

## Linear Regression

So we can use formula to calculate cost function for all points

$$\text{minimize}_{\theta_0, \theta_1} \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

- Where
  - $h_{\theta}(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$
  - $(x^{(i)}, y^{(i)})$  is the  $i^{th}$  training data
  - $m$  is the number of training example
  - $\frac{1}{2}$  is a constant that helps cancel 2 in derivative of the function when doing calculations for gradient descent

So, **cost function** is defined as follows,

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2 = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$



# $R^2$ & ADJUSTED $R^2$



$$R^2 = 1 - \frac{SS_{res}}{SS_{avg}}$$

$$\text{Adjusted } R^2 = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

**Basic assumptions:**

1. Linear relation between dependent and independent variables (otherwise use non-linear models).
2. No or Little multicollinearity
3. Homoscedasticity
4. Normality of residual

**Advantages:**

1. Implementation and interpretation are very easy.
2. Performs well with linearly separable data.
3. Using regularization, dimension reduction technique etc. makes it easier to handle overfitting.
4. Can be used for extrapolation to some degree.

**Disadvantages:**

1. Sensitive to outliers.
2. Prone to noise and overfitting.
3. Not efficient in handling missing values.
4. Sometimes lots of feature engineering is needed.

**Evaluation Metrics:**

1. Mean Squared Error (MSE)
2. Root Mean Squared Error (RMSE)
3. Mean Absolute Error (MAE)
4. R-squared ( $R^2$ )
5. Adjusted R-squared (adjusted  $R^2$ )

## Logistic Regression:

Handwritten notes on a blackboard:

$$J(\theta_1) = -\frac{1}{2m} \sum_{i=1}^m \left[ (y^i \log(h_{\theta}(x^i)) + (1 - y^i) \log(1 - h_{\theta}(x^i))) \right]$$

↓  
cost

$$h_{\theta}(x^i) = \frac{1}{1 + e^{-\theta_1 x}}$$

→ { repeat until convergence

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} (J(\theta_1)).$$

}

Where:

1.  $N$  is the total number of instances.
2.  $y$  is the actual class label (0 or 1) of an instance.
3.  $p$  is the predicted probability of the positive class (output of the sigmoid function) for that instance.

$$\hat{y} = b_0 + b_1 x_1$$

$$\hat{y} = \frac{1}{1 + e^{-x}}$$

$$e = \text{Euler's Number} \sim 2.7182$$

✓ When  $x = \infty$

$$y = \frac{1}{1 + e^{-\infty}}$$

$$= \frac{1}{1 + 0} = 1$$

When  $x = -\infty$

$$y = \frac{1}{1 + e^{+\infty}}$$

$$= \frac{1}{\infty} = 0$$

When  $x = 0$

$$y = \frac{1}{1 + e^0}$$

$$= \frac{1}{1 + 1} = 0.5$$

The process in logistic regression can be summarized as follows:

1. Calculate the linear equation:  $z = mx + c$ , where 'z' is the linear combination of the predictors (independent variables) and their respective coefficients.
2. Apply the sigmoid function to obtain the predicted probabilities:  $p = \text{sigmoid}(z) = 1 / (1 + e^{(-z)})$ .
3. The predicted probabilities can be used directly for classification or further analysis.

The logit function is not explicitly used in the computation of the logistic regression model. Instead, it serves as the mathematical concept that connects the linear equation to the predicted probabilities.

## Assumptions:

1. Logistic regression does not require a linear relationship between the dependant and independent variables. However, it still needs the independent variable to be linearly related to the log-odds (logit function) of the outcome.

When we say that the independent variables should have a linear relationship with the log-odds or logit of the outcome, it means that the relationship between the predictors and the log-odds should be linear when expressed in terms of the coefficients of the logistic regression model.

Mathematically, in logistic regression, the linear relationship is expressed as follows:

$$\log(p / (1 - p)) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$$

Where:

- $\log(p / (1 - p))$  is the logit (log-odds) of the outcome variable.
- $\beta_0, \beta_1, \beta_2, \dots, \beta_p$  are the coefficients of the logistic regression model.
- $x_1, x_2, \dots, x_p$  are the values of the independent variables.

This equation represents the linear combination of the predictors ( $x_1, x_2, \dots, x_p$ ) and their respective coefficients ( $\beta_1, \beta_2, \dots, \beta_p$ ) on the logit scale.

2. Observations should be independent of each other.
3. No multi collinearity among variables.
4. There is a linear relationship between explanatory variables and the logit of the response variable.
5. The sample size sufficiently large enough.

## Advantages:

1. Easy implementation.
2. Can extend to multiple classes (multinomial regression).
3. Performs well with linearly separable dataset.
4. Model allows to be updated relatively easier with new data.

## Disadvantages:

1. High dimensional data may cause overfitting (use regularization techniques).
2. Linearly separable data is rarely found in the real-world scenario.

3. Scaling is required for better convergence hence sometimes lots of feature engineering is needed.

## Evaluation Metrics:

1. **Accuracy (ranges from 0 to 1):** Accuracy measures the overall correctness of the model's predictions by calculating the proportion of correct predictions (both true positives and true negatives) out of the total number of predictions. While accuracy provides a general measure of the model's performance, it can be misleading in cases of imbalanced datasets where the classes are not equally represented.

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

2. **Confusion Matrix:** A confusion matrix provides a tabular representation of the model's predictions against the actual class labels. It includes metrics such as true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). From the confusion matrix, various metrics can be derived, including:

- a. **Precision (ranges from 0 to 1):** Precision calculates the proportion of correctly predicted positive instances (TP) out of the total instances predicted as positive (TP + FP). It measures the model's ability to minimize false positives.

$$\text{Precision} = TP / (TP + FP)$$

- b. **Recall (Sensitivity or True Positive Rate) (ranges from 0 to 1):** Recall calculates the proportion of correctly predicted positive instances (TP) out of the total actual positive instances (TP + FN). It measures the model's ability to minimize false negatives.

$$\text{Recall} = TP / (TP + FN)$$

- c. **Specificity (True Negative Rate) (ranges from 0 to 1):** Specificity calculates the proportion of correctly predicted negative instances (TN) out of the total actual negative instances (TN + FP). It measures the model's ability to minimize false positives in the negative class.

$$\text{Specificity} = TN / (TN + FP)$$

- d. **F1 Score (ranges from 0 to 1):** The F1 score is the harmonic mean of precision and recall. It provides a single metric that balances both precision and recall. The F1 score is useful when there is an uneven distribution between positive and negative instances.

$$\text{F1 Score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$$

3. **Area Under the ROC Curve (AUC-ROC):** The ROC curve (Receiver Operating Characteristic curve) is a graphical representation of the trade-off between the true positive rate (sensitivity) and the false positive rate (1 - specificity) at various classification thresholds. The AUC-ROC summarizes the overall performance of the model by calculating the area under the ROC curve. A higher AUC-ROC indicates better discrimination and classification performance.

4. **Log Loss (Cross-Entropy Loss)**

$$\text{Log Loss} = -(1/N) * \sum (y * \log(p) + (1 - y) * \log(1 - p))$$

Where:

- N is the total number of instances.
- y is the actual class label (0 or 1) of an instance.
- p is the predicted probability of the positive class (output of the sigmoid function) for that instance.

## Difference between Linear and Logistic Regression:

### LINEAR REGRESSION VS LOGISTIC REGRESSION

#### DIFFERENCES IN ASSUMPTIONS

- Logistic regression does not require a linear relationship between the dependent and independent variables. However, it still needs the independent variable to be linearly related to the log-odds of the outcome.
- Homoscedasticity (constant variance) is not required in logistic regression.
- The error terms (residuals) must be normally distributed for linear regression but not required for logistic regression.

#### SIMILARITIES IN ASSUMPTIONS

- Observations should be independent of each other.
- Absence of multicollinearity.

# Decision Tree:

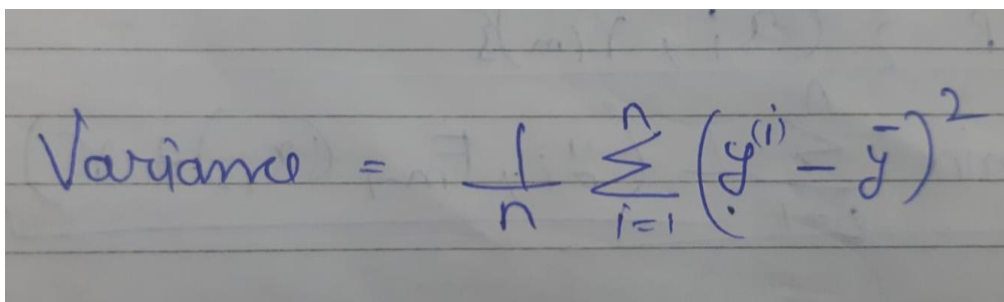
## Classification:

$$\begin{aligned} \text{Entropy}(p) &= - \sum_{i=1}^N p_i \log_2 p_i \\ \text{Gini Impurity}(p) &= 1 - \sum_{i=1}^N p_i^2 \\ \text{Information Gain} &= \text{Entropy}(\text{before}) - \sum_{j=1}^K \text{Entropy}(j, \text{after}) \end{aligned}$$

The purity difference between root node and leaf node is called “**INFORMATION GAIN**”

For numerical columns, we need to find the mean of adjacent values to find the threshold for split.

## Regression:



A photograph of a handwritten formula on lined paper. The word 'Variance' is written in blue ink on the left. To its right is the formula: 
$$\frac{1}{n} \sum_{i=1}^n (y^{(i)} - \bar{y})^2$$

**Variance Reduction: Variance of parent node – (Sum of weighed variance of child nodes)**

In case of regression tree, do need not to find the mean of adjacent values of numerical values. Instead, we can directly use them as a threshold to split.



## Hyperparameters:

1. **'max\_depth'**: Specifies the maximum depth of the decision tree. It limits the number of levels in the tree. Setting a smaller value helps to prevent overfitting, while a larger value can lead to more complex trees.
2. **'min\_samples\_split'**: Specifies the minimum number of samples required to split an internal node. If the number of samples at a node is less than this value, the node will not be split further. It helps to control the tree's tendency to overfit by setting a higher value.
3. **'min\_samples\_leaf'**: Specifies the minimum number of samples required to be at a leaf node. If the number of samples at a leaf node is less than this value, additional splitting is not performed. It helps to prevent overfitting by setting a higher value.
4. **'min\_impurity\_decrease'**: Specifies the minimum decrease in impurity required for a split to occur. It helps to control the quality of splits by considering impurity measures such as Gini impurity or entropy. A higher value results in fewer and more significant splits.
5. **'max\_features'**: Specifies the number of features to consider when looking for the best split. It can be set to a fixed number or a fraction of total features. A smaller value reduces the complexity and potential overfitting of the model.
6. **'criterion'**: Specifies the impurity criterion to measure the quality of a split. For classification, commonly used criteria are "gini" for the Gini impurity and "entropy" for information gain. For regression, "mse" (mean squared error) is often used.

## Advantages:

1. Algorithm is very intuitive and easy to understand.
2. Performs very well with non-linear data.
3. It can handle both numerical and categorical data.
4. Can be used for classification and regression problem.
5. Fast training as no concept of iterating for convergence.
6. Robust to missing values and outliers.

## Disadvantages:

1. Unstable: small change in the data leads to large change in the structure of the tree.
2. Not recommended for large dataset.
3. Prone to overfitting.
4. Impacted by imbalance dataset.

## Random Forest:

### STEP BY STEP APPROACH

1. Create a bootstrap dataset.
2. Create a Decision Tree using the bootstrap dataset with random subset of variables.
3. Now repeat step 1 and step 2 for n times.  
(Given n number for Decision Trees)

### `sklearn.ensemble.RandomForestClassifier`

```
class sklearn.ensemble.RandomForestClassifier(n_estimators=100, *, criterion='gini', max_depth=None,  
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None,  
min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False,  
class_weight=None, ccp_alpha=0.0, max_samples=None)
```

[source]

### HYPERPARAMETERS FOR RANDOM FOREST

- `n_estimators` (number of trees).
- Criterion (Gini or Entropy).
- `max_depth` (Max depth of the tree).
- `min_samples_split` (min value required to split further).
- `max_features` (Maximum number of features used by each tree).
- `bootstrap` (random sample selection).
- `max_samples` (maximum number of samples in each tree).

## ADVANTAGES OF RANDOM FOREST

- Handles non-linear parameters efficiently.
- It can be used to classification and regression both.
- Performs well with categorical data.
- Good at handling missing values.
- No feature scaling needed. Fast training as no concept of iterations for convergence.
- Very stable as compared to Decision Tree.

## DISADVANTAGES OF RANDOM FOREST

- Complexity: due to working with lots of trees.
- Requires larger training time as compared to Decision Tree.
- Computationally expensive.

## Adaboost:

Step by step approach...

- Assigning weights
- Creating stumps (weak learners)
- Choose the best performing stump
- Calculate the weights for each stump
- Adjust & normalize sample weights
- Data Preparing for next stage

### FORMULA TO CHECK STUMP PERFORMANCE

$$\frac{1}{2} \log_e \left( \frac{1 - \epsilon}{\epsilon} \right)$$

### FORMULA TO ASSIGN NEW WEIGHTS

☐ New Sample Weight =  $\text{Weight} \times e^{\text{Performance}}$

☒ New Sample Weight =  $\text{Weight} \times e^{-\text{Performance}}$

## Hyperparameters:

1. **'n\_estimators'**: Specifies the number of weak learners (base estimators) to be iteratively trained. Increasing the number of estimators can improve model performance but also increase training time.
2. **'learning\_rate'**: Controls the contribution of each weak learner to the final ensemble. A smaller learning rate requires more estimators to achieve the same level of accuracy but can make the model more robust to overfitting.
3. **'base\_estimator'**: Specifies the type of weak learner used as the base estimator. For classification tasks, it can be a decision tree classifier, while for regression tasks, it can be a decision tree regressor. Different types of base estimators can be used depending on the problem at hand.
4. **'algorithm'**: Determines the algorithm used to update the weights of samples and adjust their importance. The two options are "SAMME" (Discrete AdaBoost) and "SAMME.R" (Real AdaBoost). "SAMME.R" is a variant that performs better when probability estimates are available.
5. **'random\_state'**: Sets the seed for reproducibility. It ensures that the same sequence of random numbers is generated during training, allowing for consistent results.

## Advantages of AdaBoost:

1. **Improved Accuracy**: AdaBoost is known for its high accuracy in classification tasks. It combines weak classifiers into a strong ensemble, allowing for better generalization and higher accuracy on complex problems.
2. **Handling Complex Data**: AdaBoost can effectively handle complex datasets with overlapping or non-linear decision boundaries. It can adapt to different patterns in the data by giving more weight to misclassified samples, improving the model's performance.
3. **Feature Selection**: AdaBoost automatically selects relevant features by assigning higher weights to them during training. It focuses on informative features, which can enhance the overall predictive power of the ensemble.
4. **Reduced Risk of Overfitting**: By iteratively adjusting the weights of misclassified samples, AdaBoost reduces the risk of overfitting. The algorithm pays more attention to challenging samples, making it less prone to memorizing the training data.

#### Disadvantages of AdaBoost:

1. **Sensitive to Noisy Data and Outliers:** AdaBoost is sensitive to noisy data and outliers. It tends to assign higher weights to misclassified outliers, leading to potential model performance degradation.
2. **Training Time:** The training time of AdaBoost can be higher compared to other algorithms, especially when dealing with large datasets or complex weak classifiers. The algorithm requires sequential training of multiple weak classifiers, which can be computationally expensive.
3. **Vulnerable to Biased Data:** AdaBoost can be influenced by biased datasets, where certain classes or patterns are overrepresented or underrepresented. Biased data can lead to suboptimal model performance and inaccurate predictions.
4. **Parameter Tuning:** AdaBoost has parameters that need to be tuned, such as the number of iterations (stumps) and the learning rate. Selecting the optimal values for these parameters can be challenging and may require cross-validation or other techniques.



# Gradient Boost

Final Prediction = Initial Prediction + Learning Rate \* (Prediction from Tree 1 + Prediction from Tree 2 + ... + Prediction from Tree N)

In this formula:

- Initial Prediction refers to the initial prediction made by the model, which is often a constant value or the mean of the target variable.
- Learning Rate is a hyperparameter that controls the contribution of each weak learner (decision tree) to the final prediction. It is typically a value between 0 and 1. A smaller learning rate reduces the impact of individual weak learners and helps prevent overfitting.
- Prediction from Tree 1, Prediction from Tree 2, ..., Prediction from Tree N represent the individual predictions made by each weak learner (decision tree) in the ensemble. These predictions are weighted by the learning rate and added together to obtain the aggregated prediction.

## Regression:

Initial Prediction = Mean of target feature values

## Classification:

Initial Prediction =  $\log(p / (1 - p))$

Probability =  $1 / (1 + \exp(-\text{Initial Prediction}))$

Or

$$\text{Probability} = \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}}$$

Transformation Formula:

$$\frac{\sum \text{Residual}_i}{\sum [\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)]}$$

Advantages of Gradient Boost algorithm (GBM):

1. **High prediction accuracy:** GBM is known for its ability to achieve high prediction accuracy. By combining multiple weak learners (decision trees) in an ensemble, it can capture complex relationships in the data and produce accurate predictions.
2. **Handles different data types:** GBM can handle a variety of data types, including numerical and categorical features. It can handle missing values in the data and does not require extensive data preprocessing.
3. **Feature importance:** GBM provides a measure of feature importance, allowing you to identify the most influential features in the prediction process. This information can be valuable for feature selection and understanding the underlying data relationships.
4. **Robust to outliers:** GBM is generally robust to outliers in the data. The ensemble approach helps mitigate the impact of individual outliers by aggregating predictions from multiple trees.
5. **Flexibility:** GBM can be used for both regression and classification tasks. It supports various loss functions and can be customized by adjusting hyperparameters to suit different problem domains.

Disadvantages of Gradient Boost algorithm (GBM):

1. **Computationally intensive:** GBM can be computationally expensive and time-consuming, especially when dealing with large datasets and complex models. Training and tuning the algorithm may require substantial computational resources.
2. **Prone to overfitting:** If not properly regularized, GBM is prone to overfitting, especially when the number of weak learners (trees) is large. Regularization techniques like learning rate adjustment and tree depth limitation are crucial to prevent overfitting.
3. **Sensitive to hyperparameters:** GBM performance is sensitive to the selection of hyperparameters, such as learning rate, number of trees, and tree-specific parameters. Finding the optimal combination of hyperparameters can require careful tuning.
4. **Potential for biased predictions:** Gradient Boosting tends to perform well on imbalanced datasets, but it may produce biased predictions towards the majority class if not appropriately addressed through class balancing techniques or appropriate loss functions.
5. **Lack of interpretability:** The final ensemble model of GBM can be complex and challenging to interpret compared to individual decision trees. While feature importance can be estimated, understanding the inner workings of the entire ensemble can be difficult.



## XG Boost

1. The process of Gradient Boost and XG Boost are almost similar. However, unlike Gradient Boost, the information gain is calculated using **SIMILARITY WEIGHT**.

For Regression:

$$\text{Similarity Score} = \frac{(\text{sum of residuals})^2}{\text{sum of residuals} + \lambda}$$

$$\text{Output Value} = \frac{\text{Sum of res}}{\text{no. of res} + 1}$$

$$\text{Prediction} = \text{Initial Pred} + (\text{LR} \times \text{Output Value})$$

$$\text{Gain} = \text{Left}_{ss} + \text{Right}_{ss} - \text{Root}_{ss}$$

### Hyperparameters for XGBoost Regressor:

1. `learning_rate`: Controls the step size shrinkage used during each boosting iteration.
2. `n_estimators`: The number of boosting rounds or trees to build.
3. `max_depth`: Maximum depth of each tree. It controls the complexity of the trees and helps prevent overfitting.
4. `subsample`: Subsample ratio of the training instances. It controls the fraction of samples used for training each tree.
5. `colsample_bytree`: Subsample ratio of columns when constructing each tree. It controls the fraction of features used for training each tree.
6. `alpha`: L1 regularization term on leaf weights. It can help in reducing overfitting.
7. `lambda`: L2 regularization term on leaf weights. It can help in reducing overfitting.
8. `gamma`: Minimum loss reduction required to make a further partition on a leaf node. It controls the complexity of the trees.

### For Clustering:

Similarity Score :

$$\frac{(\sum \text{Residual}_i)^2}{\sum [\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)] + \lambda}$$

COVER

$$\sum [\text{Previous Probability}_i \times (1 + \text{Previous Probability}_i)]$$

$$\frac{(\sum \text{Residual}_i)}{\sum [\text{Previous Probability}_i \times (1 + \text{Previous Probability}_i)] + \lambda}$$

$$\log(\text{odds}) = \log\left(\frac{p}{1-p}\right)$$

$$\log(\text{odds}) \text{ Prediction} = \text{Initial Pred} + (\text{LR} \times \text{Leaf Output})$$

$$\text{Probability} = \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}}$$

$$\text{Gain} = \text{Left}_{ss} + \text{Right}_{ss} - \text{Root}_{ss}$$

In clustering, we have to convert the final prediction to probability using sigmoid function.

Hyperparameters for XGBoost Classifier (in addition to the above):

1. **objective:** The loss function to be optimized during training. It depends on the specific classification task and can be set to various values, such as "binary:logistic" for binary classification or "multi:softmax" for multiclass classification.
2. **num\_class:** The number of classes in the classification problem. It is required when using "multi:softmax" as the objective.
3. **eval\_metric:** The evaluation metric used to assess model performance during training. Common options include "rmse" for regression tasks, "logloss" for binary classification, and "mlogloss" for multiclass classification.

#### Advantages of XGBoost:

1. **High Performance:** XGBoost is highly efficient and scalable, making it suitable for large datasets. It implements parallel processing and tree pruning techniques that significantly speed up the training process.
2. **Excellent Predictive Power:** XGBoost often achieves state-of-the-art performance on a wide range of machine learning problems, including classification, regression, and ranking tasks. It handles complex relationships and non-linear interactions between features well.
3. **Regularization Techniques:** XGBoost provides multiple regularization techniques, such as L1 and L2 regularization, to prevent overfitting. These techniques help in reducing variance and improving the model's generalization ability.
4. **Handling Missing Values:** XGBoost has built-in capabilities to handle missing values in the data. It can learn how to best treat missing values during the training process, reducing the need for explicit data imputation.
5. **Feature Importance:** XGBoost provides a feature importance score, which helps identify the most influential features in the model. This information can aid in feature selection and understanding the importance of different variables.

#### Disadvantages of XGBoost:

1. **Parameter Tuning:** XGBoost has several hyperparameters that need to be tuned to achieve optimal performance. Selecting the right combination of hyperparameters can be challenging and time-consuming, requiring careful experimentation and cross-validation.
2. **Interpretability:** While XGBoost provides excellent predictive power, the resulting models are often considered as "black boxes." Interpreting the inner workings and reasoning behind the predictions can be difficult, especially for complex models.
3. **Memory Usage:** XGBoost uses a gradient boosting framework, which involves creating multiple decision trees. As the number of trees increases, memory usage can become significant, especially for large datasets with many features.
4. **Computational Resources:** XGBoost can be resource-intensive, requiring significant computational power and memory. Training a complex XGBoost model with a large number of trees and features may not be feasible on low-resource systems.
5. **Feature Engineering Dependency:** Like many other machine learning algorithms, the performance of XGBoost heavily relies on the quality of feature engineering. Extracting relevant features and transforming them appropriately can have a significant impact on the model's effectiveness.

## K – Nearest Neighbours

**Classification:** the class is defined according the major votes of nearest K neighbours which is a hyperparameter.

**Regression:** value is defined as the mean of nearest K neighbours which is a hyperparameter.

Nearest Neighbours are considered by calculating either the **Euclidean** or **Manhattan Distances**.

### ADVANTAGES OF K NEAREST NEIGHBOR

- No training period - Algorithm does not require training period as the data itself is a model referencing for the future prediction.
- Easy implementation and interpretation.
- Variety of distance metrics — There is flexibility from the users side to use a distance metric which is best suited for their application (Euclidean, Minkowski, Manhattan distance etc.)
- No assumptions about data – no need to make additional assumptions, tune several parameters, or build a model.
- There is a single hyperparameter, the value of K. This makes hyper parameter tuning easy.

### DISADVANTAGES OF K NEAREST NEIGHBOR

- Does not work well with large datasets or high dimensionality because distance calculating process becomes costlier.
- Feature Scaling- Data in all the dimension should be scaled (normalized and standardized) properly.
- Sensitive to noise, outliers and missing data.
- Poor performance on imbalanced data — When majority of the data the model is being trained on represents 1 label then that label will have a high likelihood of being predicted.
- Optimal value of K — If chosen incorrectly, the model will be under or over fitted to the data



## Naive Bayes Classifier

Bayes Theorem:

$$P(A) * P(B/A) = P(B) * P(A/B)$$
$$P(B/A) = \frac{P(B) * P(A/B)}{P(A)}$$

### ADVANTAGES OF NAÏVE BAYES

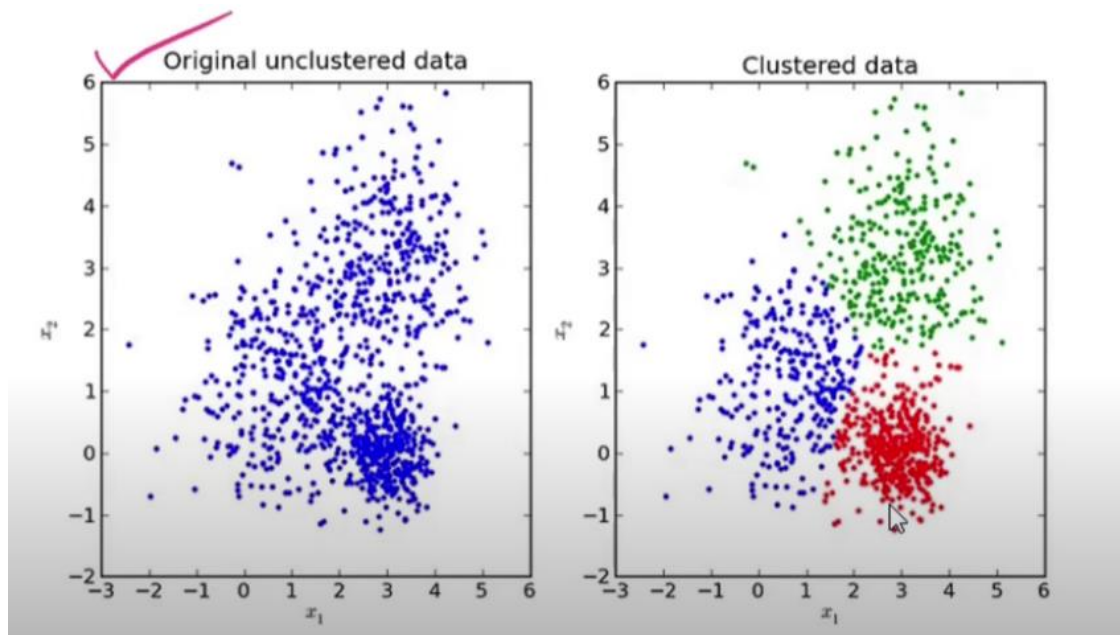
- Algorithm gives every feature the same level of importance, assuming every feature contributes equally to the result.
- Very quick as requires less training time.
- Can be used for binary and multi-class classification both.
- Handles categorical features really well.

### DISADVANTAGES OF NAÏVE BAYES

- Naive Bayes assumes that all predictors (or features) are independent, rarely happening in real life. This limits the applicability of this algorithm in real-world use cases.
- This algorithm faces the 'zero-frequency problem' where it assigns zero probability to a categorical variable whose category in the test data set wasn't available in the training dataset. It would be best if you used a smoothing technique to overcome this issue.
- Does not work well with high dimensional data.

## K Means Clustering

2. Centroids are updated by taking average of all the data points i.e. the average of x coordinates and y coordinates.
3. Using the Elbow curve, the optimal value for K is determined which is the sum of squared Euclidean or Manhattan distance of WITHIN CLUSTER SUM OF SQUARE and then all WCSD of all clusters.



### ADVANTAGES OF K MEANS CLUSTERING

- Easy implementation and interpretation.
- Algorithm is computationally faster than hierarchical clustering.
- Performs well with high dimensional data as well.
- Works well on new data.

## DISADVANTAGES OF K MEANS CLUSTERING

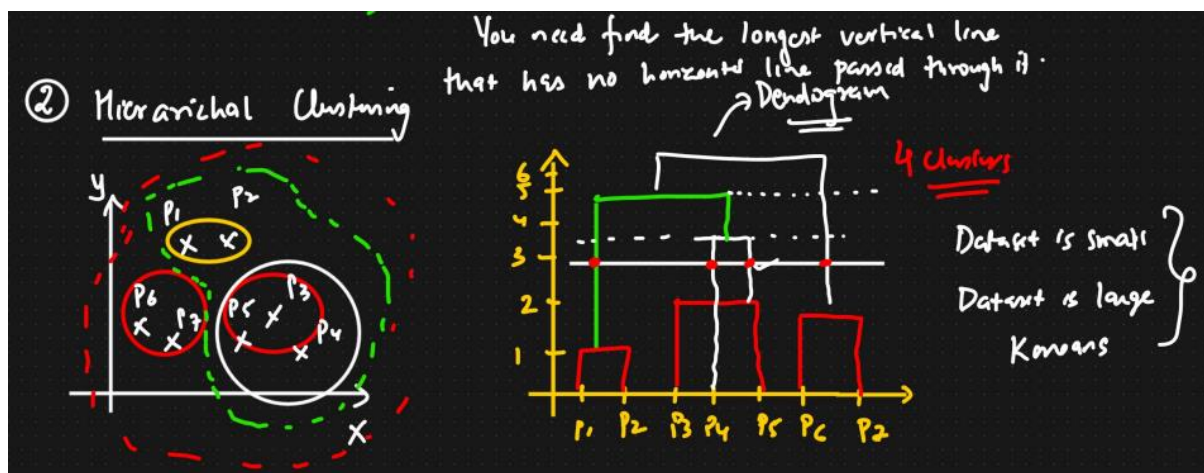
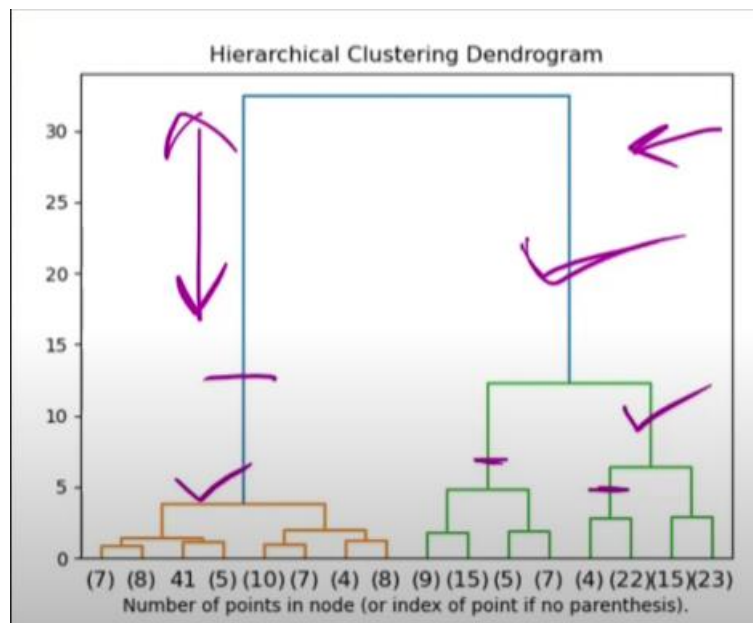
- Value of K has to be chosen manually.
- It struggles to cluster data of varying size and density. Algorithm discovers only spherical clusters.
- It can discover arbitrarily shaped and sized clusters.
- Centroids can be dragged by outliers, or outliers might get their own cluster instead of being ignored. Consider removing or clipping outliers before clustering.



## Hierarchical Clustering

### TYPES OF HIERARCHICAL CLUSTERING

- Agglomerative Clustering
- Divisive Clustering



## ADVANTAGES OF USING HIERARCHICAL CLUSTERING

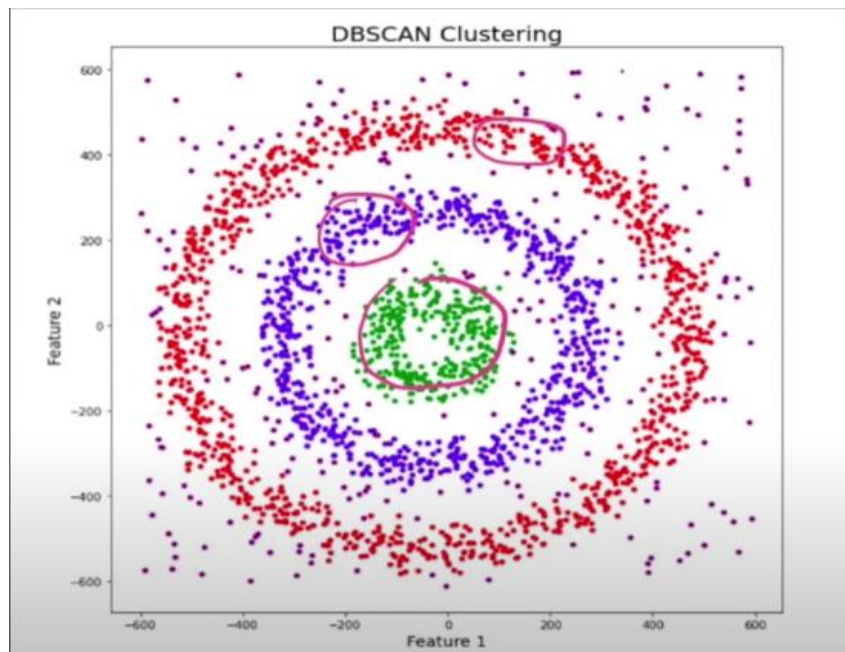
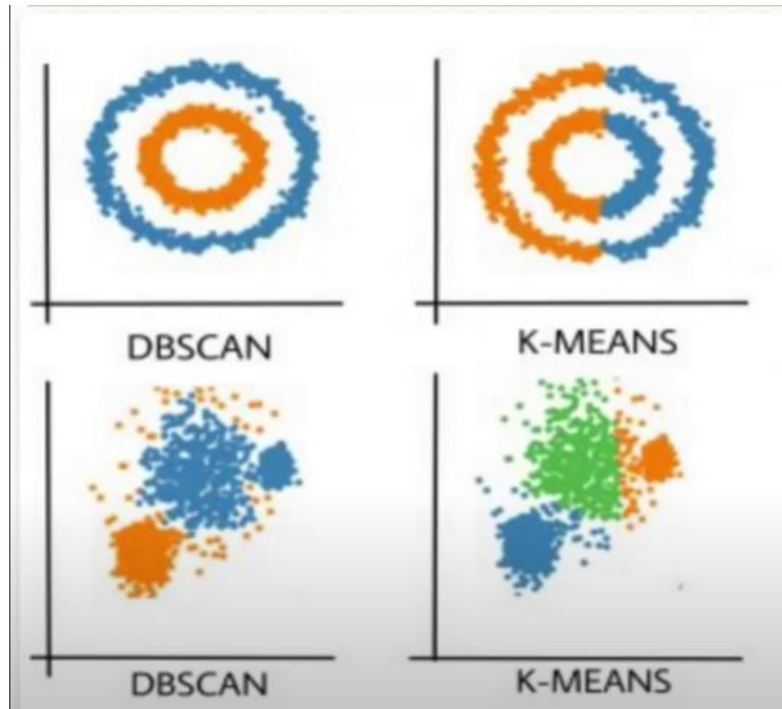
- One major advantage is that it does not require the number of clusters to be specified in advance, as is the case with some other clustering algorithms. This can make it easier to use, especially if you're not sure how many clusters your data should be divided into.
- Another advantage is that hierarchical clustering can provide a visual representation of the data, in the form of a dendrogram. This can make it easier to interpret the results and identify patterns in the data.

## DISADVANTAGES OF USING HIERARCHICAL CLUSTERING

- One major disadvantage is that it can be computationally expensive, especially for large datasets. The algorithm can be slow to converge, and the time required to run the algorithm can increase exponentially with the size of the dataset.
- Another disadvantage is that hierarchical clustering can be sensitive to noise and outliers in the data. If the dataset contains a lot of noise or outliers, it can be difficult to achieve a meaningful clustering result.

# DBSCAN Clustering

Density Based Spatial Clustering Application with Noise



- EPS - Epsilon is the radius from each datapoints that we use to draw the boundary or circle.
- Minimum Points - Minimum Points is the minimum count of datapoints that we want within our core. e.g. 4
- Core - It is the area with (Minimum Points) number of datapoints inside.
- Red Datapoints : Datapoints within the core.
- Yellow Datapoints : Datapoint with atleast 1 core point inside the circle. Circle is called boundary.
- Blue Datapoints : Also referred as noise or outlier with no other datapoint inside its circle.

## ADVANTAGES OF DBSCAN

- No need to specify K value for number of clusters.
- Able to identify noise data while clustering.
- It can discover arbitrarily shaped and sized clusters.

## DISADVANTAGES OF DBSCAN

- Algorithm does not perform well in case of varying density clusters.
- Fails in case of neck types of dataset.
- Does not work well with high dimensional data.