MACHINE LEARNING

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?**

   R-squared indicates the proportion of variance explained by the model and is intuitive, but it factors in the number of variables, potentially leading to overfitting. RSS measures the model fit directly by summing squared residuals, with a lower RSS indicating a better fit. The choice between them depends on dataset specifics, and commonly a combined approach alongside other metrics, like adjusted R-squared, provides a clearer measure of model fit. It depends on the specific characteristics of the dataset.

$$R^2 = 1 - \frac{Residual\ sum\ of\ squares}{Total\ sum\ of\ squares}$$

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

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

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

TSS = total sum of squares
n = number of observations
y_i = value in a sample
$\bar{y}$ = mean value of a sample

To determine which is a better measure of the goodness of fit model in regression between R-squared and Residual Sum of Squares (RSS), we need to consider both measures in the context of regression analysis. R-squared, or the coefficient of determination, indicates the proportion of variance in the dependent variable that is predictable from the independent variables. It is obtained by comparing the fitted model to a model with no explanatory variables except for the mean of the dependent variable. As R-squared approaches 1.0, it suggests a model with a better fit.

RSS, on the other hand, measures the overall model fit by summing up the squared differences between observed and predicted values, which are known as the squared residuals. In contrast to R-squared, the lower the RSS, the better the model's predictions match the actual data.

While R-squared is commonly used for its interpretability as a percentage of the variance explained, it is not without its limitations. Particularly, R-squared will always increase as more variables are added, regardless of whether those variables are meaningful to the model. This is where the adjusted R-squared comes into play in multiple regression, as it adjusts for the number of variables and helps prevent overfitting. Furthermore, if we consider non-linear models, the mean may not be a feature, making R-squared harder to calculate or interpret.

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.**

   The R-squared statistic provides a measure of fit. It takes the form of a proportion – the proportion of variance explained – and so it always takes on a value between 0 and 1.

   In simple words, it represents how much of our data is being explained by our model. For example, R^2 statistic = 0.75, it says that our model fits 75% of the total data set.

   Similarly, if it is 0, it means none of the data points is being explained and a value of 1 represents 100% data explanation.

   Mathematically, R^2 statistic is calculated as:

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

   Where, RSS: Residual Sum of Squares and is given by: RSS = Actual - Predicted

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

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

TSS = total sum of squares
n = number of observations
y_i = value in a sample
$\bar{y}$ = mean value of a sample

ESS:

Explained sum of square (ESS) or **Regression sum of squares** or Model sum of squares is a statistical quantity used in modelling of a process. ESS gives an estimate of how well a model explains the observed data for the process.

It tells how much of the variation between observed data and predicted data is being explained by the model proposed. Mathematically, it is the sum of the squares of the difference between the predicted data and mean data.

Let $y_i = a + b_1x_{1i} + b_2x_{2i} + ... + \varepsilon_i$ is regression model, where:

y_i is the i^{th} observation of the response variable

x_{ji} is the i^{th} observation of the j^{th} explanatory variable

a and b_i are coefficients

i indexes the observations from 1 to n

ε_i is the i^{th} value of the error term

Then

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

This is usually used for regression models. The variation in the model values is contrasted with the variation in the observed data (total sum of squares) and variation in model errors (residual sum of squares). The result of this comparison is given by ESS as per the following equation:

ESS = total sum of squares – residual sum of squares

As a generalization, a high ESS value signifies greater amount of variation being explained by the model, hence meaning a better model.

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

Regularization: If test data is more or less similar to train data, then the model will give very good accuracy then it is an overfitting problem. To avoid overfitting problem, we use regularization concept. Regularization helps to sort this overfitting problem by restricting the degrees of freedom of a given equation i.e. simply reducing the number of degrees of a polynomial function by reducing their corresponding weights.

In a linear equation, we do not want huge weights/coefficients as a small change in weight can make a large difference for the dependent variable (Y).

So, regularization constraints the weights of such features to avoid overfitting problem.

To regularize the model, a shrinkage penalty is added to the cost function.

Different types of Regularization in regression:

1. **LASSO**
2. **RIDGE**
3. **ELASTICNET (less popular)**

1. LASSO (Least Absolute Shrinkage and Selection Operator) Regression (L1 Form):
   LASSO is one of regression model. LASSO regression penalizes the model based on the sum of magnitude of the coefficients.

   The regularization term is given by: regularization = $\lambda * \Sigma|\beta_j|$

   Where, λ is the shrinkage factor.

   Let us consider an example of predicting the salary of an employee in an organization.

   There are many factors which are useful for predicting the salary.

   Many features like, skills, no. of years' experience, previous company salary and so on.

   But some unnecessary features like Date of Birth, Phone Number, Email, Aadhar Number which are not useful for predicting the salary.

   Hence, LASSO identifies whichever features are not correlated with label. LASSO gives zero importance to those features.

   Also, LASSO acts as a 'features selector'. More features also lead to overfitting problem. With LASSO will avoid it. So that is what LASSO does.

2. Ridge Regression (L2 Form):
   It is also does the same thing but it will give very minimum importance to those unused features. It gives little amount of importance say 0.0001%

   Ridge regression penalizes the model based on the sum of squares of magnitude of the coefficients.

   Regularization = $\lambda * \Sigma|\beta_j^2|$

   Where, λ is the shrinkage factor.

   Difference between RIDGE and LASSO:

   Ridge regression shrinks the coefficients for those predictors which contribute very less in the model but have huge weights, very close to zero. But it never makes them exactly zero.

   Thus, the final model will still contain all those predictors, though with less weights. This doesn't help in interpreting the model very well. This is where LASSO regression differs with RIDGE regression.

   In LASSO, the L1 penalty does reduce some coefficients exactly to zero when we use a sufficiently large tuning parameter λ.

   So, in addition to regularizing, LASSO also performs feature selection.

   Why do we use Regularization?

   Regularization helps to reduce the variance of the model, without a substantial increase in the bias.

   If there is a variance in the model, that means the model won't fit well for dataset different than training data.

   The tuning parameter λ controls this bias and variance tradeoff.

   When the value of λ is increased up to a certain limit, it reduces the variance without losing any important properties in the data.

   But after a certain limit, the model will start losing some important properties which will increase the bias in the data.

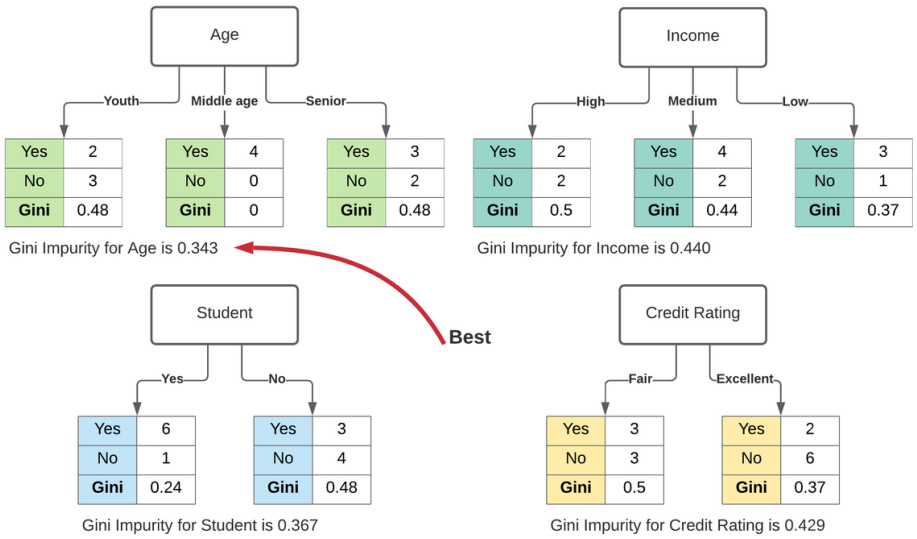
Thus, the selection of good value of λ is the key. The value of λ is selected using cross validation methods.

A set of λ is selected and cross validation error is calculated for each value of λ and that value of λ is selected for which the cross-validation error is minimum.

4. What is Gini–impurity index?

Gini-impurity:

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



For example, say you want to build a classifier that determines if someone will default on their credit card. You have some labelled data with features, such as bins for age, income, credit rating, and whether or not each person is a student. To find the best feature for the first split of the tree – the root node – you could calculate how poorly each feature divided the data into the correct class, default ("yes") or didn't default ("no"). This calculation would measure the impurity of the split, and the feature with the lowest impurity would determine the best feature for splitting the current node. This process would continue for each subsequent node using the remaining features.

In the image above, age has minimum Gini impurity, so age is selected as the root in the decision tree.

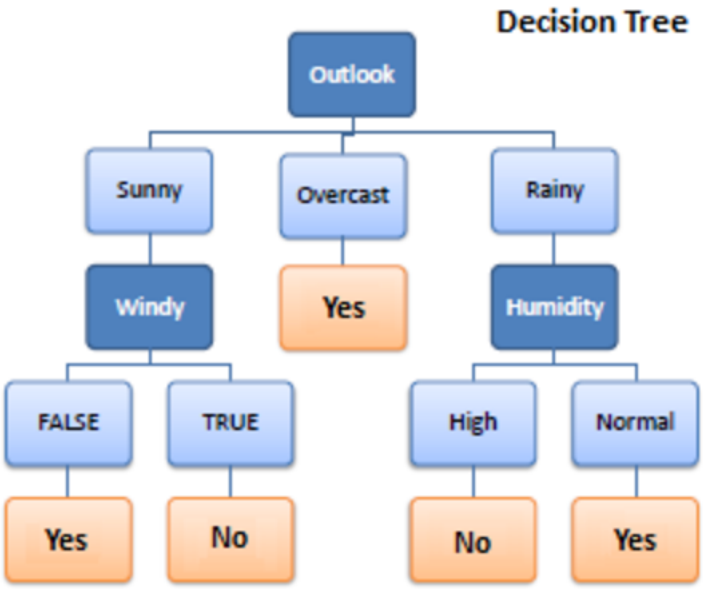
Mathematical definition Consider a dataset D that contains samples from k classes. The probability of samples belonging to class i at a given node can be denoted as p_i. Then the Gini Impurity of D is defined as:

$$Gini(D) = 1 - \sum_{i=1}^{k} {p_i}^2$$

The node with uniform class distribution has the highest impurity. The minimum impurity is obtained when all records belong to the same class. Several examples are given in the following table to demonstrate the Gini Impurity computation.

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

Decision trees are a type of model used for both classification and regression. Trees answer sequential questions which send us down a certain route of the tree given the answer. The model behaves with "if this than that" conditions ultimately yielding a specific result. This is easy to see with the image below which maps out whether or not to play golf.



The flow of this tree works downward beginning at the top with the outlook. The outlook has one of three options: sunny, overcast, or rainy. If sunny, we travel down to the next level. Will it be windy? True or false? If true, we choose not to play golf that day. If false we choose to play. If the outlook was changed to overcast, we would end there and decide to play. If the outlook was rainy, we would then look at the humidity. If the humidity was high we would not play, if the humidity is normal we would play.

Tree depth is an important concept. This represents how many questions are asked before we reach our predicted classification. We can see that the deepest the tree gets in the example above is two. The sunny and rainy routes both have a depth of two. The overcast route only has a depth of one, although the overall tree depth is denoted by its longest route. Thus, this tree has a depth of two.

Advantages to using decision trees:
1. Easy to interpret and make for straightforward visualizations.
2. The internal workings are capable of being observed and thus make it possible to reproduce work.
3. Can handle both numerical and categorical data.
4. Perform well on large datasets
5. Are extremely fast

Disadvantages of decision trees:

1. Building decision trees require algorithms capable of determining an optimal choice at each node. One popular algorithm is the Hunt's algorithm. This is a greedy model, meaning it makes the most optimal decision at each step, but does not take into account the global optimum. What does this mean? At each step the algorithm chooses the best result. However, choosing the best result at a given step does not ensure you will be headed down the route that will lead to the optimal decision when you make it to the final node of the tree, called the leaf node.

2. 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. An example of this could be predicting if the Boston Celtics will beat the Miami Heat in tonight's basketball game. The first level of the tree could ask if the Celtics are playing home or away. The second level might ask if the Celtics have a higher win percentage than their opponent, in this case the Heat. The third level asks if the Celtic's leading scorer is playing? The fourth level asks if the Celtic's second leading scorer is playing. The fifth level asks if the Celtics are traveling back to the east coast from 3 or more consecutive road games on the west coast. While all of these questions may be relevant, there may only be two previous games where the conditions of tonight's game were met. Using only two games as the basis for our classification would not be adequate for an informed decision. One way to combat this issue is by setting a max depth. This will limit our risk of overfitting; but as always, this will be at the expense of error due to bias. Thus if we set a max depth of three, we would only ask if the game is home or away, do the Celtics have a higher winning percentage than their opponent, and is their leading scorer playing. This is a simpler model with less variance sample to sample but ultimately will not be a strong predictive model.

Ideally, we would like to minimize both error due to bias and error due to variance. Enter random forests. Random forests mitigate this problem well. A random forest is simply a collection of decision trees whose results are aggregated into one final result. Their ability to limit overfitting without substantially increasing error due to bias is why they are such powerful models.

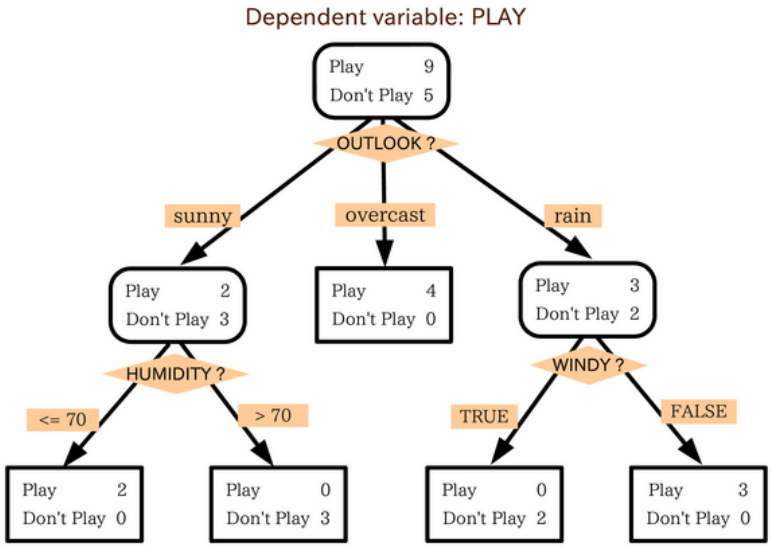
One way Random Forests reduce variance is by training on different samples of the data. A second way is by using a random subset of features. This means if we have 30 features, random forests will only use a certain number of those features in each model, say five. Unfortunately, we have omitted 25 features that could be useful. But as stated, a random forest is a collection of decision trees. Thus, in each tree we can utilize five random features. If we use many trees in our forest, eventually many or all of our features will have been included. This inclusion of many features will help limit our error due to bias and error due to variance. If features weren't chosen randomly, base trees in our forest could become

highly correlated. This is because a few features could be particularly predictive and thus, the same features would be chosen in many of the base trees. If many of these trees included the same features we would not be combating error due to variance.

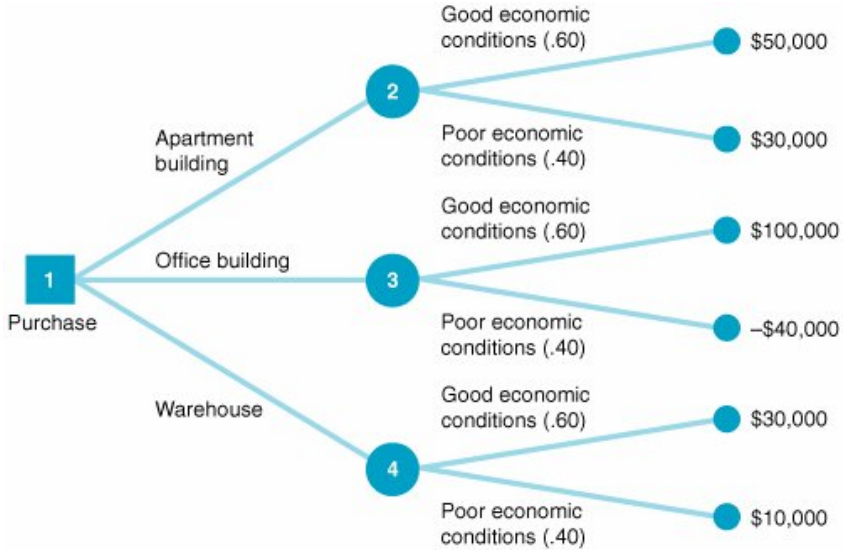
With that said, random forests are a strong modelling technique and much more robust than a single decision tree. They aggregate many decision trees to limit overfitting as well as error due to bias and therefore yield useful results.

6. What is an ensemble technique in machine learning?

Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model. To better understand this definition let's take a step back into ultimate goal of machine learning and model building. This is going to make more sense as I dive into specific examples and why Ensemble methods are used. I will largely utilize Decision Trees to outline the definition and practicality of Ensemble Methods (however it is important to note that Ensemble Methods do not only pertain to Decision Trees).



A Decision Tree determines the predictive value based on series of questions and conditions. For instance, this simple Decision Tree determining on whether an individual should play outside or not. The tree takes several weather factors into account, and given each factor either makes a decision or asks another question. In this example, every time it is overcast, we will play outside. However, if it is raining, we must ask if it is windy or not? If windy, we will not play. But given no wind, tie those shoelaces tight because were going outside to play.



Decision Trees can also solve quantitative problems as well with the same format. In the Tree to the left, we want to know whether or not to invest in a commercial real estate property. Is it an office building? A Warehouse? An Apartment building? Good economic conditions? Poor Economic Conditions? How much will an investment return? These questions are answered and solved using this decision tree.

When making Decision Trees, there are several factors we must take into consideration: On what features do we make our decisions on? What is the threshold for classifying each question into a yes or no answer? In the first Decision Tree, what if we wanted to ask ourselves if we had friends to play with or not. If we have friends, we will play every time. If not, we might continue to ask ourselves questions about the weather. By adding an additional question, we hope to greater define the Yes and No classes.

This is where Ensemble Methods come in handy! Rather than just relying on one Decision Tree and hoping we made the right decision at each split, Ensemble Methods allow us to take a sample of Decision Trees into account, calculate which features to use or questions to ask at each split, and make a final predictor based on the aggregated results of the sampled Decision Trees.

Summary:
The goal of any machine learning problem is to find a single model that will best predict our wanted outcome. Rather than making one model and hoping this model is the best/most accurate predictor we can make, ensemble methods take a myriad of models into account, and average those models to produce one final model. It is important to note that Decision Trees are not the only form of ensemble methods, just the most popular and relevant in Data Science today.

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

Bagging

Bagging is an acronym for ‘Bootstrap Aggregation’ and is used to decrease the variance in the prediction model. Bagging is a parallel method that fits different, considered learners independently from each other, making it possible to train them simultaneously.

Bagging generates additional data for training from the dataset. This is achieved by random sampling with replacement from the original dataset. Sampling with replacement may repeat some observations in each new training data set. Every element in Bagging is equally probable for appearing in a new dataset.

These multi datasets are used to train multiple models in parallel. The average of all the predictions from different ensemble models is calculated. The majority vote gained from the voting mechanism is considered when classification is made. Bagging decreases the variance and tunes the prediction to an expected outcome.

Implementation Steps of Bagging

- **Step 1:** Multiple subsets are created from the original data set with equal tuples, selecting observations with replacement.
- **Step 2:** A base model is created on each of these subsets.
- **Step 3:** Each model is learned in parallel with each training set and independent of each other.
- **Step 4:** The final predictions are determined by combining the predictions from all the models.

Boosting

Boosting is an ensemble modelling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models is added.

Boosting Algorithms

There are several boosting algorithms. The original ones, proposed by **Robert Schapire** and **Yoav Freund** were not adaptive and could not take full advantage of the weak learners. Schapire and Freund then developed AdaBoost, an adaptive boosting algorithm that won the prestigious Gödel Prize. AdaBoost was the first really successful boosting algorithm developed for the purpose of binary classification. AdaBoost is short for Adaptive Boosting and is a very popular boosting technique that combines multiple “weak classifiers” into a single “strong classifier”.

Similarities Between Bagging and Boosting, both being the commonly used methods, have a universal similarity of being classified as ensemble methods. Here we will explain the similarities between them.

Both are ensemble methods to get N learners from 1 learner. Both generate several training data sets by random sampling. Both make the final decision by averaging the N learners (or taking the majority of them i.e. Majority Voting). Both are good at reducing variance and provide higher stability.

Differences between Bagging and Boosting

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

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

A random forest is an ensemble machine-learning model that is composed of multiple decision trees. A decision tree is a model that makes predictions by learning a series of simple decision rules based on the features of the data. A random forest combines the predictions of multiple decision trees to make more accurate and robust predictions.

Random Forests are often used for classification and regression tasks. In classification, the goal is to predict the class label (e.g., “cat” or “dog”) of each sample in the dataset. In regression, the goal is to predict a continuous target variable (e.g., the price of a house) based on the features of the data.

Random forests are popular because they are easy to train, can handle high-dimensional data, and are highly accurate. They also have the ability to handle missing values and can handle imbalanced datasets, where some classes are more prevalent than others.

To train a random forest, you need to specify the number of decision trees to use (the **n_estimators** parameter) and the maximum depth of each tree (the **max_depth** parameter). Other hyperparameters, such as the minimum number of samples required to split a node and the minimum number of samples required at a leaf node, can also be specified.

Once the random forest is trained, you can use it to make predictions on new data. To make a prediction, the random forest uses the predictions of the individual decision trees and combines them using a majority vote or an averaging technique.

OOB (out-of-bag) Errors:

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.

Implementation of OOB Errors for Random Forests

To compute the OOB error, the samples that are not used in the training of an individual tree are known as "out-of-bag" samples. These samples are not used in the training of the tree, but they are used to compute the OOB error for that tree. The OOB error for the entire random forest is computed by averaging the OOB errors of the individual trees.

9. What is K-fold cross-validation?

K-fold cross validation: To tackle the high variance of Hold-out method, the k-fold method is used. The idea is simple, divide the whole dataset into 'k' sets preferably of equal sizes.

Then the first set is selected as the test set and the rest 'k-1' sets are used to train the data.

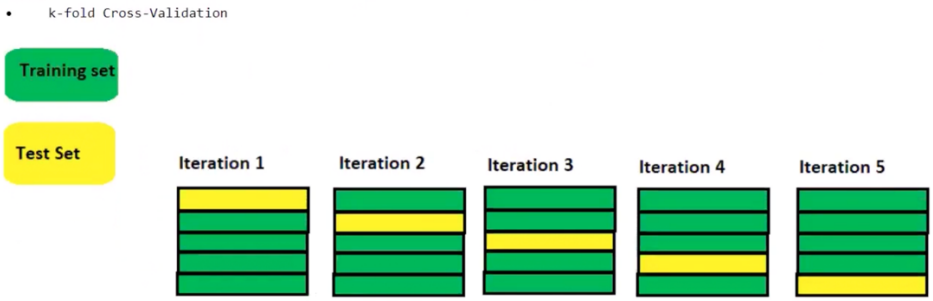
Error is calculated for this particular dataset. Then the steps are repeated i.e. the second set is selected as the test data, and the remaining 'k-1' sets are used as the training data.

Again, the error is calculated. Similarly, the process continues for 'k' times.
In the end, the CV error is given as the mean of the total errors calculated individually.

Mathematically given as:

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^{k} MSE_i$$

The variance in error decreases with the increase in 'k'. The disadvantage of k-fold cv is that it is computationally expensive as the algorithm runs from scratch for 'k' times.



LOOCV: Leave One Out Cross Validation (It is very rarely used technique)

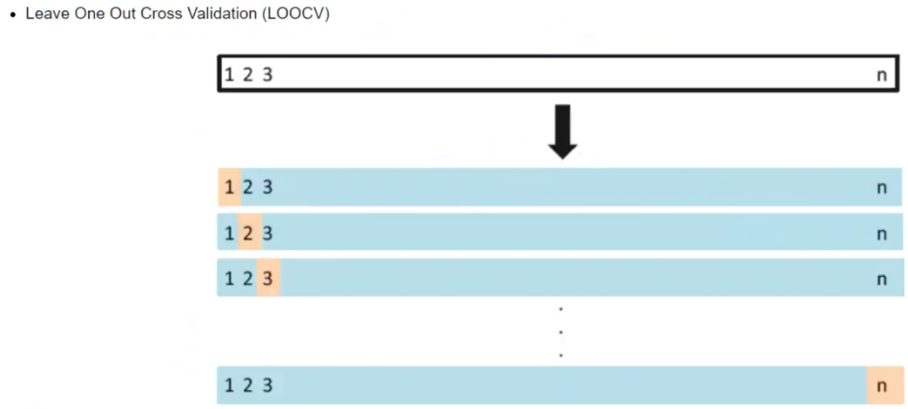
It is a special case of k – fold CV, where k becomes equal to n (number of observations). So instead of creating two subsets, it selects a single observation as a test data and rest of data as the training data.

The error is calculated for this test observations. Now, the second observation is selected as test data and the rest of the data is used as the training set.

Again, the error is calculated for this particular test observation.

This process continues 'n' times and in the end, CV error is calculated as:

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} MSE_i$$



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

Hyperparameter Tuning:

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

What are Hyperparameters?

In the context of machine learning, hyperparameters are configuration variables that are set before the training process of a model begins. They control the learning process itself, rather than being learned from the data. Hyperparameters are often used to tune the performance of a model, and they can have a significant impact on the model's accuracy, generalization, and other metrics.

Different Ways of Hyperparameters Tuning

Hyperparameters are configuration variables that control the learning process of a machine learning model. They are distinct from model parameters, which are the weights and biases that are learned from the data. There are several different types of hyperparameters:

Some other examples of model hyperparameters include: The penalty in Logistic Regression Classifier i.e. L1 or L2 regularization Number of Trees and Depth of Trees for Random Forests. The learning rate for training a neural network. Number of Clusters for Clustering Algorithms. The k in k-nearest neighbours.

Hyperparameter Tuning techniques: 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

Bayesian Optimization

Applications of Hyperparameter Tuning

- Model Selection: Choosing the Right Model Architecture for the Task
- Regularization Parameter Tuning: Controlling Model Complexity for Optimal Performance
- Feature Pre-processing Optimization: Enhancing Data Quality and Model Performance
- Algorithmic Parameter Tuning: Adjusting Algorithm-Specific Parameters for Optimal Results

Advantages of Hyperparameter tuning:

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

Disadvantages of Hyperparameter tuning:

- Computational cost
- Time-consuming process
- Risk of overfitting
- No guarantee of optimal performance
- Requires expertise

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

Gradient descent is an optimization algorithm used in machine learning to minimize the cost function by iteratively adjusting parameters in the direction of the negative gradient, aiming to find the optimal set of parameters.

The cost function represents the discrepancy between the predicted output of the model and the actual output. The goal of gradient descent is to find the set of parameters that minimizes this discrepancy and improves the model's performance.

The algorithm operates by calculating the gradient of the cost function, which indicates the direction and magnitude of steepest ascent. However, since the objective is to minimize the cost function, gradient descent moves in the opposite direction of the gradient, known as the negative gradient direction.

By iteratively updating the model's parameters in the negative gradient direction, gradient descent gradually converges towards the optimal set of parameters that yields the lowest cost. The learning rate, a hyperparameter, determines the step size taken in each iteration, influencing the speed and stability of convergence.

Gradient descent can be applied to various machine learning algorithms, including linear regression, logistic regression, neural networks, and support vector machines. It provides a general framework for optimizing models by iteratively refining their parameters based on the cost function.

How Does Gradient Descent Work?

1. It is an optimization algorithm used to minimize the cost function of a model.
2. The cost function measures how well the model fits the training data and is defined based on the difference between the predicted and actual values.
3. The gradient of the cost function is the derivative with respect to the model's parameters and points in the direction of the steepest ascent.
4. The algorithm starts with an initial set of parameters and updates them in small steps to minimize the cost function.
5. In each iteration of the algorithm, the gradient of the cost function with respect to each parameter is computed.
6. The gradient tells us the direction of the steepest ascent, and by moving in the opposite direction, we can find the direction of the steepest descent.
7. The size of the step is controlled by the learning rate, which determines how quickly the algorithm moves towards the minimum.

8. The process is repeated until the cost function converges to a minimum, indicating that the model has reached the optimal set of parameters.
9. There are different variations of gradient descent, including batch gradient descent, stochastic gradient descent, and mini-batch gradient descent, each with its own advantages and limitations.
10. Efficient implementation of gradient descent is essential for achieving good performance in machine learning tasks. The choice of the learning rate and the number of iterations can significantly impact the performance of the algorithm.

Challenges of Gradient Descent

While gradient descent is a powerful optimization algorithm, it can also present some challenges that can affect its performance. Some of these challenges include:

1. Local Optima: Gradient descent can converge to local optima instead of the global optimum, especially if the cost function has multiple peaks and valleys.
2. Learning Rate Selection: The choice of learning rate can significantly impact the performance of gradient descent. If the learning rate is too high, the algorithm may overshoot the minimum, and if it is too low, the algorithm may take too long to converge.
3. Overfitting: Gradient descent can overfit the training data if the model is too complex or the learning rate is too high. This can lead to poor generalization performance on new data.
4. Convergence Rate: The convergence rate of gradient descent can be slow for large datasets or high-dimensional spaces, which can make the algorithm computationally expensive.
5. Saddle Points: In high-dimensional spaces, the gradient of the cost function can have saddle points, which can cause gradient descent to get stuck in a plateau instead of converging to a minimum.

To overcome these challenges, several variations of gradient descent algorithm have been developed, such as adaptive learning rate methods, momentum-based methods, and second-order methods. Additionally, choosing the right regularization method, model architecture, and hyperparameters can also help improve the performance of gradient descent algorithm.

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

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.

While logistic regression makes core assumptions about the observations such as IID (each observation is independent of the others and they all have an identical probability distribution), the use of a linear decision boundary is not one of them. The linear decision boundary is used for reasons of simplicity following the Zen mantra – when in doubt simplify. In those cases where we suspect the decision boundary to be nonlinear, it may make sense to formulate logistic regression with a nonlinear model and evaluate how much better we can do. That is what this post is about. Here is the outline. We go through some code snippets here but the full code for reproducing the results can be downloaded from github.

- Briefly review the formulation of the likelihood function and its maximization. To keep the algebra at bay we stick to 2 classes, in a 2-d feature space. A point [x, y] in the feature space can belong to only one of the classes, and the function f (x,y; c) = 0 defines the decision boundary as shown in Figure 1 below.

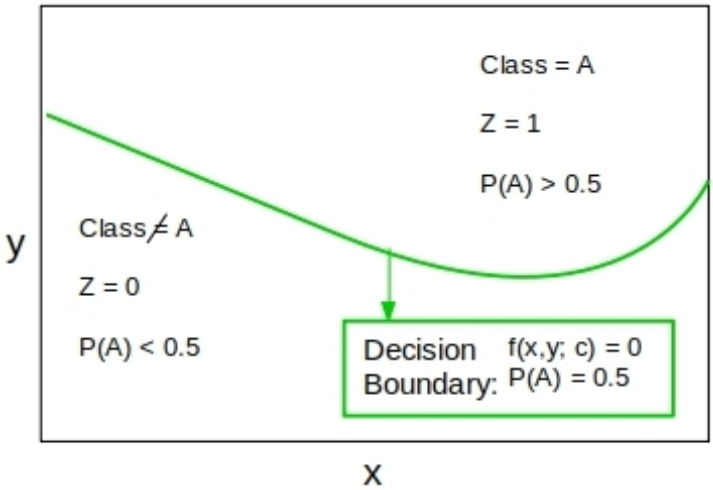


Figure 1. The decision boundary f(x,y; c) = 0 in feature space, separates the observations that belong to class A from those that do not belong to class A. c are the parameters to be estimated.

- Consider a decision boundary that can be expressed as a polynomial in feature variables but linear in the weights/coefficients. This case lends itself to be modeled within the (linear) framework using the API from scikit-learn.
- Consider a generic nonlinear decision boundary that cannot be expressed as a polynomial. Here we are on our own to find the model parameters that maximize the likelihood function. There is excellent API in the SciPy. Optimize module that helps us out here.

13. Differentiate between Adaboost and Gradient Boosting

Difference Between Boosting Algorithms

Algorithms	Gradient Boosting	AdaBoost
Year	–	1995
Handling Categorical Variables	May require pre-processing like one-hot encoding	No
Speed/Scalability	Moderate	Fast
Memory Usage	Moderate	Low
Regularization	NO	No
Parallel Processing	No	No
GPU Support	No	No
Feature Importance	Available	Available

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

Bias Variance Trade off:

Bias:

Bias represents the error introduced by a model's assumptions and simplifications about the data.

A model with high bias tends to be too simplistic, making it unable to capture the true underlying patterns in the data.

It often leads to underfitting, meaning the model performs poorly both on the training data and new, unseen data because it oversimplifies the problem.

Variance:

It refers to the model's sensitivity to the fluctuations or noise in the training data. A model with high variance is too complex and flexible, essentially "memorizing" the noise in the training data rather than generalizing well to new, unseen data.

High variance leads to overfitting, where the model performs excellently on the training data but poorly on new data because it hasn't learned the true patterns but instead has adapted to the specific examples in the training set.

In summary, the bias-variance trade-off is about finding the right balance between simplicity and complexity in a machine learning model.

A model that is too simple has high bias and low variance but lacks the capacity to learn from the data.

In contrast, a highly complex model has low bias but high variance, causing it to overfit and perform poorly on new data.

The goal of machine learning is to strike a balance between bias and variance, so the model generalizes well to unseen data. This process is typically achieved through model selection, hyperparameter tuning, and sometimes by using techniques like regularization to prevent overfitting.

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

SVM Explained

The **Support Vector Machine** is a **supervised learning algorithm** mostly used for **classification** but it can be used also for **regression**. The main idea is that based on the labeled data (training data) the algorithm tries to find the **optimal hyperplane** which can be used to classify new data points. In two dimensions the hyperplane is a simple line.

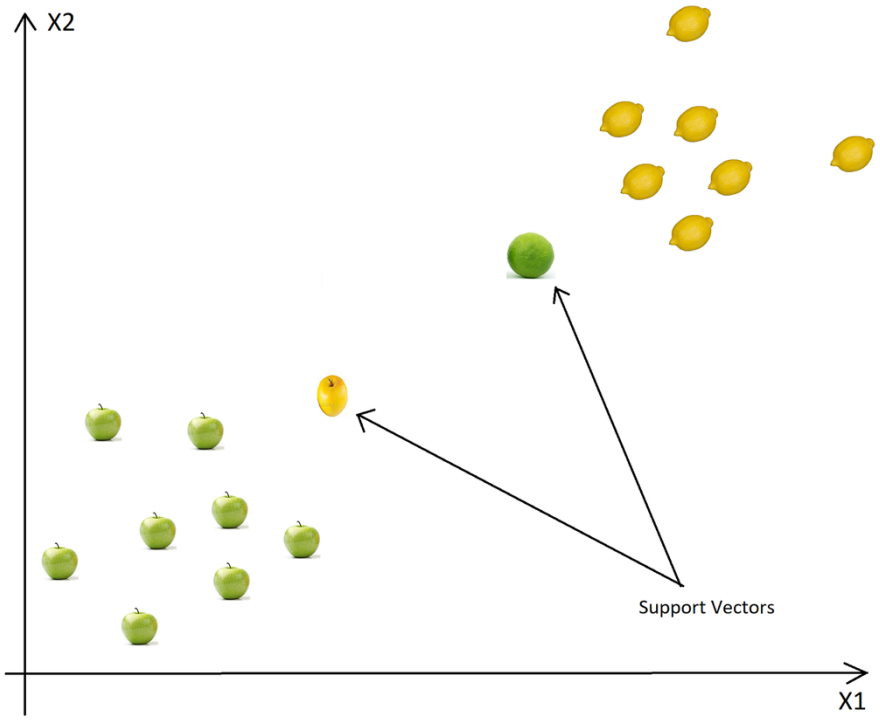
Usually a learning algorithm tries to learn the **most common characteristics (what differentiates one class from another)** of a class and the classification is based on those representative characteristics learnt (so classification is based on differences between classes). The **SVM** works in the other way around. It **finds** the **most similar examples** between classes. Those will be the **support vectors**.

As an example, let's consider two classes, apples and lemons.

Other algorithms will learn the most evident, most representative characteristics of apples and lemons, like apples are green and rounded while lemons are yellow and have elliptic form.

In contrast, SVM will search for apples that are very similar to lemons, for example apples which are yellow and have elliptic form. This will be a support vector. The other support vector will be a lemon similar to an apple (green and rounded). So **other algorithms** learns the **differences** while **SVM** learns **similarities**.

If we visualize the example above in 2D, we will have something like this:



STATISTICS WORKSHEET-5

1. Using a goodness of fit, we can assess whether a set of obtained frequencies differ from a set of frequencies.

Ans. d) Expected

2. Chisquare is used to analyse

Ans. c) Frequencies

3. What is the mean of a Chi Square distribution with 6 degrees of freedom?

Ans. c) 6

4. Which of these distributions is used for a goodness of fit testing?

Ans. b) Chi-squared distribution

5. Which of the following distributions is Continuous?

Ans. c) F Distribution

6. A statement made about a population for testing purpose is called?

Ans. b) Hypothesis

7. If the assumed hypothesis is tested for rejection considering it to be true is called?

Ans. a) Null Hypothesis

8. If the Critical region is evenly distributed then the test is referred as?

Ans. a) Two tailed

9. Alternative Hypothesis is also called as?

Ans. b) Research Hypothesis

10. In a Binomial Distribution, if ‘n’ is the number of trials and ‘p’ is the probability of success, then the mean value is given by

Ans. a) np