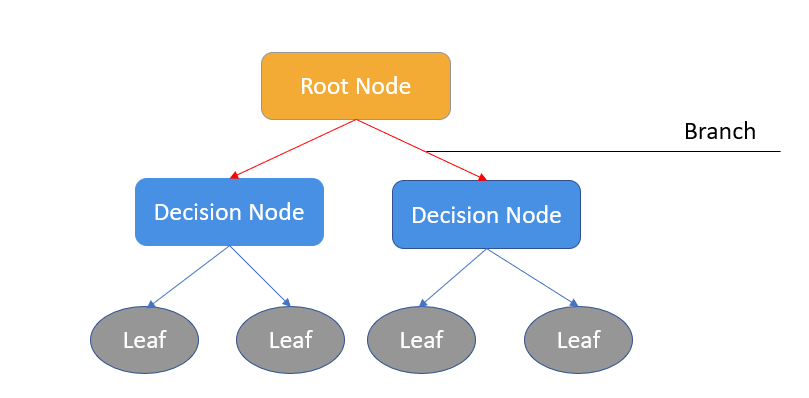
**Machine Learning II**

**MODULE – Decision Trees**

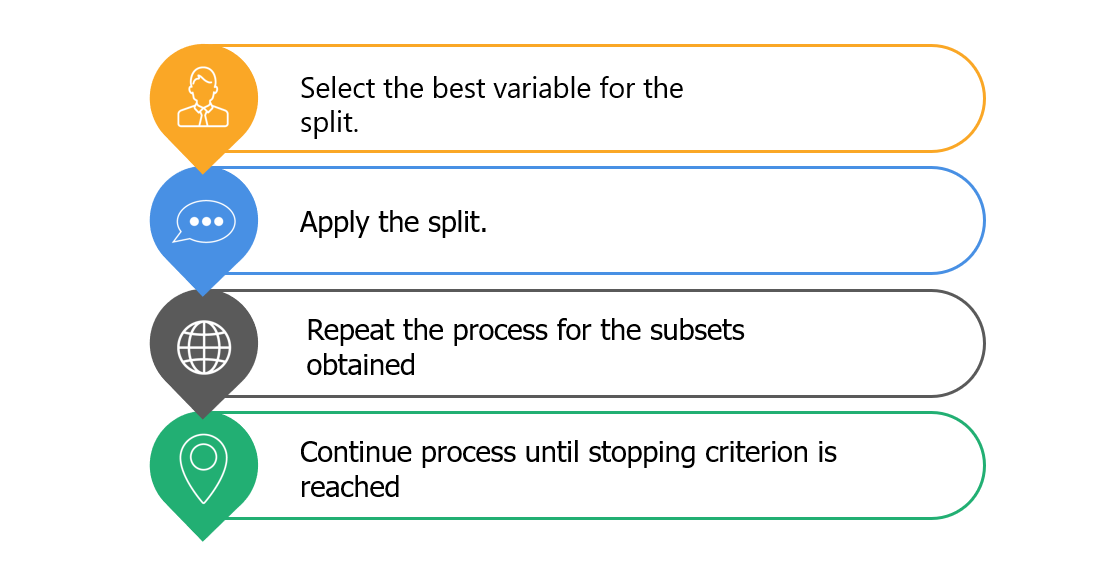
**Decision Trees**: A decision tree is a graph that uses branching methods to illustrate a course of action and various outcomes.

Decision Tree Components

* **Root Node**: The topmost decision node of the tree.
* **Branch**: The connectors between any two nodes are called branches.
* **Decision Node**: Decision Nodes or Sub-nodes also hold some decision parameter in them. Root Node is a Decision Node but at the top.
* **Leaf Node**: The last nodes in the decision tree where the classified groups lie.



Decision Tree Building Process



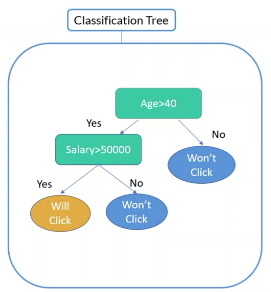
The Decision Tree Building process is a **top-down** and **greedy** approach. Greedy approach essentially refers to the algorithm making an optimal local choice at each node (and not care about future splits at that time) to arrive at the approximate optimal solution globally.

Decision Tree Types: Decision Trees being *Supervised Learning Algorithm* can be used for classification where the outcome is a class (Categorical variable) or for Regression where the outcome is a Quantitative field.

* Classification Tree
* Regression Trees

**Classification Trees**: Classification Trees aim to classify objects into one or more categories by taking the **mode** of the subset’s dependent variable.

Convention: If the decision criteria is met, you go to the left, else you go to the right.



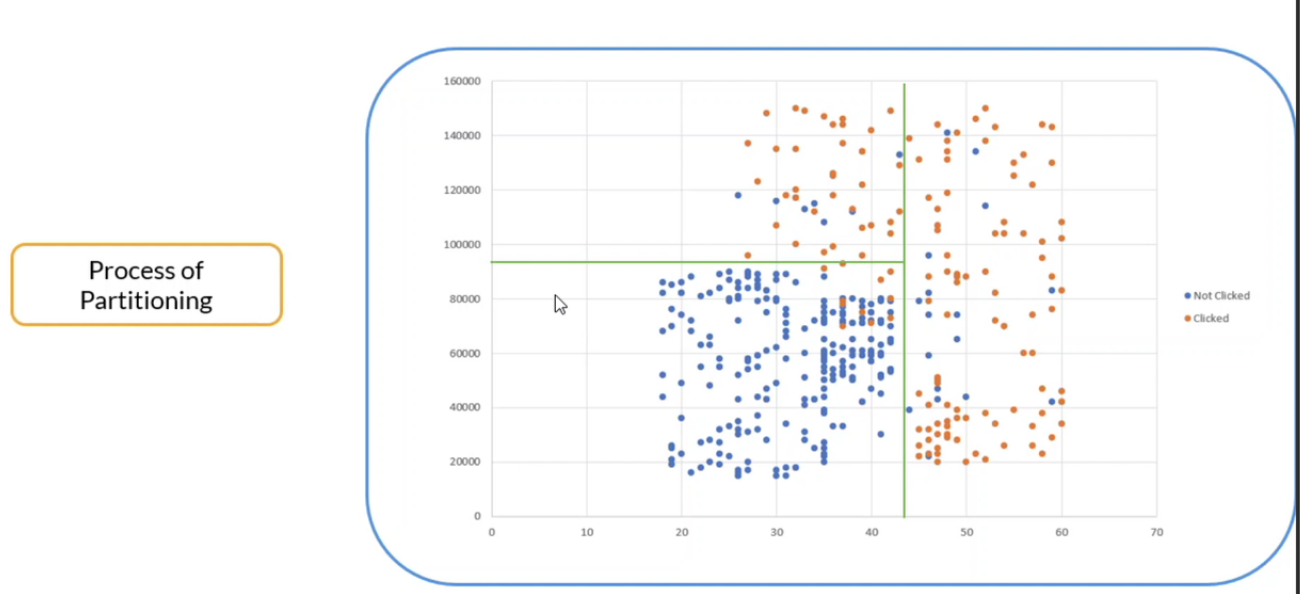
When we predict for a group that they will not click, it doesn’t have to be so that all of them didn’t click. How we are really predicting is that we see that subset and, in that subset, if the majority of them didn’t click, that is what our prediction would be. Prediction effectively is defined by what the majority in that subset is doing. In the above tree, for the Group (Age> 40 and Salary > 50000) it doesn’t mean that all the people in that group be click but a majority of the people in that group would.

**Impurity/ Homogeneity Measure**:

How to decide the split of a decision tree? Example: In the above diagram, how did we arrive at the first split, Age > 40. This cannot be done randomly. There are certain parameters based upon which we can decide where to make the split (which variable and for which value of that variable). These parameters are:

* ***Homogeneity***: It is used to assess whether a dataset or its subsequent partitions are homogenous or not.
* ***Impurity***: Impurity is the opposite of Homogeneity measure. As we go down a decision tree, impurity should hence decrease.

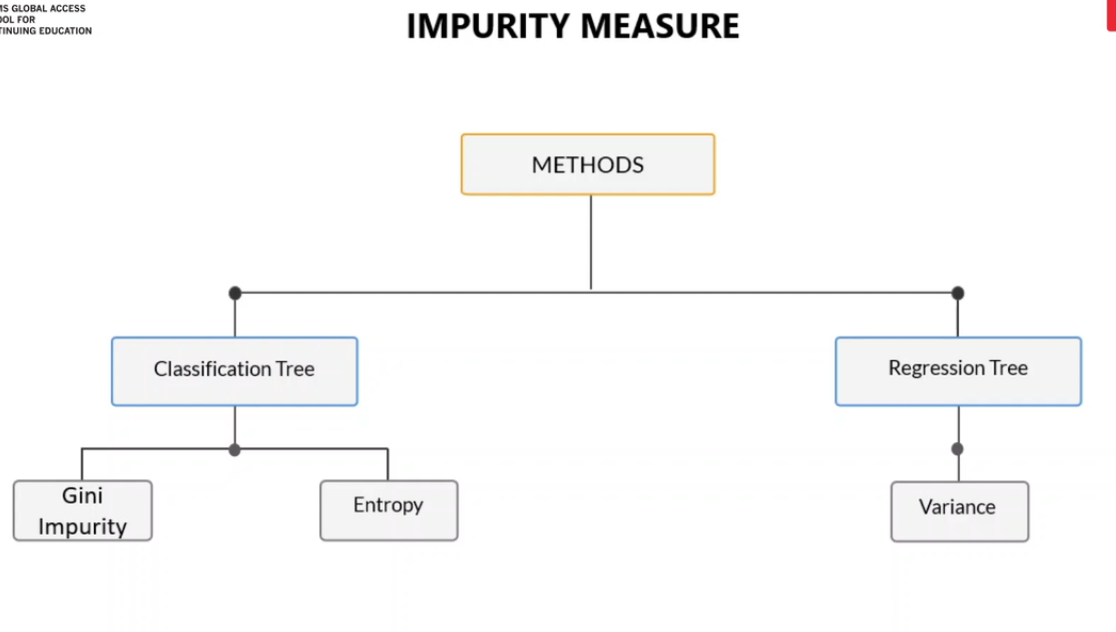
So, when we make a split, we need to access if when making a split is helping us to reduce the impurity or not. Only if that is the case, we would go ahead with the split.



**Impurity Measures**

Impurity Measurement Methods

Different methods are employed for Classification Trees and Regression Trees.



Classification Tree Methods:

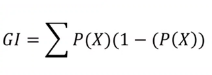
* **Gini Impurity**
* **Entropy**

Regression Tree Methods:

* **Variance**

**Quantifying the Splitting Criteria of Classification Tree**

**Gini Impurity** is calculated as:



If one type of class is present and the other is absent, then there is no uncertainty, so it is complete purity, so here, the Gini Impurity will be 0.

So,

**Best Gini Impurity = 0**

**Worst Gini Impurity for n variables = (n-1)/n.**

For 2 variable system, Worst Gini Impurity = ½.

For 3 variable system, Worst Gini Impurity = 2/3.

**Entropy** quantifies the amount of impurity present in any random variable and is calculated as:



Like Gini, Entropy is highest where there is more impurity (meaning both the classes are equally present).

**Entropy and Information Gain**

**Information Gain** is calculated as:

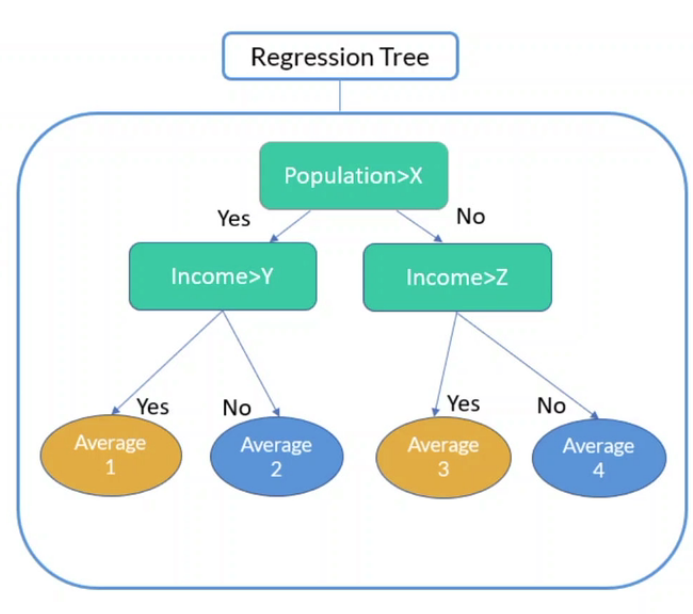
Information Gain = Entropy parent – [Weighted Average]\*Entropy children

A good split will give a higher Information Gain which also means that a good split will be one which reduces entropy more.

**Regression Trees**

There are certain cases where you cannot directly apply linear regression to solve a regression problem. Linear regression will fit only to a specific type of data set; however, you may want to divide the data set into multiple subsets and assess them separately.

Regression trees create similar subsets and then assign the **average** of those subsets as the dependent variable.

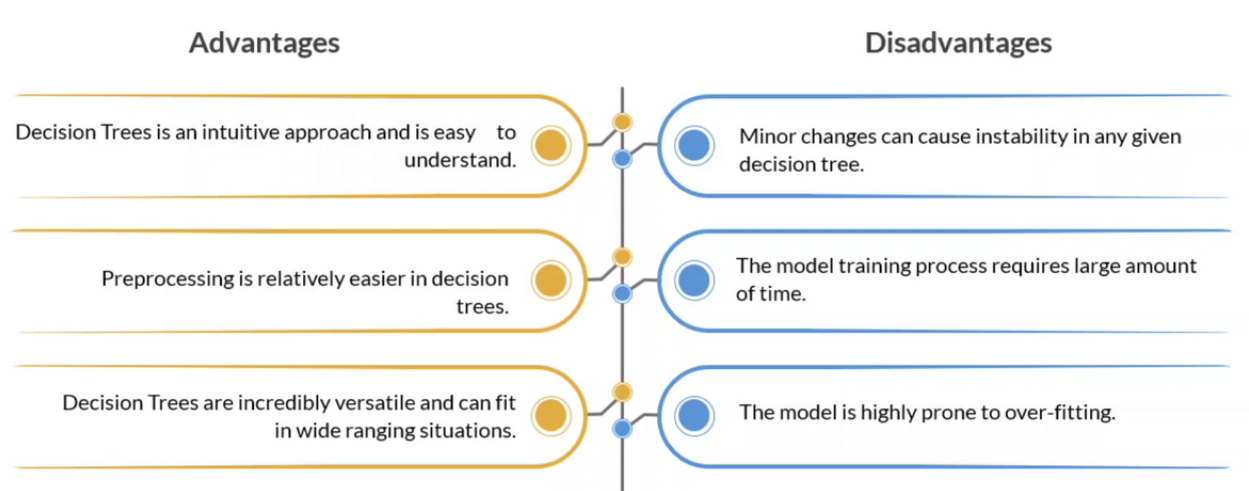


**Quantifying the Splitting Criteria of Regression Tree**

**Variance**: For Continuous variables, (or regression trees) variance reduction is the method employed for splitting. This involves measuring the net change in variance when you split a node.

**Variance** is the average of the net deviation from the mean.

**Advantages and Disadvantages of Decision Trees**



**Controlling overfitting in decision trees**

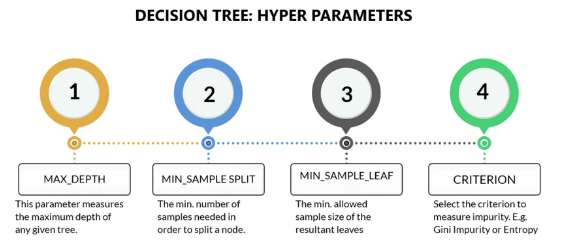
If you let the Decision tree grow freely, the tree can grow very deep. Means it will go to the extent to have just one member in the leaf node. This is a problem of overfitting.

There are two broad strategies to control overfitting in decision trees:

* **Truncation**: is the process of reducing the size of the tree by limiting its growth downwards. There are multiple ways of doing it. One of the ways is setting a hyperparameter for depth (say depth <3) which means there should be a maximum of three decision levels.
* **Pruning**: is the process of reducing the size of the tree by keeping only the model relevant decision nodes. For doing so, you start from the bottom of the tree and ask if that particular split is really adding any value( or is it adding sufficient incremental value) and if that is not the case, you remove that split.

**Hyperparameters and Algorithms**

Hyperparameters of a Decision Tree



* ***max\_depth***: The max\_depth parameter denotes the maximum depth of a tree. It can take any integer value or the ‘None’ value. If the ‘None’ value is taken, then the nodes are expanded until all the leaves are either pure or contain less than min\_samples\_split samples. By default, it takes the 'None' value.
* ***min\_samples\_split***: This denotes the minimum number of samples that are **required to** **split an internal decision node**. If an integer value is taken, then consider min\_samples\_split as the minimum number. If a float value is taken, then the parameter shows the percentage at each split. By default, it takes the '2' value.
* ***min\_samples\_leaf***: The denotes the minimum number of samples that are **required at a leaf node**. If an integer value is taken, then consider min\_samples\_leaf to be the minimum number. If a float value is taken, then the parameter shows the percentage at each split. By default, it takes the '1' value.
* ***criterion (Gini/IG or entropy):*** It defines the function to measure the quality of a split. Sklearn supports the 'gini' criteria for the Gini index and the 'entropy' criteria for information gain. By default, it takes the value 'gini'.

Decision Tree Algorithms

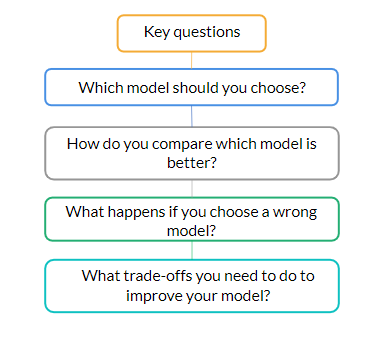
Below are the Algorithms used to create a Decision Tree:

* **CART Tress**
  + CART stands for Classification and Regression Tree and is a classic supervised learning model.
  + It produces only binary splits.
* **ID3**
  + It is called Iterative dichotomizer and was one of the first three decision trees.
  + It maximises information gain or minimises entropy.
* **C4.5**
  + C4.5 is an advanced version of ID3.
  + It can handle continuous and categorical variables.
* **C5**
  + C5.0 is an improved version of C4.5
* **CHAID**
  + This is an unsupervised model called chi squared auto iteration detector.
  + It produces multiple splits.

**MODULE – Model Selection and Optimisation**

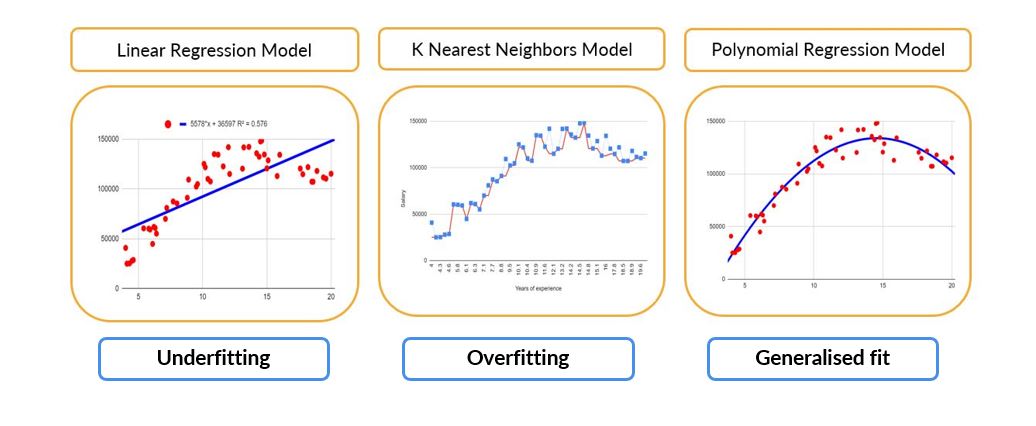
**Model Selection**: Selecting the best model for a given problem statement from a plethora of applicable machine learning models.

One needs to answer some key questions before selecting a model:

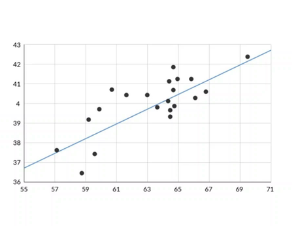


**Important Note**: When speaking of model selection, we are not merely speaking of the algorithm (Linear Regression, Logistic Regression, Decision Trees etc) but also of the ***hyperparameter*** that we are going to use( Value of k in k-NN, Depth in a Decision Tree etc).

Example of a given Business Problem data set and impact of various regressions on it:



**Underfitting**: Underfitting means not being able to extract information from the data. An underfit model fails to significantly grasp the relationship between the input values and target variables.



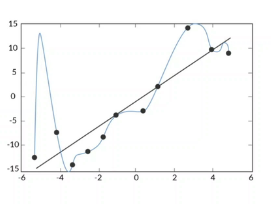
Causes of Underfitting

* ***Too less data***
  + Due to scarcity of data, model may not be able to capture the true relationship between source and target variable.
  + Building a model on few data points would make a bad model for prediction.
* ***Model is too simple***
  + The model is not good enough to learn the underlying trend in the data.
  + For a true polynomial relationship, trying to implement a linear regression model would lead to a bad model making.

Resolving the problem of underfitting

* *Step 1*: Identify the cause of the underfitting from the above two causes (Too less data, Model is too simple)
* *Step 2*: Having identified the cause, remedy accordingly:
  + Add Information: Increase the number of datapoints to be fed into the model for training.
  + Increase Complexity: Chose a more complex model if the simple model is leading to underfit. This can mean increasing the hyperparameters as well or even going from a simple linear regression to a multiple linear regression.

**Overfitting**: Overfitting is understanding the given data too well instead of learning the concept. The model tries to memorize all the data used to train it rather than learning the underlying trend of the data.

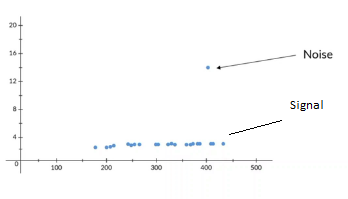


Causes of Overfitting

* ***Noise Capture***: A machine learning model overfits when it captures the noise of the data. Rather than focussing on the underlying trend, it moves towards fitting every point in the dataset in the model.
* ***Complexity***: Highly complex models are vulnerable to overfitting large numbers of input features.

Signal vs Noise

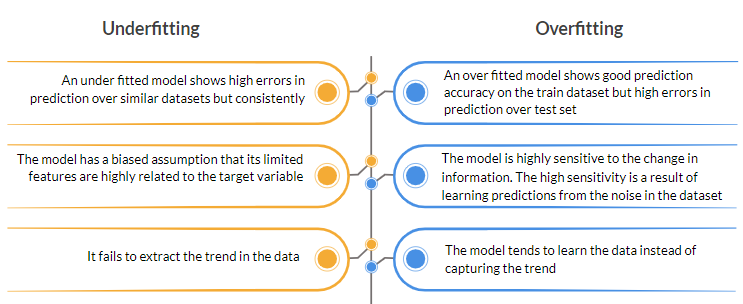
* ***Signal*** is the data which supports or transmits the underlying trend in the dataset. These data points occur more frequently and establishes the underlying trend of the data.
* ***Noise*** is an outlier. It occurs less frequently and does not contribute in establishing the underlying trend of the data.



Resolving the problem of overfitting

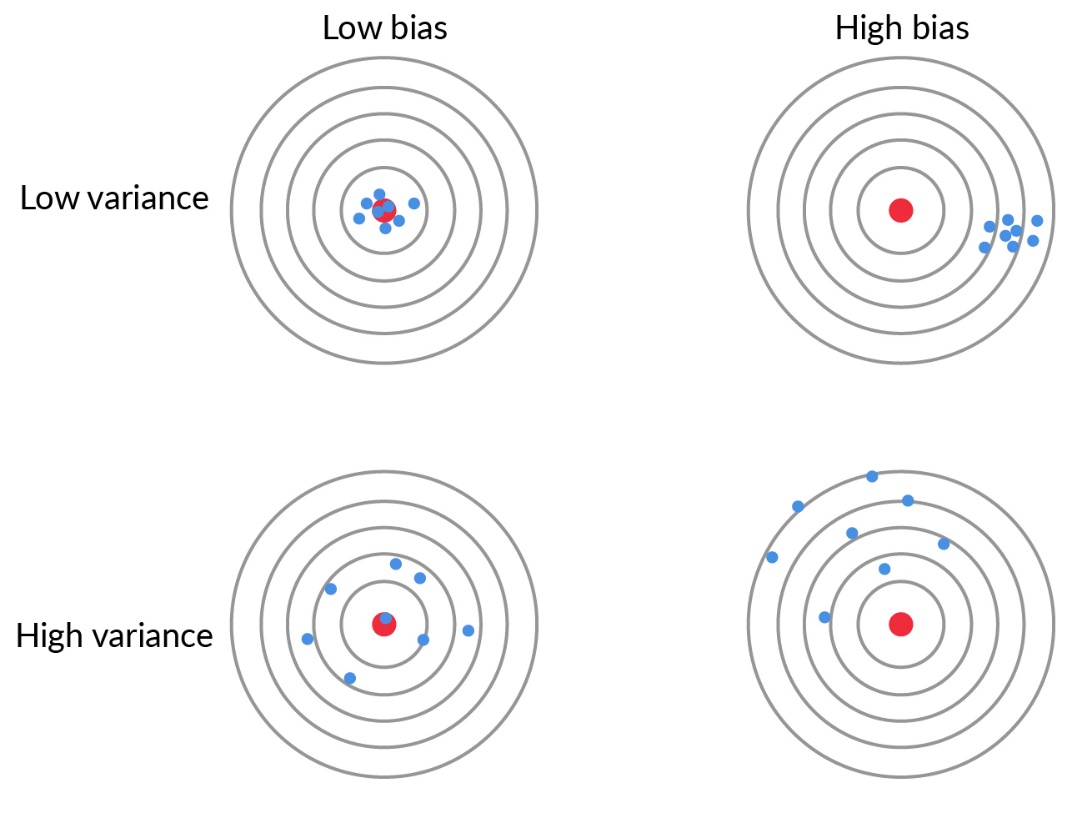
* **Train the model** with more data so that the signal points outnumber the noise points.
* **Reduce the model complexity** and emphasise the important features of the model.

**Underfitting vs Overfitting**



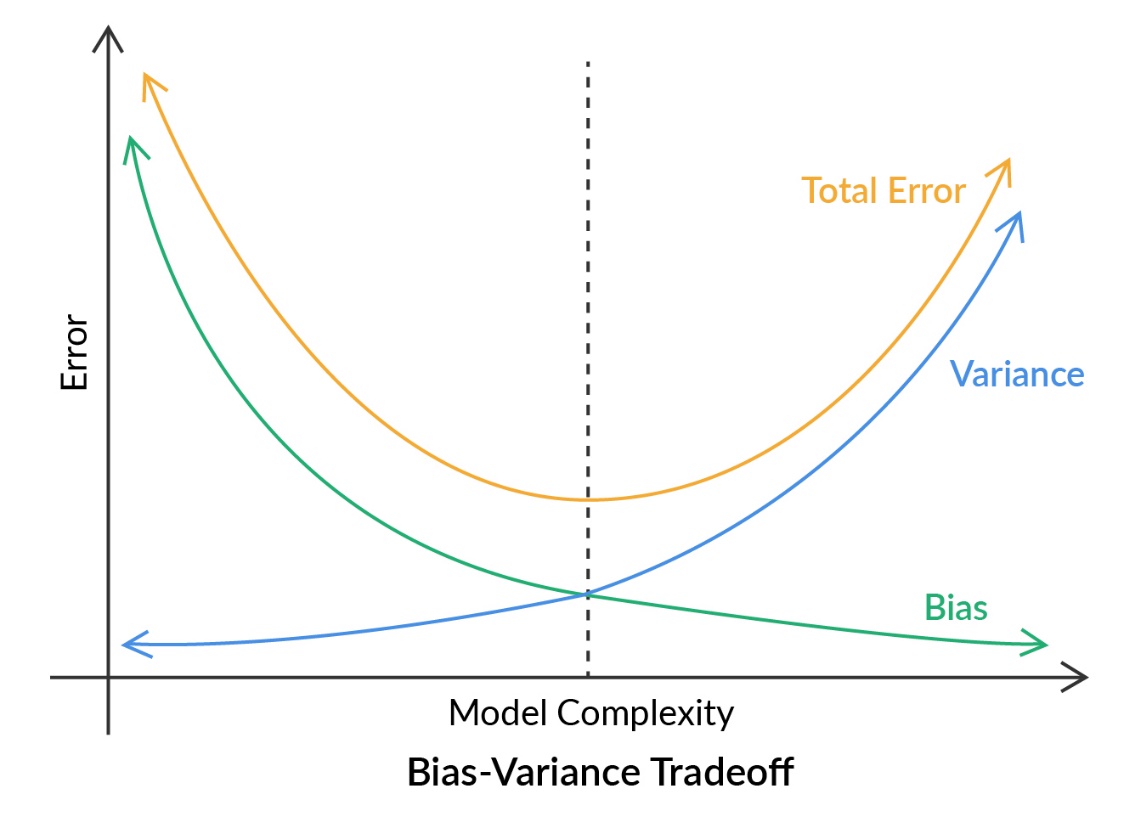
**Bias and Variance**:

* **Bias**: Bias is a phenomenon in which the predictions of the model are ***not close to the actual values*** but the ***difference*** between the prediction and the actual value is ***consistent***. Bias can be understood as a model which is ***not accurate but is precise*** in its predictions. A High Bias model would be one which is not able to capture the underlying associations doesn’t have much complexity. e.g., Simple Linear Regression.
  + Bias is the bias of the model in assuming the fact that a particular feature relates more to the target variable than the other variables.
  + The prediction results for a model with a high bias are significantly inaccurate as compared with its actual results. However, the error profile of the prediction remains similar when tested on different data sets of similar type.
  + A model with a high bias can be understood as a model that is not accurate but precise.
* **Variance**: When the model is very sensitive to the input data. Variance is a phenomenon in which the predictions of the ***model are close to the actual values*** for one dataset but show ***high errors*** when tested ***on another similar dataset***. In other words, the errors in the ***predictions show high inconsistency***. Variance could be understood as a model with ***good accuracy*** about its predictions but shows ***high variations***.
  + Variance refers to the sensitivity of a model based on the information change, i.e., how the error margins change for a model when it is tested over similar types of datasets.
  + A model with a high variance may show accurate predictions with a particular data set. However, it might show a high number of errors when tested with another similar type of data set.
  + A model with a high variance has predictions that are accurate but have a high variation in terms of results, i.e., high accuracy but low precision.

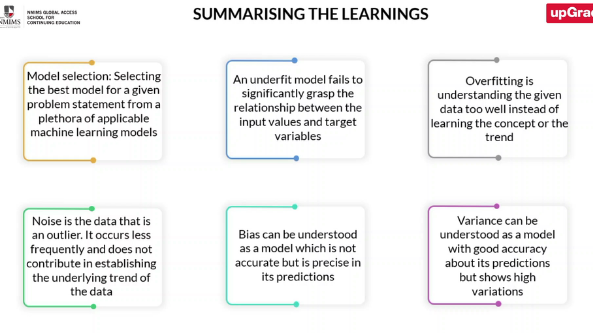


**Bias Variance Trade Off**

* The objective of any supervised model is to attain low variance and low bias so that it makes more accurate predictions.
* Increasing the bias will decrease the variance, and vice versa.
* It is always a challenge to select the parameters for the algorithm in order to maintain a bias-variance equilibrium.
* In the case of an underfitted model, the bias is high, and the variance is usually low, i.e., when tested on similar data sets, the error profiles in predictions are consistent, but individual errors are high. This leads to the model making faulty predictions for the majority of the data sets.
* In the case of an overfitted model, the bias is low, i.e., the model makes accurate predictions for the given data set. However, when it is tested on other similar data sets, the errors in predictions are extremely high.



Summarizing:



**Model Stability**

The key areas through which the model can be improved are:

* Data
  + How do you divide data into train and test?
  + How to get a more generalized model?
* Features
  + How features can affect model performance?
  + Are some features more important than others (e.g., Is one independent variable more important than the other)?
  + How many features should you have in your model (e.g., how many independent variables to consider)?
* Hyper-Parameters
  + What are in-built or hyperparameters (e.g., value of *k* in knn)?
  + How do we get the best set of inbuilt parameters?

**Model Optimization and Validation**

**Model Optimization** is fine tuning the model in order to generate the high model scores for the given data set.

**Model Validation** is a method to evaluate the trained model against a test data set. The main objective of using the test data set is to test the ability of a trained model for generalizing underlying logic.

**Drawbacks of Train-Test Split**

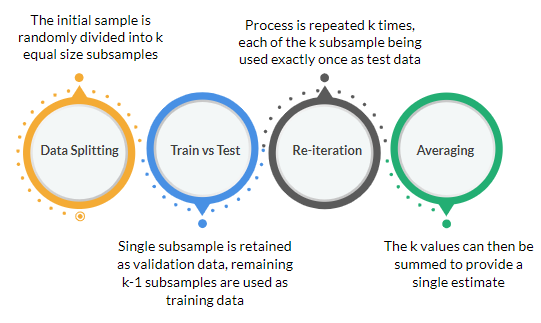
* ***Randomness of split***: Since the split of the data is random, there is a possibility that the split might be unhealthy for the model.
* ***Model performance***: Model scores are not consistent with say two or more different train data sets. Thus, we cannot decide which model fits best for the particular dataset.

**KFold Cross Validation**

KFold is a cross validation technique used to ***evaluate the model in an unbiased way***. ***K*** in KFold defines the ***number of splits*** in the cross-validation step.

In normal train-test, while arriving at a model, significance is given only to those data points which are selected for the train set. It will be better that all the data points in the entire data set get equal significance in arriving at the final model.

**KFold Process**



Advantages of KFold

* ***Uniformity***: All observations are used for both training and validation; hence the issue of irregular data split is removed.
* ***Unbiased validation***: Every observation is used exactly once for validation, which avoids a bias towards some data points and minimizes the chance of overfitting.

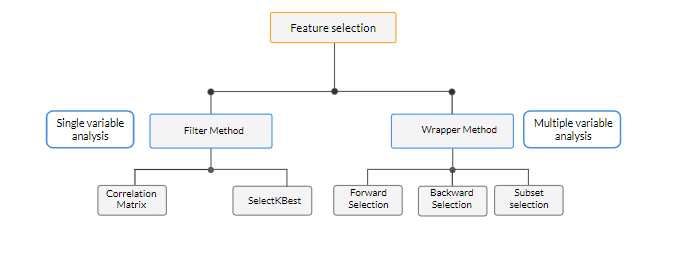
**Feature Selection**

Feature selection is a process to select the best feature combination to improve the efficiency of the ML model.

Advantages of Feature Selection

* ***Noise Reduction***: A large number of features and low sample / feature ratios adds noise to the dataset which causes the model to overfit and gives a false sense of performance and efficiency.
* ***Effort Reduction***: Reducing the number of features would minimize running time and also saves good amount of computation power. This allows us to perform more tests on the selected features and use algorithms of higher complexity.
* ***Simplicity***: A reduced feature set is easier for humans to understand and allows to concentrate on the key predictability sources and to do more accurate feature design.

Methods of Feature Selection



* ***Filter Method (For Single variable analysis):*** Here, we look at *one independent variable at a time* and analyse the independent variable on how it related to the dependent variable and then take a call whether we want to retain this independent variable or not. Methods are:
  + ***Correlation Matrix***: We see the correlation of the independent variable with the target variable through a correlation matrix and we can remove some features which don’t seem to be having much of a correlation.
  + ***SelectKBest***: We again check the correlation, but we choose the top K features.
* ***Wrapper Method (For Multiple variable analysis):*** Association between variables change in the presence of other variables. So, one variable by itself may not give the right picture. So, in wrapper method, we look at multiple methods together how they relate to the output or the target variable. In this method, the subset of the features that provide the best results over the given data is selected in a stepwise procedure. Methods are:
  + ***Forward Selection***
  + ***Backward Selection***
  + ***Subset Selection***

Wrapper method gives slightly better output than filter method as it also accounts for correlation among input variables, but requires high computational power, cost and time.   
ONLY FILTER METHODS IN THIS COURSE

**SelectKBest**:

Steps:

1. *Association tests* (Chi-sqaure (association between categorical independent and categorical dependent) , ANOVA(association between categorical and numerical), Correlation(association between numerical and numerical)) are performed of the dependent variable with the independent features and we get the best independent features.
2. *Selecting variables*: We select ‘K’ best features out of all which are highly related to the target variable.

**Correlation Matrix**:

Steps:

1. *Establishing Connection*: Establish a correlation matrix among dependent and independent variables and we create a correlation matrix.
2. *Selecting Variables*: Absolute correlation values to be considered for selecting features.

**Hyper-parameter Tuning**:

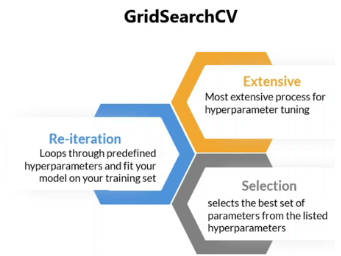
Hyperparameters: Parameters which can be modified to control the behaviour of any machine learning algorithm are called hyperparameters. E.g., K in kNN, Tree depth in Decision Trees. Models can have many parameters and determining the right set of parameters can be treated as a ***search issue***.

Advantages of Hyperparameter Tuning

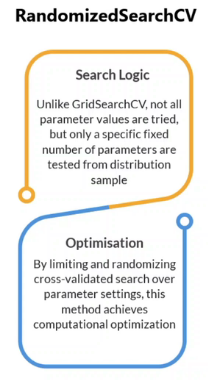
* ***Model Optimization***: Choose hyperparameters for highest model score.
* ***Effort Reduction***: Saves computational power and time for future.

Methods to find best hyperparameter combination

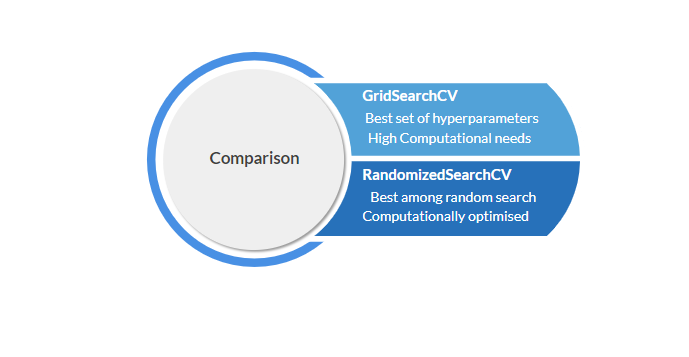
**GridSearchCV function**: The GridsearchCV function runs the model on all possible permutations of the hyperparameters to obtain the highest model score value. The GridsearchCV function runs an extensive process throughout the model and requires a huge amount of computational complexity and cost.



**RandomizedSearchCV function**: Unlike GridsearchCV, the RandomizedSearchCV function runs a specified fixed number of random combinations and returns the hyperparameters corresponding to the best model score out of the tested combinations.



Comparison between GridSearchCV and RandomizedSearchCV



**MODULE – Ensembles**

**Ensembles**: Ensemble combines a number of algorithms (**base learners**) together by some algorithm into a single predictive model with ***increased accuracy***. Ensemble learners or ensembles are known as **strong learners**.

*Base learners* are the individual learners or models which use a machine learning algorithm such as decision trees, linear regression etc to predict the results as accurately as possible. Essentially, you can create ensembles by combining all types of models. An ensemble can have various models, such as logistic regression, a neural network, a few decision trees, etc., working in unison.

***Why ensembles?***

* Improves predictions/accuracy and make model more robust.
* Avoids overfitting by introducing randomisation.
* Reduces Bias in the models.
* Reduces Variance in the models.
* Results in a more generalised model. i.e., a model that tends to reduce overfitting.

**Diversity Channels**

Ensembles means combining a number of algorithms – base learners. But how to choose individual models to develop an ensemble. The two key factors are:

**Diversity**: Diversity ensures that the models serve complementary purposes, i.e., they can work independently. For example, each base learner would derive different insights from the given data, i.e., it would make different assumptions while building the model; and then when you combine the result of each model, you get a model that does not rely on the assumptions of a single model.

**Acceptability**: Acceptability implies that each model is at least better than a random model. A random model is the one in which the probability of any event occurring is 50%. A base learner better than a random guesser means that the probability of the result being incorrect is less than 50%; hence, the probability of the ensemble being wrong would be far less than the probability of any individual model going wrong.

**How to achieve Diversity?**

***Sub-sampling the training data***: When you make an ensemble, you do expose the exact same data to all the different models. You expose different models to different subsets of the data. Thus, some randomization/some diversity has been introduced.

***Manipulating the training hyperparameters***: Just be changing the hyperparameters, a new model will be created even if the models have been exposed to the same data set. This also introduces randomization.

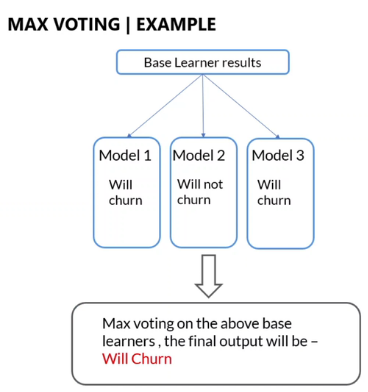
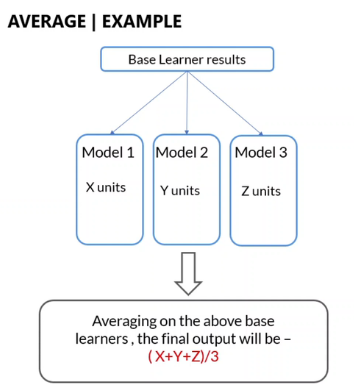
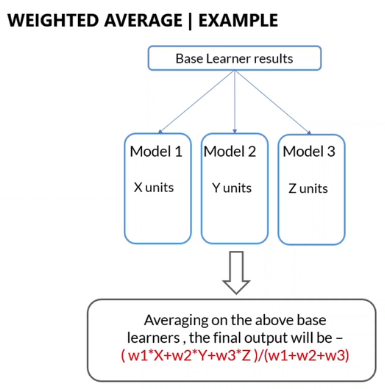
***Use different features***: Say for first decision tree, you show five features. For another decision tree, you show another five features. Since the features that each model is looking into is different, the models will learn different things from different models.

***Using different base learning algorithms***: Use different base learning model like Multiple Regression, Logistic Regression etc.

***Using a combination of the above methods***: We can combine above methods.

**Combining Output**

Ways of combining output:

* ***Max Voting***
  + Problem domain: Classification
  + The final output is obtained by voting through the outputs of all the models. The class with maximum votes is considered as the predicted class.
  + 
* ***Average***
  + Problem domain: Regression (Quantitative)
  + The final output is obtained by taking the mean of predicted outputs of all the models.
  + 
* ***Weighted Average***
  + Problem domain: Regression (Quantitative)
  + The final output is obtained by taking the weighted mean of predicted outputs of all the models. The base models with higher accuracy tend to have more weights.
  + 

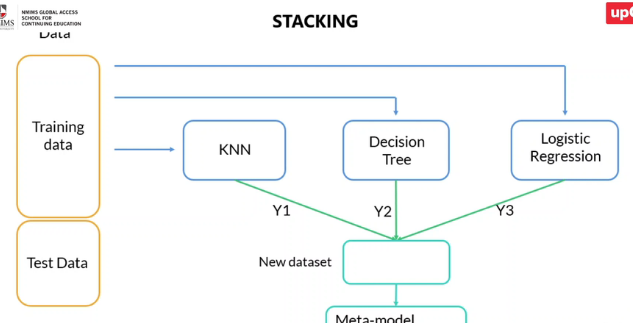
How to take the weighted average?

This is done by the means of **Stacking**.

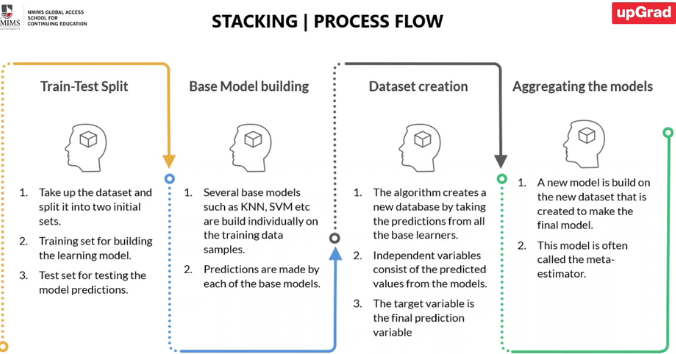
Stacking uses different machine learning algorithms to build **first level learners** and then the outputs from these are given as inputs to another model to form a **second level learner**. It is also called as **Stack Generalization**.

Why stacking?

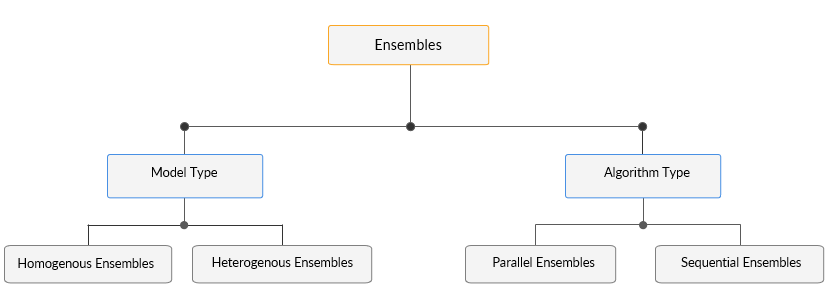
Combining outputs of multiple models manually is not optimal. So, in Stacking, multiple models make intermediate predictions on the training data. Then a new model uses these predictions to make the final predictions. The model is said to be stacked on top of other models.

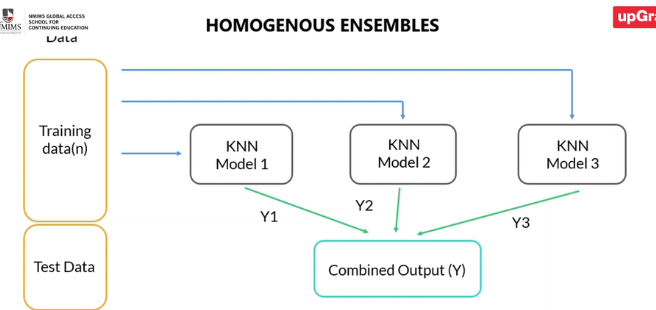
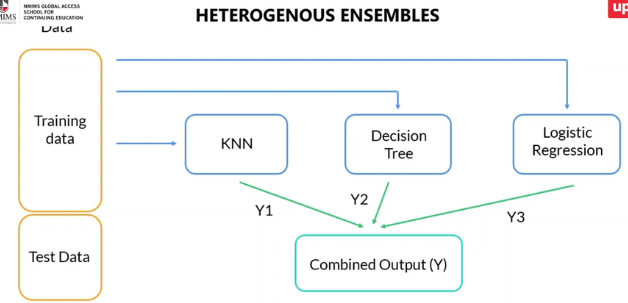
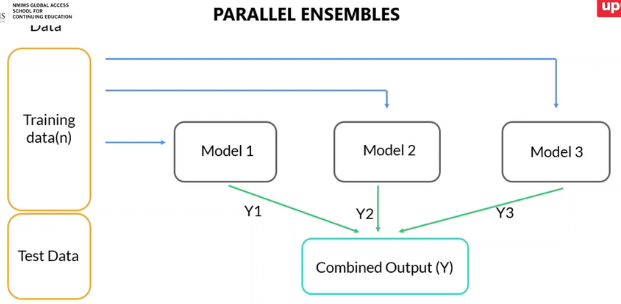
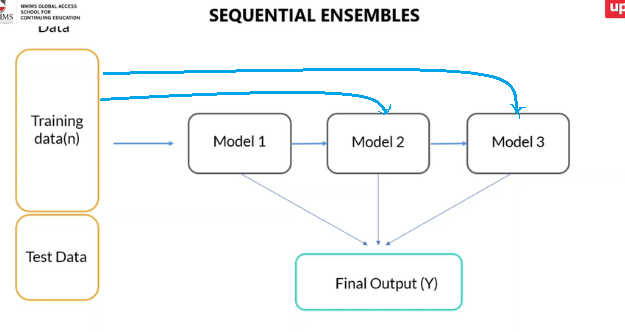


Stacking – Process Flow



**Types of Ensembles**



* ***Ensembles***
  + ***Model Type***
    - ***Homogenous Ensembles:*** Ensemble methods which use same learning algorithm for all base learners, that is, learners of same type are homogenous ensembles. Most ensemble methods are homogenous ensemble methods. The data is sampled (in order to achieve diversity since the learning algorithm is same) into different subsets for homogenous learners.  
      
    - ***Heterogenous Ensembles***: Ensemble methods which use **different learning algorithm** for the base learners. For increasing the accuracy of model, the base learners should be as accurate as possible and as diverse as possible. Heterogenous ensembles add **more diversity** in base learners.  
      
  + ***Algorithm Type***
    - ***Parallel Ensembles***: The ensemble methods in which base learners are independent of each other, that is, they are generated in parallel. Parallel ensembles take advantage of independence between the base learners and significantly reduce error by averaging.  
      
    - ***Sequential Ensembles***: The ensemble methods in which base learners are generated sequentially, that is, each model is generated by learning from the previous models. Sequential ensembles take advantage of dependence between the base learners and improve performance by increasing the weight of previously mislabelled learners.   
      ******

**Parallel Ensembles**

Two models for the course:

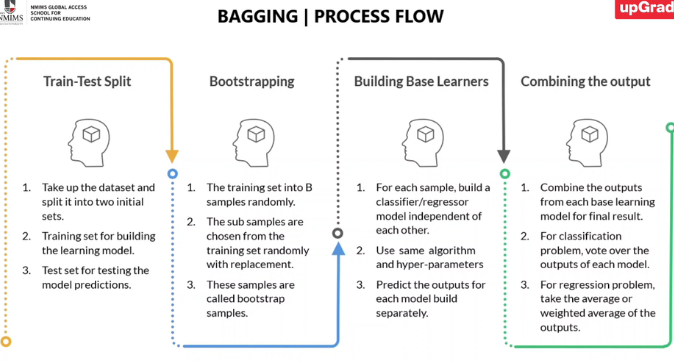
* Bagging: Intuition
* Random Forest: Intuition

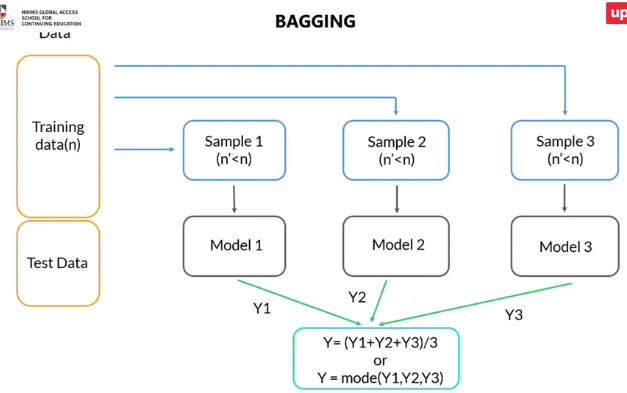
**Bagging: Intuition**

Bagging considers **homogenous** weak learners, learns them independently from each other in **parallel** and combines through **majority voting in classification** and **averaging in regression**. Bagging is also known as **Bootstrap Aggregating**.

Bootstrapping is a statistical technique of generating samples of size B called bootstrap samples from an initial dataset of size N by randomly drawing with replacement B observations. For bootstrap sample, some training examples may appear, but some may not.

Bagging Process





Benefits of Bagging

Works well with high variance algorithms like Decision trees, KNN, SVM, Neural Networks. Bagging uses averaging which significantly reduces the variance. Thus, the problem of overfitting is also resolved.

Easy to parallelise. No dependence on various models in the ensemble.

Faster execution. The models don’t see the entire data due to bootstrapping.

Limitations of Bagging

Loss of interpretability. Example – you can just look at a decision tree and interpret it easily. But in bagging, say you are using 100 decision trees, we can’t look at all of them together.

If some feature dominates, then models end up being similar. Say you build 50 models, but in each of them if the same feature is dominating, then all the models end up being similar which causes a lack of diversity.

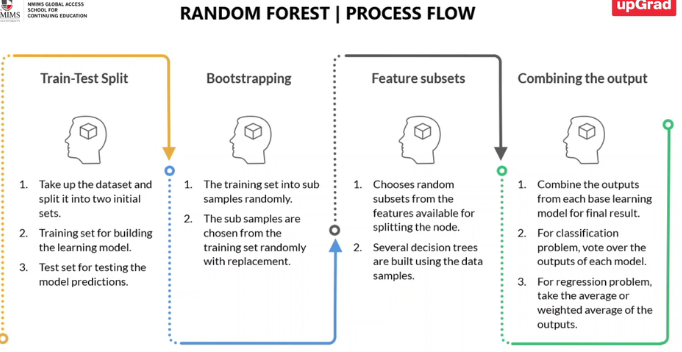
**Random Forest: Intuition**

* Random Forest is a **variant of bagging**. It combines several **decision trees** as a base learner to build an ensemble learner. Random forest brings further diversity by **subsetting the features**. (Random forest is limited to using decision trees. Bagging could use any model).
* Please note that bagging is a general technique and does not pertain to random forests only. On the other hand, random forest is a specific type of bagging algorithm that only uses decision trees as the base learners.
* Random Forest helps in improving diversity by **ALSO subsetting the features**. Note that random subsetting is done at the time of node splitting and not every time a new tree is built.
* In random forest, while building a tree at a particular node, we DON’T consider all the features – we consider a random set of features. By doing this, we are bringing in further diversity.

Why is a Random Forest better than a Decision Tree?

* Decision trees tend to overfit. Random Forest use bagging which reduces overfitting.
* Random Forest introduces more generalization in the model. This is because the individual learners in Random Forest are not only being built on different subsets of data but also different subsets of features and therefore the individual base learners are learning different things.
* Introduces diversity by random sampling of training data points and features.
* Combining the outputs of several decision trees.

Random Forest – Process Flow



**Sequential Ensembles**

**Boosting**: Boosting is a **sequential ensemble** technique which combines several weak learners iteratively to build a strong learner. Each learner at a given iteration depends upon the learner at the previous iteration. The goal is to build a strong learner with **low bias**.

Boosting algorithms are used for combining many weak learners to build a strong learner. A weak learner refers to a model that performs slightly better than a model that makes predictions at random. Weak learners are built sequentially on top of each other, giving a boost to the overall performance of the model.

Here, the sample is NOT bootstrapped. Neither is a subset of features taken i.e., all the features are considered.

Boosting is NOT effective to solve the problem of overfitting.

Different Boosting Algorithms

* Adaptive Boosting (Adaboost)
* Gradient Boosting – Not in Syllabus.
* XgBoost – Not in Syllabus.

**Adaptive Boosting**

Process of Adaptive Boosting:

