model fits the data well, while a low R-squared value indicates that the model does not fit the data well.
The formula for R-squared is:

R-squared = 1 - (RSS / TSS)

Where RSS is the residual sum of squares (the sum of the squared differences between the predicted and actual values) and TSS is the total sum of squares (the sum of the squared differences between the actual values and the mean of the actual values).

R-squared is interpretable in the sense that it gives us a percentage of the total variation in the response variable that can be explained by the predictor variable(s) in the model. It gives a quick way to compare different models and to evaluate the model.

On the other hand, Residual Sum of Squares (RSS) is the sum of the squared differences between the predicted and actual values. It can be used to compare different models, but it is affected by the scale of the data and the units of measurement, which can make it difficult to interpret. Additionally, it does not give a standardized value, which makes it hard to compare different models.

To conclude, while both R-squared and RSS can be used to evaluate the fit of a regression model, R-squared is generally considered a better measure because it is standardized, interpretable, and easy to compare different models.

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 In a regression analysis, Total Sum of Squares (TSS), Explained Sum of Squares (ESS), and Residual Sum of Squares (RSS) are used to evaluate the fit of a model.
TSS is the sum of the squared differences between the actual values of the dependent variable and the mean of the dependent variable. It represents the total variability in the dependent variable. The formula for TSS is:

TSS = Σ(Yi - Ymean)^2

where Yi is the actual value of the dependent variable, and Ymean is the mean of the dependent variable.

ESS is the sum of the squared differences between the predicted values of the dependent variable and the mean of the dependent variable. It represents the amount of variability in the dependent variable that is explained by the independent variable(s) in the model. The formula for ESS is:

ESS = Σ(Yi - Ypred)^2

where Yi is the actual value of the dependent variable, and Ypred is the predicted value of the dependent variable from the model.

RSS is the sum of the squared differences between the predicted values of the dependent variable and the actual values of the dependent variable. It represents the amount of variability in the dependent variable that is not explained by the independent variable(s) in the model. The formula for RSS is:

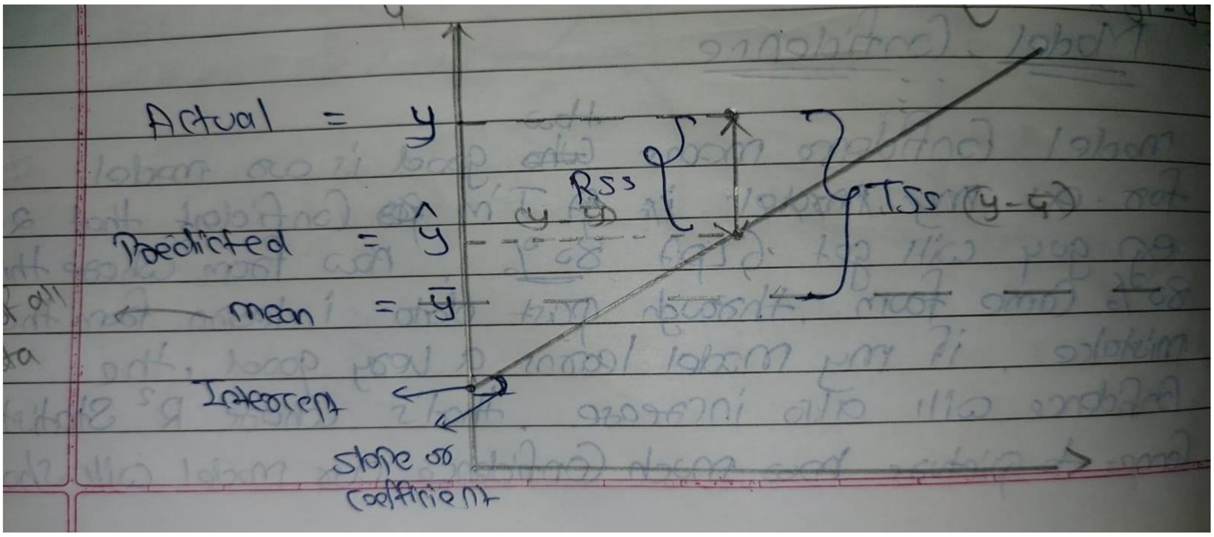
RSS = Σ(Yi - Ypred)^2

where Yi is the actual value of the dependent variable, and Ypred is the predicted value of the dependent variable from the model.

The relationship between TSS, ESS, and RSS is:

TSS = ESS + RSS

This equation states that the total variation in the dependent variable is equal to the variation explained by the model plus the variation not explained by the model.



Q3. What is the need of regularization in machine learning?

ANS Regularization is an important concept in machine learning, as it helps to prevent overfitting and improve the generalizability of a model. Regularization techniques help to reduce the complexity of a model by penalizing high-order coefficients and thus reducing the number of parameters that need to be estimated. As a result, models built with regularization are able to generalize better and make more accurate predictions on unseen data. Regularization also helps to reduce the risk of overfitting, which can lead to poor performance on test data. By using regularization, we can create models that are more accurate, robust and interpretable.

When we use regression model to train some data there is good chance that the model will overfit the given training dataset, regularization help sort this overfitting problem by restricting the degree of freedom.

There are 3-Types of Regularization: 1) LASSO (L1 FORM)
2) RIDGE (L2 FORM)
3) ELASTIONNET

Q4. What is Gini–impurity index?

ANS The Gini impurity index is a measure of the impurity or disorder of a set of items. It is commonly used in decision tree-based algorithms, such as CART (Classification and Regression Trees) and C4.5, to determine the best split point in a dataset.

The Gini impurity index is defined as the probability of misclassifying a randomly chosen item from the set, if the item is classified according to the distribution of the items in the set. The Gini impurity of a set S is defined as:

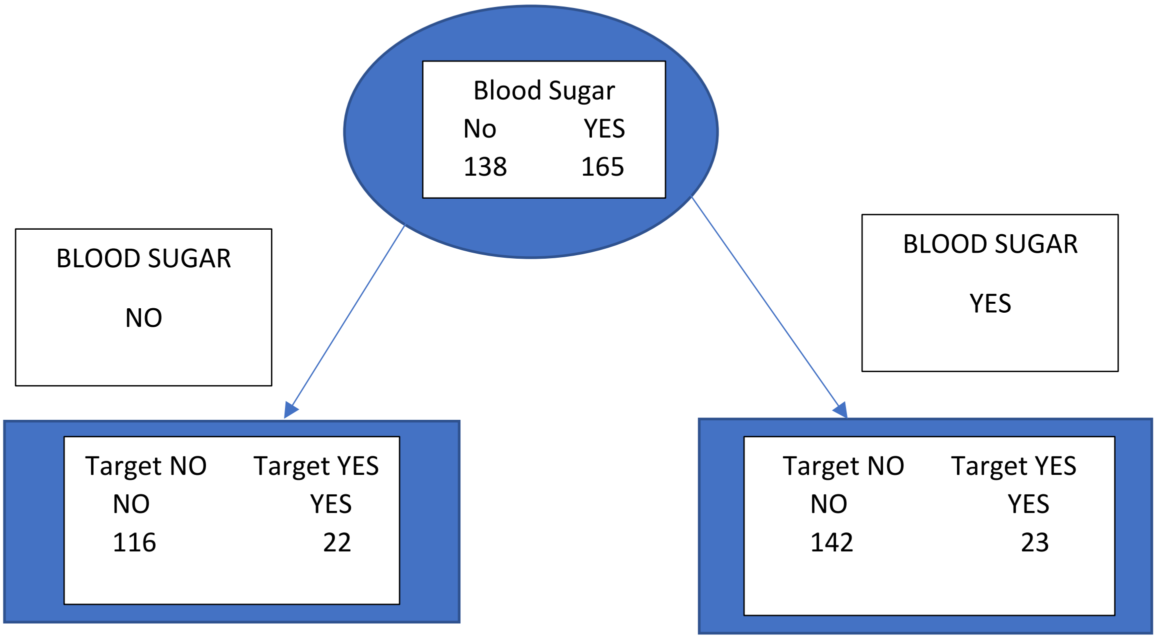
Gini impurity(S) = 1 - Σp(i)^2

Where p(i) is the proportion of items in class i in the set S.

The Gini impurity ranges between 0 and 1, with 0 indicating a pure set (all items belong to the same class) and 1 indicating a completely impure set (items belong to different classes in equal proportion).

When building a decision tree, the goal is to divide the data into subsets that have the lowest Gini impurity. The algorithm will divide the data into the subsets that will give the lowest Gini impurity, this way, it will increase the "purity" of data and classify the data into the correct class.

In summary, Gini impurity is a measure of the probability of misclassifying a randomly chosen item from the set, it is widely used in decision tree-based algorithms to determine the best split point in a dataset. The goal is to divide the data into subsets that have the lowest Gini impurity.



Gini Impurity — Left Node

I(Left - BS) = 1 – (116/(116+22))^2 – (22/(116+22))^2

I(Left - BS) = 0.268

Gini Impurity — Right Node

I(Right - BS) = 1 – (142/(142+23))^2 – (23/(142+23))^2

I(Right - BS) = 0.234

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

<u>ANS</u> Yes, unregularized decision trees are prone to overfitting. Decision trees are a type of algorithm that creates a tree-like model of decisions and their possible consequences. Each internal node in the tree represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. As the tree is built, it is designed to minimize the impurity of the data in each leaf node, this process continues until the tree is fully grown.

An unregularized decision tree is one that is allowed to grow to its full depth, without any constraints on the size or complexity of the tree. As the tree grows, it can create very complex models that fit the training data very well, but do not generalize well to new, unseen data. This is because the tree can fit to the noise present in the data, not just the underlying pattern.

The overfitting problem is caused by a tree having too many branches and too many leaf nodes. This means that the tree has learned the noise in the training data and it is not generalizing well to new data. When the tree is too complex, it can fit to the training data too well, but when it encounters new data, the tree is not able to generalize well and makes errors.

Regularization is a technique used to prevent overfitting. It is the process of adding constraints on the complexity of the model to make it simpler and reduce the risk of overfitting. Regularization techniques such as limiting the maximum depth of the tree or pruning the tree are commonly used to prevent overfitting in decision trees.

In summary, unregularized decision trees are prone to overfitting because they are allowed to grow to their full depth without any constraints on the size or complexity of the tree. This can cause the tree to fit to the noise in the training data, not just the underlying pattern, which can result in poor generalization to new data. Regularization techniques are used to prevent overfitting by adding constraints on the complexity of the model.

There are various techniques to prevent the decision tree model from overfitting. By default, the decision tree model is allowed to grow to its full depth. Pruning refers to a technique to remove the parts of the decision tree to prevent growing to its full depth. By tuning the hyperparameters of the decision tree model one can prune the trees and prevent them from overfitting. There are two types of pruning Pre-pruning and Post-pruning. Now let's discuss the in-depth understanding and hands-on implementation of each of these pruning techniques.

Pre-Pruning:

The pre-pruning technique refers to the early stopping of the growth of the decision tree. The pre-pruning technique involves tuning the hyperparameters of the decision tree model prior to the training pipeline. The hyperparameters of the decision tree including max_depth, min_samples_leaf, min_samples_split can be tuned to early stop the growth of the tree and prevent the model from overfitting.

Post-Pruning:

The Post-pruning technique allows the decision tree model to grow to its full depth, then removes the tree branches to prevent the model from overfitting. Cost complexity pruning (ccp) is one type of post-pruning technique. In case of cost complexity pruning, the ccp_alpha can be tuned to get the best fit model.

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

ANS Ensemble Technique means we are going to combine multiple algorithms, so here multiple means 30,40 or may be 100 different models and make one decision

Ensemble Technique has 2 types they are:

1] Bagging : A] Bagging Classifier / Regressor
B] Random Forest Classifier / Regressor

2] Boosting : A] Adaboost (Adaptive Boosting)
B] GBDT (Gradient Boosted Decision Tree)
C] XBGT (Xtreme Gradient Decision Tree)

Bagging Bootstrap

So what hapen hose lets say we have 15 different features and in bagging we say that in we can create different multiple model. the most multiple model can be Logistic Regression, Linear Regression, KNN, Decision Tree, SVM etc.

So, In this case, we say we have Cerate 5 different Decision Tree and say that Bootstrap = True this means with Replacement which means in 1st model we pass some feature from 15 features In 2nd model we pass Some other features likewise in five models we pass Some quantity of feature so when we say Bootstrap = True the any features will go here in any model it is In mix the feature may be Some in different model it is like Randomness So we have make 5 model so it will give 5 result and let Say 1st fail, 2nd pass , 3rd fail and 4&5th pass so the majority is pass to the Ultimate decision Is PASS so this is how it works in Classification model.

So in Regression (Predict Salary), So Same step will work on this like Sayed above 5 diff modol (DT) and 5 diff result and Bootstrap - True So Here 5 different Predict Salary we will get and we so averageof it So that is the ultimate result in regression problem
Now what if i said the Bootstrap = false then it is Called without Replacement So let say we have created Some model 5 different DT model and we have 15 features So 1st model will take 3 diff model is not available for other So now we have 12 feature the 2nd model have to pick ramdom from Remaining 12 feataure So that we get 3 different model in 5 diff DT model end this process is Called pasting and rest of prediction of classification and regression so going to be same.

Boosting

Boosting: it is Supervised Machine Learning. Used for both classification and regression problem. Boosting is an ensemble technique approach that start from a weaker decision and keep on building the model Such that the final prediction is the weighted sum of all the Weaker decision-makers. the weight are assigned based on the performance of an individual tree.

for example: Suppose there is a person and he got a job and he knows his salary the person ask us to predict his salary but we don't have any of the information. so we say 2.5 LpA then the person will Say i have 5 years of experience the we got some idea So this 5 years of experience will going to help us and We got some idea So now we change our prediction from 2.5 LPA to 6LPA, then the person will say it is wrong, You are no where closer, then we need some more boost then he will say my skill is Data Science then we got come more boost then will All the information 5 year of exp and Skill Data Science, then we again predict from 6LPA to 11LAP, then he will say that

You are not that close but it is wrong then we need again Some kind of boost here then we call say give Some hint then he will say my designation is Teamleader we got some boost here some and information then we again predict from 11 LPA to 13LPA then he will say you very close but give me one more hint then he will say his job location is Tier 2 City then we got some boost have then we again predict from 13LPA to 12 LPA then he will Say you are very close to the Salary what i got.

So this how boosting work behind the scenes so it means that those all feature the every feature is going to boost, every feature is going to help little bit. So here all the feature is connected to each other and at last we are going to make one decision.

Q7. What is the difference between Bagging and Boosting techniques?

ANS

Bagging	**Boosting**
The simplest way of combining predictions that belong to the same type.	A way of combining predictions that belong to the different types.
Aim to decrease variance, not bias.	Aim to decrease bias, not variance.
Each model receives equal weight.	Models are weighted according to their performance.
Each model is built independently.	New models are influenced by the performance of previously built models.
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.
Bagging tries to solve the over-fitting problem.	Boosting tries to reduce bias.
If the classifier is unstable (high variance), then apply bagging.	If the classifier is stable and simple (high bias) the apply boosting.
In this base classifiers are trained parallelly.	In this base classifiers are trained sequentially.
Example: The Random forest model uses Bagging.	Example: The AdaBoost uses Boosting techniques

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

ANS So in bagging we say we are going to Create diff or many Number of model So let say we have Cecate 5 diffrent Model of DT the data in that we are going to train But these are some amount of data that model does not train, the Same data we are going to used in testing the model that is nothing but out of bag evaluation.

Out-of-bag error is an important concept in random forests, which provides a measure of accuracy for the model. It is based on the idea that each tree in a random forest is built from a different subset of observations, and when testing the performance of the model, it can use those out-of-bag observations to get an estimate of its accuracy. This technique is useful as it allows one to test their model without having to use a separate validation set.

In Random Forests, the out-of-bag (OOB) error is a measure of the prediction error for instances not included in the training sets of the individual decision trees. In random forests, each tree is trained on a bootstrap sample of the data, which means that some observations are not included in the sample used to train each tree. These observations are referred to as out-of-bag observations, and the error rate for these observations is called the out-of-bag error.

The OOB error can be used as an estimate of the generalization error of the random forest. Since each tree is trained on a different subset of the data, the OOB observations for each tree are different. The OOB error is calculated by averaging the prediction error for the OOB observations across all trees in the forest.

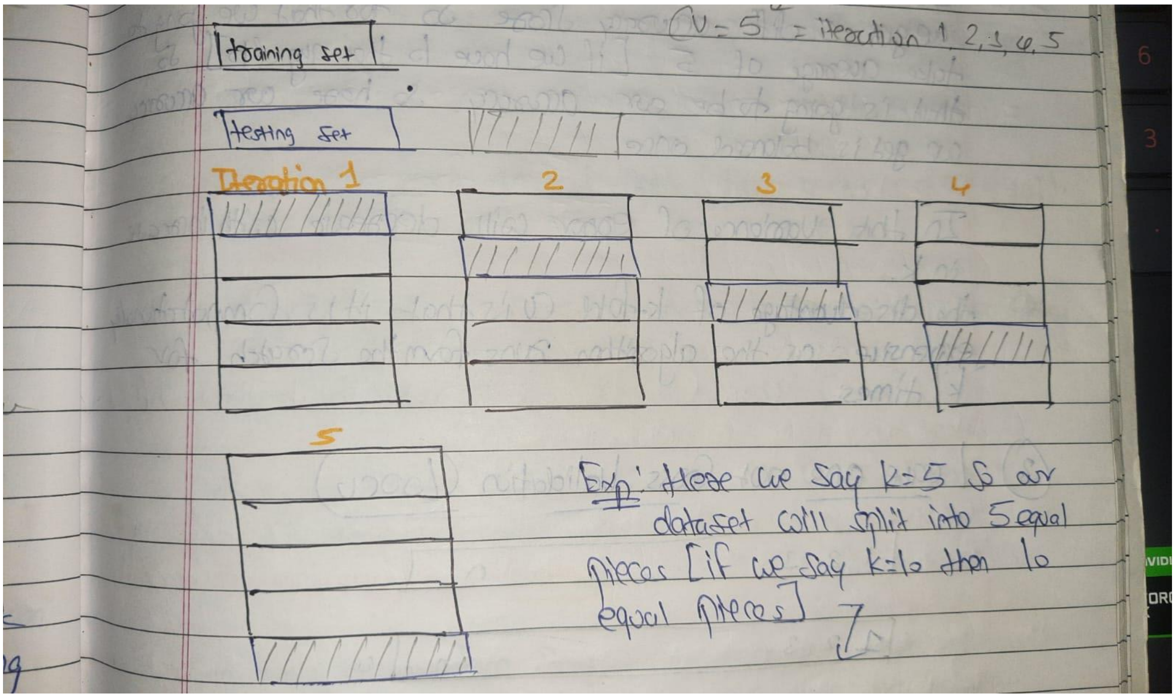
The OOB error can be calculated as follows:For each tree in the forest, calculate the prediction error for the OOB observations.Average the prediction errors across all trees in the forest.

The OOB error can be used to compare the performance of different random forests, and it can also be used as a measure of the generalization error of the forest. If the OOB error is low, it means that the random forest is likely to perform well on new, unseen data.

In summary, Out-of-bag error is a measure of the prediction error for instances not included in the training sets of the individual decision trees in a Random Forest. It can be used as an estimate of the generalization error of the random forest and can be used to compare the performance of different random forests.

Q 9. What is K-fold cross-validation?

<u>ANS</u>



Exp: Here we Say k=5 So our dataset will split into 5 equal pieces [if we say k=10 then 10 equal pieces]

So In 1st iteration let say we have 100% of data and that data is split into 5 pieces so 20% in each piece. So what to happen in 1st iteration Starting 20%. will kept for testing set and remaining 80% it will Use for training set.So Similarly thing will happen in 2nd iteration but the testing & training Set will change as shown in figure. So in every set or iteration we are training on diff set and testing on diff set so every set we are going to work get 5 different of accuracy Score because everything independently.

So at the end we need only one accuracy score but we get 5 diff accuracy score so for that we are going to take average of 5, so that is going to be our accuracy, so here our accuracy we get is balanced once.

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

<u>ANS</u>

Hyperparameter tuning is a process of optimizing the hyperparameters of a machine learning model to achieve better performance. It involves numerous iterations to optimize the hyperparameters according to the problem at hand and helps in getting the best possible results from a model. By adjusting these parameters, one can obtain higher accuracy and better generalization from their ML models. Hyperparameter tuning is an important step in machine learning as it helps improve the overall performance of models by finding optimal combinations for each parameter.

Hyperparameter tuning is an essential part of controlling the behaviour of a machine-learning model. If we don’t correctly tune our hyperparameters, our estimated model parameters produce suboptimal results, as they don’t minimize the loss function. This means our model makes more errors. In practice, key indicators like the accuracy or the confusion matrix will be worse.

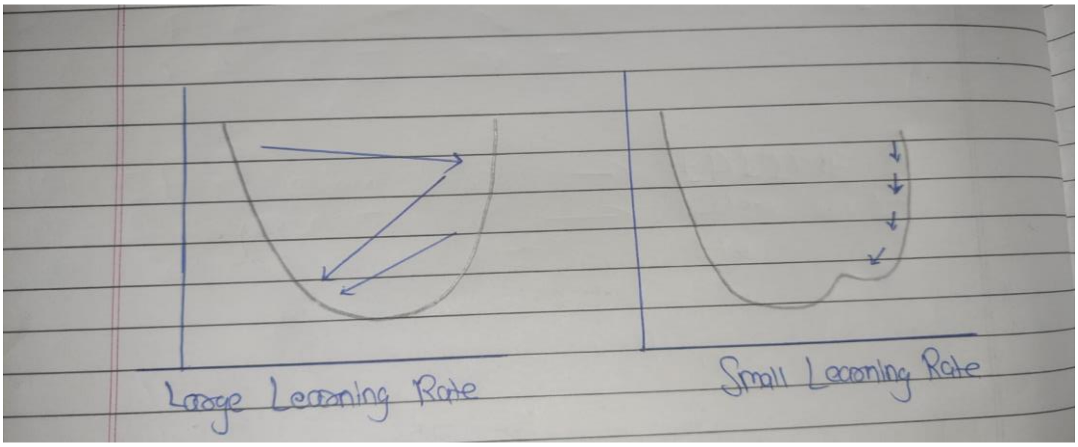
In machine learning, we need to differentiate between parameters and hyperparameters. A learning algorithm learns or estimates model parameters for the given data set, then continues updating these values as it continues to learn. After learning is complete, these parameters become part of the model. For example, each weight and bias in a neural network is a parameter.

Hyperparameters, on the other hand, are specific to the algorithm itself, so we can’t calculate their values from the data. We use hyperparameters to calculate the model parameters. Different hyperparameter values produce different model parameter values for a given data set.

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.

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

ANS Gradient Descent is a powerful algorithm used in many machine learning applications. However, it can be prone to errors due to the size of the learning rate. A large learning rate can cause an unstable convergence, which means that the algorithm will never converge on an optimal solution and instead will overshoot or undershoot and fail to reach a good result. Additionally, it can also cause oscillations in the cost function which will make it difficult for the model to learn and understand patterns correctly. .In the following scene, gradient descent is used on a linear regression model for the cost function. The learning rate is set at 10 and a constant step size of 0.1.



A large learning rate can cause the model to converge too quickly to a suboptimal solution or diverge completely. This can result in overshooting the global minimum and oscillating back and forth across the optimal solution, making it difficult for the model to converge. Additionally, a large learning rate can cause the model to converge to a high-bias solution, which can result in underfitting.

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

ANS Logistic regression is a linear model, which means that it is based on the assumption that the relationship between the features and the target variable is linear. Therefore, it is not suitable for classification of non-linear data as it cannot capture complex non-linear relationships between the features and the target variable.

Logistic regression uses a linear decision boundary to separate the classes, which is a straight line or a hyperplane. This means that it can only separate the classes if they can be separated by a linear boundary. For non-linear data, this is not the case. For example, if the data is distributed in a circular or a spiral shape, a linear boundary will not be able to separate the classes.

To classify non-linear data, other models such as decision trees, random forests, k-NN, SVM with a non-linear kernel, or neural networks should be used. These models can handle non-linear decision boundaries and can capture more complex relationships between the features and the target variable.In summary, Logistic Regression is a linear model and it is not suitable for classification of non-linear data because it cannot capture the complex non-linear relationships between the features and the target variable and it uses a linear decision boundary to separate the classes. Other models such as decision trees, random forests, k-NN, SVM with a non-linear kernel, or neural networks should be used for classifying non-linear data.

Q13 Differentiate between Adaboost and Gradient Boosting.

Features	Gradient boosting	Adaboost boosting
Model	It identifies complex observations by huge residuals calculated in prior iterations	The shift is made by up-weighting the observations that are miscalculated prior
Trees	The trees with week learners are constructed using a greedy algorithm based on split points and purity scores. The trees are grown deeper with eight to thirty-two terminal nodes. The week learners should stay a week in terms of nodes, layers, leaf nodes, and splits	The trees are called decision stumps.
Classifier	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.

Prediction	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.
Short-comings	Here, the gradients themselves identify the shortcomings.	Maximum weighted data points are used to identify the shortcomings.
Loss value	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.
Applications	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

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

ANS The bias-variance tradeoff is a fundamental concept in machine learning that refers to the tradeoff between a model's ability to fit the training data well (low bias) and its ability to generalize to new, unseen data (low variance). Bias refers to the error that is introduced by approximating a real-life problem that is too complex to be solved exactly. Variance, on the other hand, refers to the error that is introduced by the model's sensitivity to small fluctuations in the training data.

A model with high bias will have a simple structure and make strong assumptions about the data, which can lead to poor performance on unseen data. These models are often referred to as underfitting models because they do not capture the underlying pattern in the data. For example, a linear model is a high bias model as it makes a strong assumption that the relationship between the input and output is linear. This can lead to poor performance if the true relationship is non-linear.

On the other hand, a model with high variance will have a more complex structure and make fewer assumptions about the data, which can lead to overfitting to the training data. These models are often referred to as overfitting models because they capture not only the underlying pattern but also the noise in the data. For example, a high-degree polynomial model is a high variance model as it can capture even the small fluctuations in the training data. This can lead to good performance on the training data but poor performance on unseen data.

Finding the right balance between bias and variance is essential for building a successful machine learning model. In general, increasing model complexity will decrease bias and increase variance. Therefore, it is important to find the right level of model complexity that balances bias and variance.

One way to balance bias and variance is by using a technique called regularization. Regularization is a method to shrink the parameters of a model to prevent overfitting. There are many types of regularization techniques like L1 and L2 regularization. L1 regularization shrinks the parameters by adding the absolute value of the parameters to the cost function, whereas L2 regularization shrinks the parameters by adding the square of the parameters to the cost function. These regularization techniques are used to decrease the variance of the model without increasing the bias too much.

Another way to balance bias and variance is by using ensemble methods. Ensemble methods are a collection of multiple models that are combined to produce a more robust and accurate model. They are particularly useful in situations where a single model is not able to capture the underlying pattern in the data. For example, a random forest is an ensemble of decision trees. Each decision tree is trained on a different subset of the data and makes a prediction. The final prediction is made by taking the majority vote of the predictions made by all the decision trees. Ensemble methods are known to decrease the variance of the model without increasing the bias too much.

Finally, the bias-variance tradeoff can also be balanced by using cross-validation techniques. Cross-validation is a method to evaluate the performance of a model by dividing the data into training and testing sets. The model is trained on the training set and evaluated on the test set. This process is repeated multiple times to get a better estimate of the performance of the model. By using cross-validation techniques, we can avoid overfitting by evaluating the model on unseen data.

In conclusion, the bias-variance tradeoff is a fundamental concept in machine learning that refers to the tradeoff between a model's ability to fit the training data well (low bias) and its ability to generalize to new, unseen data (low variance). Finding the right balance between bias and variance

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

ANS

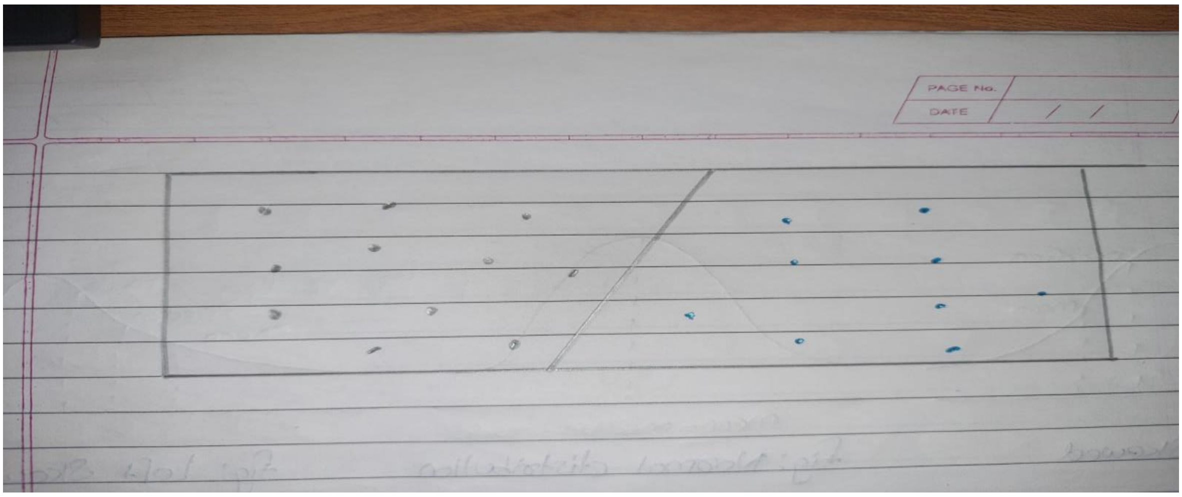
Linear Kernel

Linear Kernel is used when the data is Linearly separable, that is, it can be separated using a single Line. It is one of the most common kernels to be used. It is mostly used when there are a Large number of Features in a particular Data Set. In the above image, there are two set of features “Blue” features and the “Black” Features. Since these can be easily separated or in other words, they are linearly separable, so the Linear Kernel can be used here.

Advantages of using Linear Kernel:

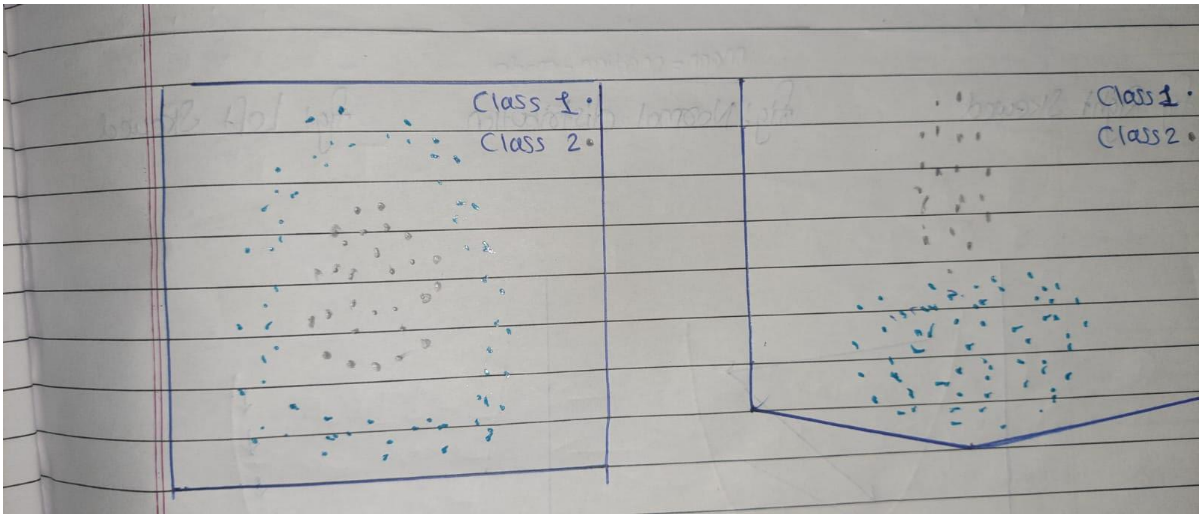
1. Training a SVM with a Linear Kernel is Faster than with any other Kernel.

2. When training a SVM with a Linear Kernel, only the optimisation of the C Regularisation parameter is required.



In the context of support vector machines (SVMs), a kernel is a function that transforms the input data into a higher dimensional space, where it is easier to find a linear boundary between different classes. Linear kernels are a type of kernel that do not perform this transformation, instead they operate directly on the input data in its original space. Linear kernels are often used when the data is linearly separable, meaning that a straight line can be drawn to separate the different classes. The linear kernel is the simplest kernel function and is defined as the inner product of the input vectors. It is represented by the equation K(x, y) = x.y, where x and y are the input vectors. Some of the common linear kernels used in SVM are Linear kernel, Polynomial kernel, and Radial basis function (RBF) kernel.

RBF



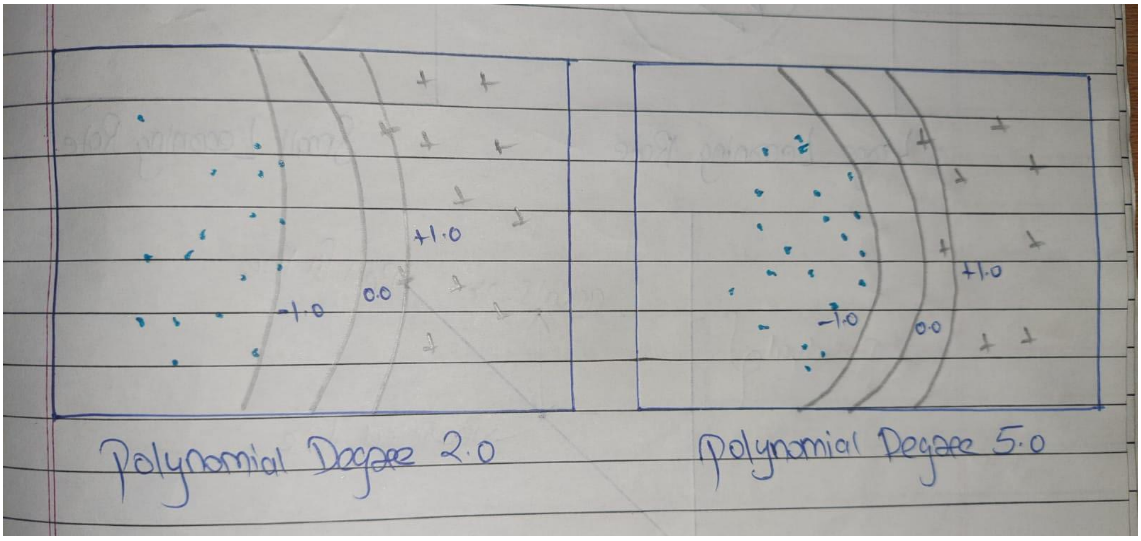
The radial basis function (RBF) kernel is a type of kernel commonly used in support vector machines (SVMs). It is a non-linear kernel that maps the input data into a higher dimensional space, where it is easier to find a non-linear boundary between different classes. The RBF kernel is defined as K(x, y) = exp(-gamma * ||x-y||^2), where x and y are the input vectors, gamma is a parameter that controls the width of the kernel and ||x-y||^2 is the squared Euclidean distance between x and y.

The RBF kernel is a popular choice for SVMs because it can handle non-linearly separable data, which is common in many real-world applications. The gamma parameter controls the complexity of the boundary, with smaller values leading to a smoother boundary and larger values leading to a more complex boundary. The choice of gamma is often determined through cross-validation.

In addition to its flexibility, the RBF kernel also has the property of universal approximation, meaning that it can approximate any continuous function to arbitrary accuracy. This makes it a useful choice for a wide range of datasets.

It's important to note that when the number of features is high and the number of samples is low, RBF kernel can be computationally expensive as it requires calculating the similarity between all pairs of samples.

Polynomial kernel



The polynomial kernel is a non-linear kernel that maps the input data into a higher dimensional space, where it is easier to find a non-linear boundary between different classes. The polynomial kernel is defined as K(x, y) = (x.y + c)^d, where x and y are the input vectors, c is a constant, and d is the degree of the polynomial. The polynomial kernel is particularly useful when the data has a polynomial relationship between the input and output variables, it can also be used for cases where the data is not linearly separable. The degree parameter d controls the complexity of the boundary, with smaller values leading to a simpler boundary and larger values leading to a more complex boundary. It's less computationally expensive than RBF kernel, but when the degree is too high, it can lead to overfitting and a high computational cost. When the dimensionality of the data is high, the polynomial kernel can be computationally expensive as it requires calculating the dot product between each pair of samples.