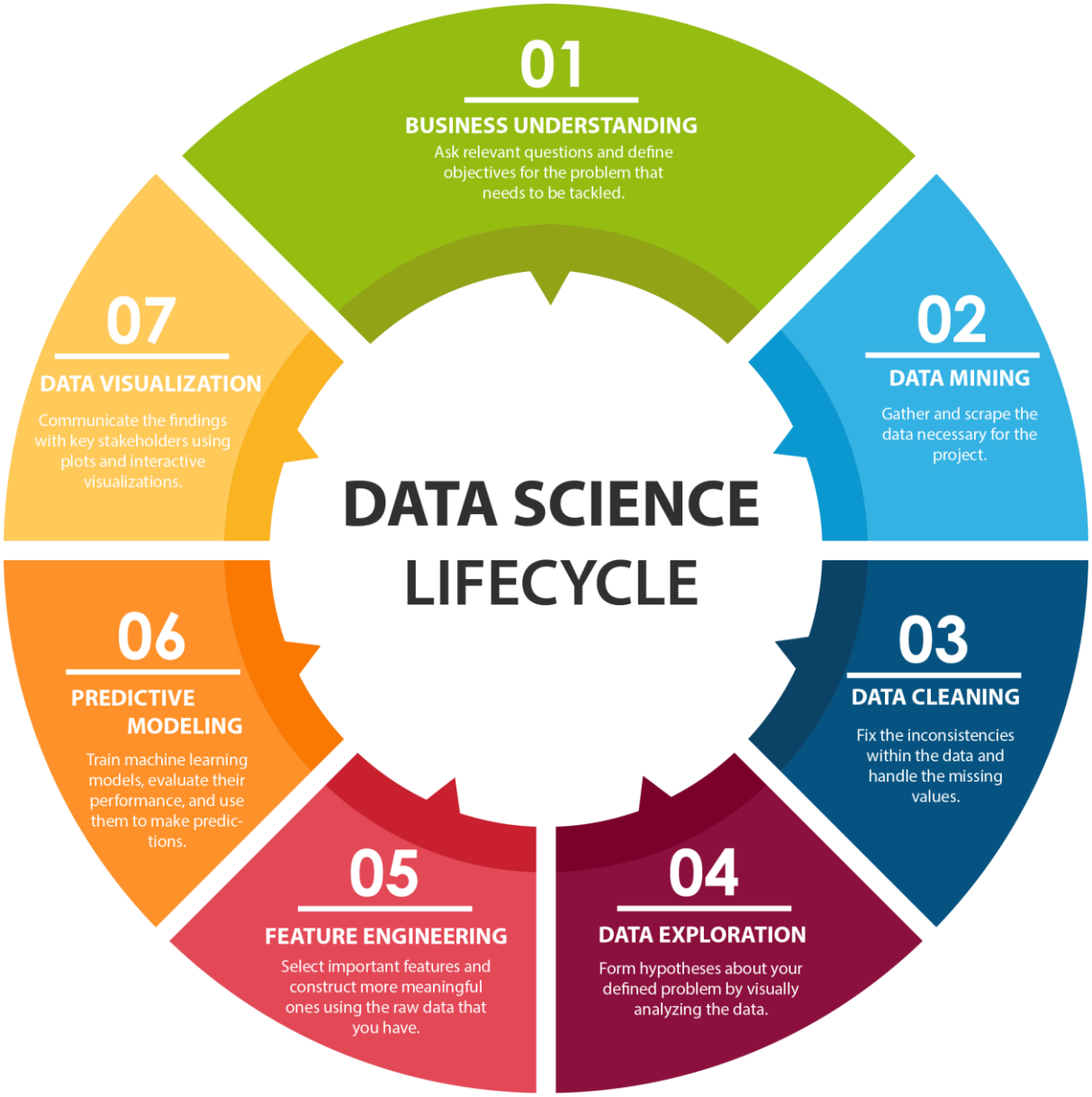
DATA SCIENCE SUMMARY

Utkarsh Gaikwad

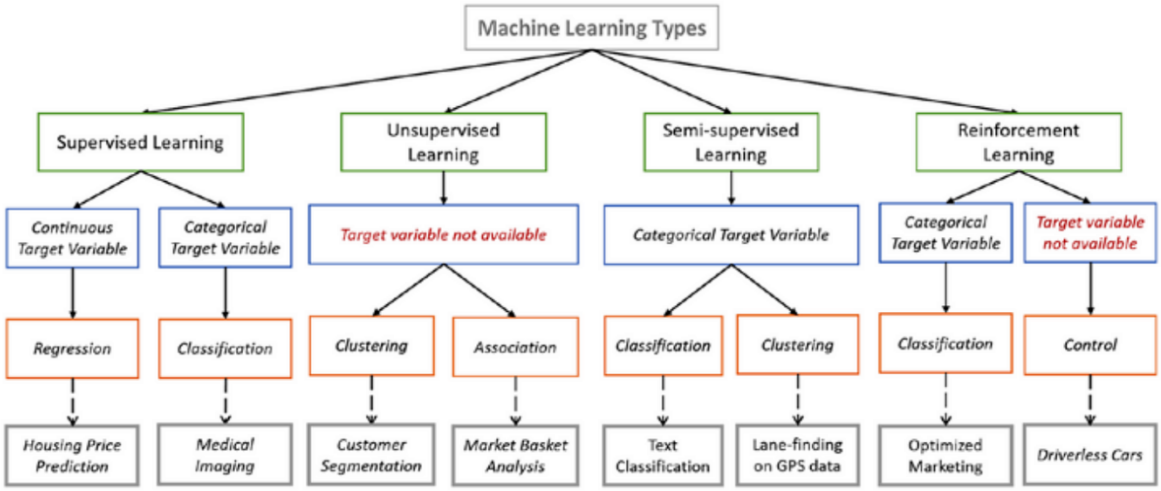
AUGUST 9, 2023

ETLHIVE

Data Science Project Lifecycle



Types of Machine Learning algorithms



Data Pre-processing

For structured data containing both categorical and continuous features, the data preprocessing steps involve handling each type of feature appropriately. Below are the detailed data preprocessing steps for such data:

1. **Data Cleaning:**

- Handle Missing Values: Identify and handle missing values in both categorical and continuous features. For continuous features, you can impute missing values with the mean or median, while for categorical features, you can consider creating a new category to represent missing values or use the most frequent category.

2. **Feature Encoding**:

- One-Hot Encoding: Convert categorical features into numerical form using one-hot encoding. Each category is represented by a binary vector, indicating the presence or absence of that category.

- Label Encoding: Alternatively, you can use label encoding to convert categorical features into integer labels. However, be cautious as this may inadvertently introduce ordinality in non-ordinal categorical variables.

3. **Feature Scaling/Normalization:**

- Scale Continuous Features: Normalize the numerical features to a similar scale using techniques like min-max scaling or standardization (z-score normalization). This ensures that all features contribute equally to the model training process.

4. **Train-Test Split:**

- Split the data into training and testing sets, ensuring that both the training and testing sets have a representative distribution of the target variable.

Feature Selection

A. Feature Selection with Forward Selection:

Forward selection is a feature selection method that starts with an empty feature set and iteratively adds one feature at a time based on its contribution to the model's performance. The process continues until a specified number of features is selected or until a desired model performance is achieved. Here's the step-by-step process of forward selection:

1. Initialization: Start with an empty feature set.

2. Iteration:

- For each feature not in the current feature set, train the model with the current feature set plus the new feature.
- Evaluate the model's performance (e.g., using cross-validation or a validation set).
- Select the feature that results in the best model performance improvement.

3. Add Selected Feature: Add the selected feature to the current feature set.

4. Repeat: Repeat steps 2 and 3 until the desired number of features is selected or the desired model performance is achieved.

B. Feature Selection with Backward Elimination:

Backward elimination is a feature selection method that starts with all features and iteratively removes one feature at a time based on its contribution to the model's performance. The process continues until a specified number of features is selected or until a desired model performance is achieved. Here's the step-by-step process of backward elimination:

1. Initialization: Start with all features.

2. Iteration:

- For each feature in the current feature set, train the model with the feature removed.
- Evaluate the model's performance (e.g., using cross-validation or a validation set).
- Remove the feature that results in the least model performance degradation.

3. Remove Selected Feature: Remove the selected feature from the current feature set.

4. Repeat: Repeat steps 2 and 3 until the desired number of features is selected or the desired model performance is achieved.

Both forward selection and backward elimination are computationally intensive as they involve training the model multiple times for different feature subsets. However, they can be effective for finding a subset of features that contribute most to the model's performance.

C. Feature Selection with Mutual Information Scores:

Mutual information (MI) is a measure of the dependency between two random variables. In the context of feature selection, mutual information scores can be used to quantify the relationship between each feature and the target variable. Features with high mutual information scores have a strong relationship with the target and are likely to be informative for the model.

Here's how to perform feature selection using mutual information scores:

1. Compute Mutual Information: Calculate the mutual information between each feature and the target variable.

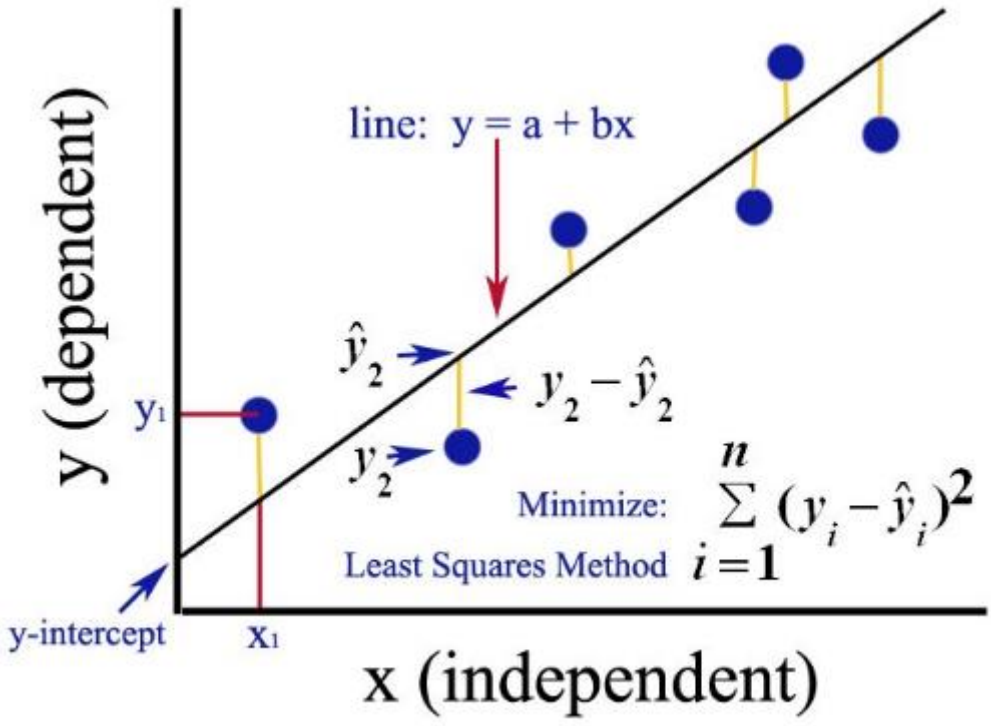
2. Rank Features: Rank the features based on their mutual information scores in descending order.

3. Select Top Features: Select the top-k features with the highest mutual information scores, where k is the desired number of features to be selected.

Mutual information-based feature selection is computationally efficient and does not require training the model iteratively. It can be particularly useful when dealing with high-dimensional datasets, as it quickly identifies the most informative features without explicitly training the model multiple times.

Overall, the choice of feature selection method depends on the complexity of the problem, the size of the dataset, the computational resources available, and the desired model performance. Each method has its advantages and limitations, and it's essential to experiment with different feature selection techniques to find the most suitable one for your specific machine learning task. Backward elimination is a feature selection method that starts with all features and iteratively removes one feature at a time based on its contribution to the model's performance. The process continues until a specified number of features is selected or until a desired model performance is achieved.

Linear Regression



1. **Type:** Supervised

2. **Purpose**: Regression Only

3. **Equations**: $\hat{y} = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \cdots + \beta_n \cdot x_n$
   $\hat{y}$ *is Predicted value of Target Feature (Dependent Feature)*
   β_0 *is intercept* $\beta_1, \beta_2 \ldots, \beta_n$ *are coefficients*
   $x_1, x_2, \ldots, x_n$ *are Independent Features*

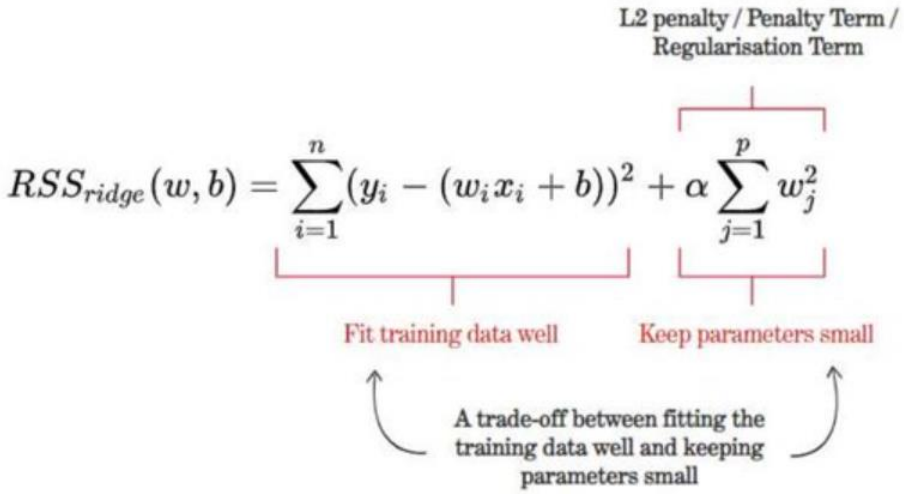
4. **Cost Function:** Values of Intercept and Coefficients are calculated in such a way that the least square error is minimum.
   $J = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$
   J *is called as Cost Function which is to be Minimised in the problem*
   y_i *is Actual values of Target Feature(Dependent Feature)*
   $\hat{y}_i$ *is Predicted values by Linear Regression model*

5. **Advantages:**
   a. **Interpretability**: Linear regression provides easily interpretable results. The model's coefficients represent the relationships between the input features and the target variable. These coefficients show the direction and magnitude of the impact each feature has on the target.

b. **Simple and Fast**: Linear regression is computationally efficient and easy to implement. It does not require complex calculations, making it suitable for large datasets and real-time applications.
c. **No Assumptions on Feature Distribution**: Linear regression does not assume any specific distribution for the input features, making it more flexible and applicable to a wide range of datasets.

6. **Disadvantages:**
   a. **Sensitive to Outliers**: Linear Regression is sensitive to outliers in the data. A single outlier can significantly influence the regression line and affect the model's accuracy.
   b. **Assumes Linearity**: Linear Regression assumes a linear relationship between the input features and the target variable. If the true relationship is nonlinear, the model may not perform well and may lead to inaccurate predictions.
   c. **Multicollinearity**: When features are highly correlated with each other multicollinearity, it becomes challenging to isolate the impact of individual features on the target variable. This can lead to unstable coefficients and reduce the interpretability of the model.

Regularization- Ridge Regression(L2)



1. **Type:** Supervised

2. **Purpose**: Regression Only

3. **Equations**: $\hat{y} = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \cdots + \beta_n \cdot x_n$
   $\hat{y}\ is\ Predicted\ value\ of\ Target\ Feature\ (Dependent\ Feature)$
   $\beta_0\ is\ intercept\ \beta_1, \beta_2 \ldots, \beta_n\ are\ coefficients$
   $x_1, x_2, \ldots, x_n\ are\ Independent\ Features$

4. **Cost Function:** Values of Intercept and Coefficients are calculated in such a way that the least square error is minimum + a penalty term is applied on square of weights
   $$J = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha \cdot \sum \beta_k^2$$
   $J\ is\ called\ as\ Cost\ Function\ which\ is\ to\ be\ Minimised\ in\ the\ problem$
   $y_i\ is\ Actual\ values\ of\ Target\ Feature(Dependent\ Feature)$
   $\hat{y}_i\ is\ Predicted\ values\ by\ Linear\ Regression\ model$
   $\alpha\ is\ Regularization\ Parameter$
   $\beta_k\ are\ coefficents$

5. **Advantages of Ridge Regression:**
   a. **Regularization**: Ridge regression introduces regularization by adding a penalty term to the linear regression cost function. This penalty term helps prevent overfitting by penalizing large coefficients, reducing the risk of model complexity.
   b. **Handles Multicollinearity**: Ridge regression is effective in handling multicollinearity, which occurs when predictor variables are highly correlated with each other. The

regularization term helps stabilize the coefficients, making the model more robust to correlated features.

c. **Stability**: Ridge regression is more stable than ordinary linear regression, especially when dealing with datasets with high dimensionality or a small number of samples.

6. **Disadvantages of Ridge Regression:**
   a. **Not Suitable for Feature Selection**: Ridge regression does not perform feature selection automatically. All features are retained in the model, even if they have little impact on the target variable. For feature selection, other methods like LASSO (L1 regularization) are more appropriate.
   b. **Not Robust to Outliers**: Ridge regression is sensitive to outliers, which can disproportionately influence the regularization term and lead to biased coefficients.

Bias Variance Trade-off

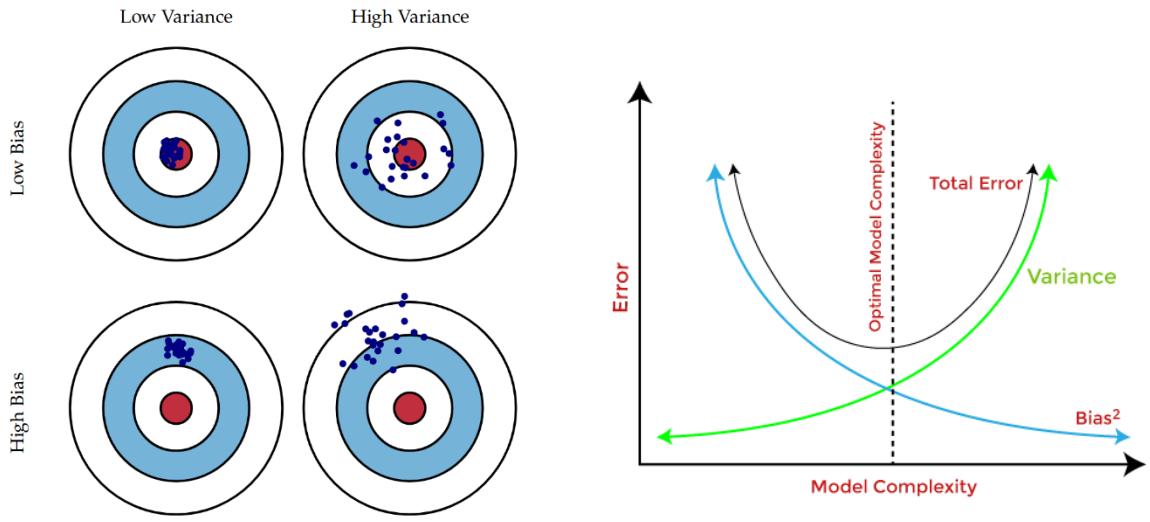


Fig. 1 Graphical illustration of bias and variance.

Regularization Lasso (L1)

$$\sum_{i=1}^{n}(y_i - \sum_{j} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

1. **Type:** Supervised

2. **Purpose**: Regression Only

3. **Equations**: $\hat{y} = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \cdots + \beta_n \cdot x_n$
   $\hat{y}$ *is Predicted value of Target Feature (Dependent Feature)*
   β_0 *is intercept* $\beta_1, \beta_2 \ldots, \beta_n$ *are coefficients*
   $x_1, x_2, \ldots, x_n$ *are Independent Features*

4. **Cost Function:** Values of Intercept and Coefficients are calculated in such a way that the least square error is minimum + a penalty term is applied on absolute value of weights
   $J = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha \cdot \sum |\beta_k|$
   J *is called as Cost Function which is to be Minimised in the problem*
   y_i *is Actual values of Target Feature (Dependent Feature)*
   $\hat{y}_i$ *is Predicted values by Linear Regression model*
   α *is Regularization Parameter*
   β_k *are coefficents*

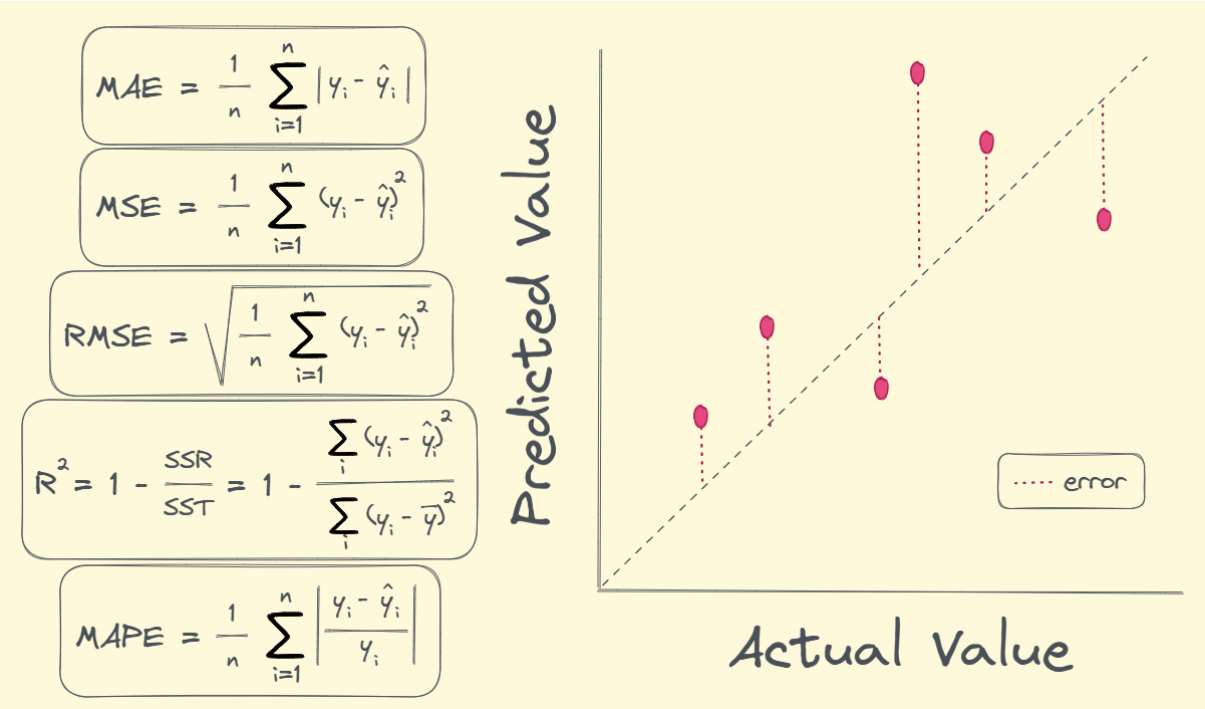
5. **Advantages**:
   a. **Feature Selection**: One of the main advantages of Lasso regression is its inherent feature selection capability. The L1 penalty encourages coefficients of less important features to become exactly zero, effectively excluding them from the model. This results in a more interpretable and potentially simpler model.
   b. **Regularization:** Lasso introduces regularization by adding the absolute values of the coefficients to the cost function. This helps prevent overfitting by controlling the model complexity, especially when dealing with high-dimensional datasets where the number of features is greater than the number of samples.
   c. **Handles Multicollinearity**: Lasso regression can handle multicollinearity (high correlation between features) better than standard linear regression. It tends to select one of the correlated features and shrink the coefficients of the others towards zero, making the model more robust.

6. **Disadvantages**:
    a. **Selection of Penalty Parameter**: The effectiveness of Lasso regression depends on selecting an appropriate value for the penalty parameter (λ). If λ is too small, the model might not achieve effective feature selection, while if λ is too large, it may cause excessive coefficients to be shrunk to zero, resulting in underfitting.
    b. **Sparse Solutions**: While sparsity can be an advantage, it can also be a drawback in some cases. If some of the features that are truly important are suppressed due to the L1 penalty, the model might not capture the full complexity of the underlying relationships in the data.
    c. **Bias in Coefficient Estimates**: Lasso can introduce bias in coefficient estimates when it shrinks the coefficients towards zero. This bias might lead to a trade-off between model interpretability and prediction accuracy.

Regression Metrics

Metrics used to evaluate regression models are as below

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



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

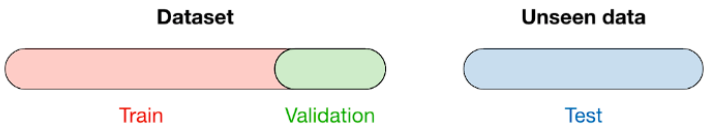
Where
R^2 Sample R-Squared
N Total Sample Size
p Number of independent variable

Cross Validation

❒ **Vocabulary** — When selecting a model, we distinguish 3 different parts of the data that we have as follows:

Training set	Validation set	Testing set
• Model is trained • Usually 80% of the dataset	• Model is assessed • Usually 20% of the dataset • Also called hold-out or development set	• Model gives predictions • Unseen data

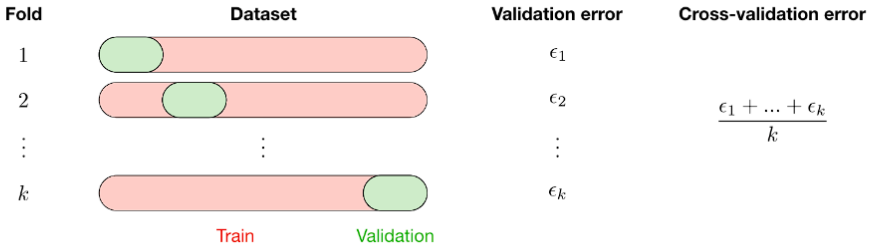
Once the model has been chosen, it is trained on the entire dataset and tested on the unseen test set. These are represented in the figure below:



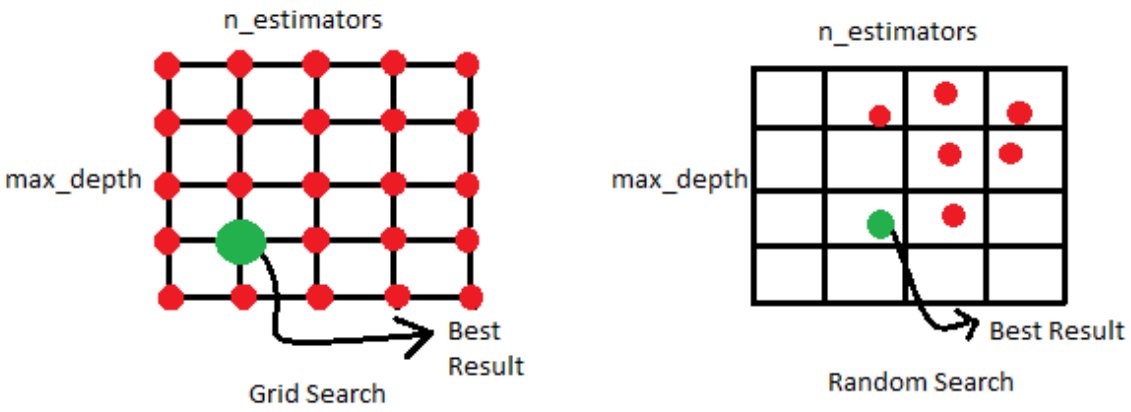
❒ **Cross-validation** — Cross-validation, also noted CV, is a method that is used to select a model that does not rely too much on the initial training set. The different types are summed up in the table below:

k-fold	Leave-p-out
• Training on $k-1$ folds and assessment on the remaining one • Generally $k=5$ or 10	• Training on $n-p$ observations and assessment on the p remaining ones • Case $p=1$ is called leave-one-out

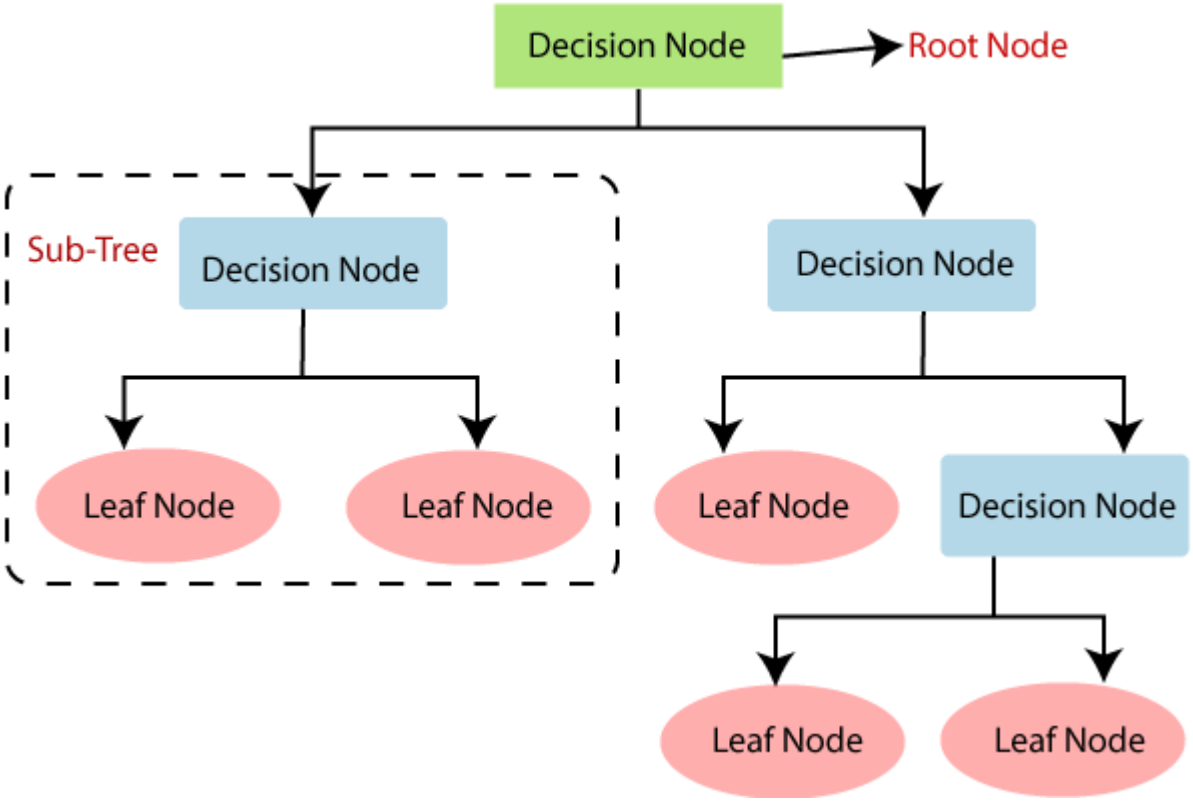
The most commonly used method is called k-fold cross-validation and splits the training data into k folds to validate the model on one fold while training the model on the $k-1$ other folds, all of this k times. The error is then averaged over the k folds and is named cross-validation error.



Hyperparameter Tuning – GridSearchCV and RandomisedSearchCV



Decision Tree



1. **Type:** Supervised
2. **Purpose**: Regression and Classification both
3. **Equations**:

Impurity Criterion

Gini Index

$$I_G = 1 - \sum_{j=1}^{c} p_j^2$$

p_j: proportion of the samples that belongs to class c for a particular node

Entropy

$$I_H = -\sum_{j=1}^{c} p_j log_2(p_j)$$

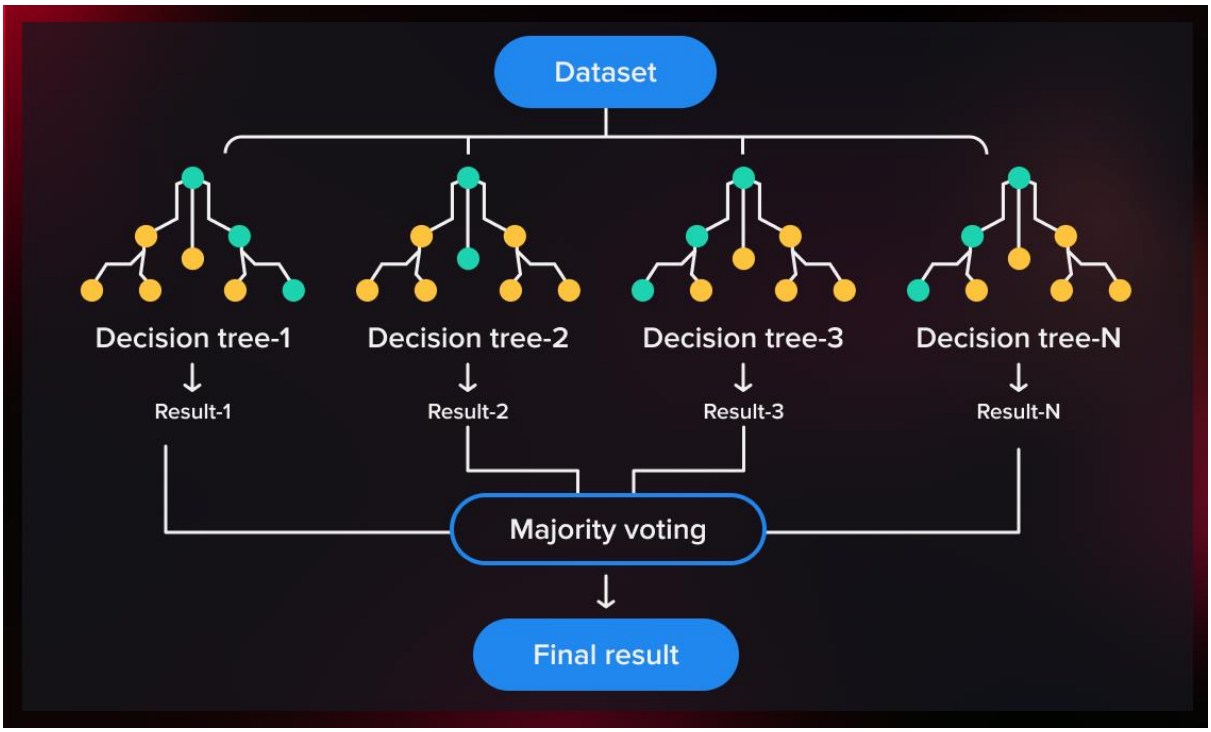
p_j: proportion of the samples that belongs to class c for a particular node.

*This is the the definition of entropy for all non-empty classes ($p \neq 0$). The entropy is 0 if all samples at a node belong to the same class.

4. **Advantages**:
    a. **Interpretability**: Decision trees are highly interpretable. The structure of the tree is easy to understand, as it resembles a flowchart of decisions. This makes it suitable for explaining how decisions are being made to non-technical stakeholders.
    b. **No Assumption of Linearity**: Decision trees can capture non-linear relationships between features and the target variable without requiring any assumptions about the data distribution.
    c. **Handles Both Numeric and Categorical Data**: Decision trees can handle both numeric and categorical data, making them versatile for a wide range of datasets.
    d. **Feature Importance**: Decision trees provide a measure of feature importance, allowing you to understand which features are most influential in making predictions. This can be useful for feature selection or gaining insights into the problem.
    e. **Handles Irregularities**: Decision trees can handle missing values and outliers effectively by creating branches to handle different cases separately.
    f. **Ensemble Methods**: Decision trees serve as building blocks for powerful ensemble methods like Random Forests and Gradient Boosting, which can improve predictive accuracy by combining multiple decision trees.

5. **Disadvantages**:
    a. **Overfitting**: Decision trees are prone to overfitting, especially when they become deep and complex. They can memorize noise in the training data, leading to poor generalization on new, unseen data.
    b. **Instability:** Small changes in the training data can lead to significantly different tree structures. This makes decision trees somewhat unstable and sensitive to minor variations in the data.
    c. **Bias Towards Dominant Classes**: In classification tasks with imbalanced classes, decision trees may be biased towards predicting the majority class since it often results in higher accuracy.
    d. **High Variance**: Single decision trees tend to have high variance, meaning they can give different predictions on similar data if the training set is slightly modified. This is mitigated by ensemble methods.
    e. **Lack of Global Optimality**: The greedy nature of decision tree construction may lead to suboptimal global solutions. At each step, the algorithm chooses the locally optimal feature, which might not result in the best overall tree structure.
    f. **Inefficiency in Handling Large Datasets**: Constructing a decision tree can be computationally expensive, especially when dealing with large datasets. The algorithm may require substantial time and memory resources.
    g. **Bias Towards Rectangular Decision Boundaries**: Decision trees tend to create orthogonal (rectangular) decision boundaries, which might not accurately capture more complex data distributions.

Random Forest



1. **Type:** Supervised – Ensemble Bagging Technique
2. **Purpose**: Regression and Classification both
3. **Equations**:

Impurity Criterion

Gini Index

$$I_G = 1 - \sum_{j=1}^{c} p_j^2$$

p_j: proportion of the samples that belongs to class c for a particular node

Entropy

$$I_H = -\sum_{j=1}^{c} p_j log_2(p_j)$$

p_j: proportion of the samples that belongs to class c for a particular node.

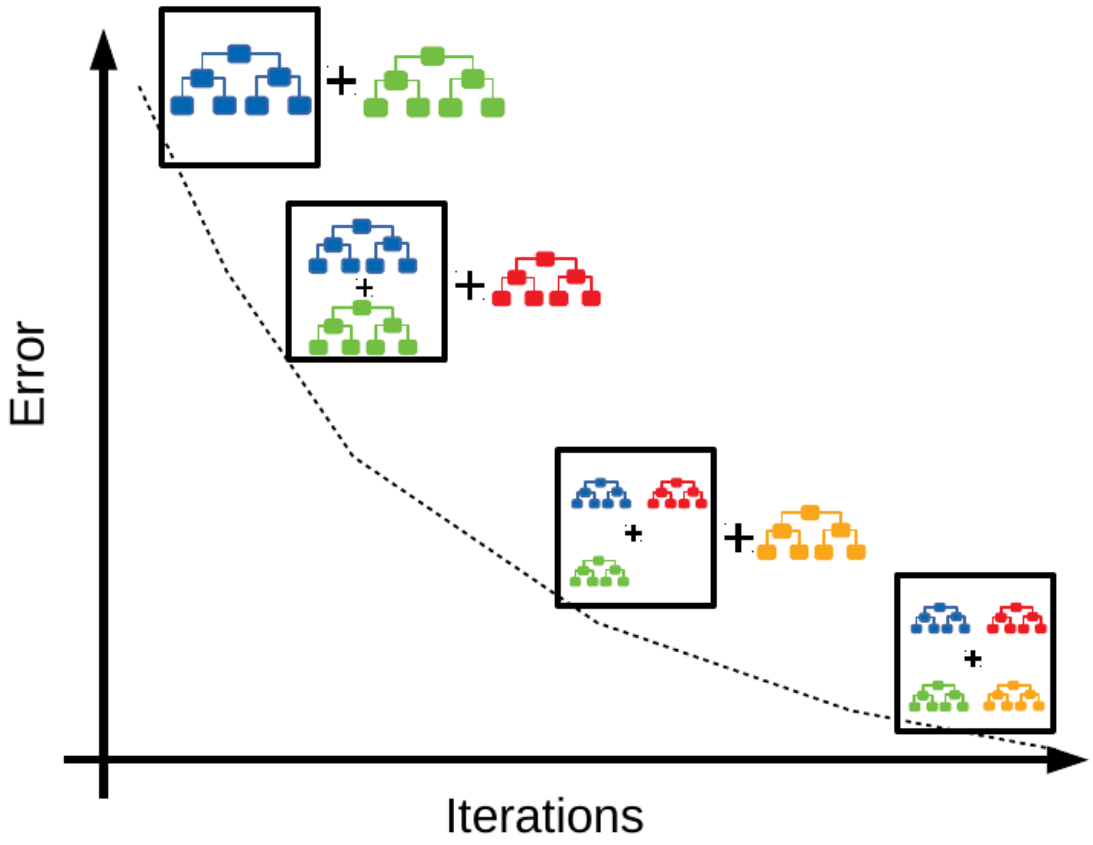
*This is the the definition of entropy for all non-empty classes ($p \neq 0$). The entropy is 0 if all samples at a node belong to the same class.

Random Forest is an ensemble learning technique that combines multiple decision trees to improve predictive accuracy and control overfitting. Here are some advantages and disadvantages of Random Forest:

4. **Advantages**:
   a. **Improved Predictive Accuracy**: Random Forest usually provides better predictive performance compared to a single decision tree, as it reduces overfitting by averaging out the predictions of multiple trees.
   b. **Reduced Overfitting**: By aggregating predictions from multiple decision trees, Random Forest reduces the risk of overfitting that can occur when using a single complex tree.
   c. **Feature Importance**: Random Forest provides a measure of feature importance based on how often a feature is used across all the trees. This can help in understanding which features are most relevant for making predictions.
   d. **Robustness to Outliers and Noisy Data**: The majority vote or averaging of predictions in Random Forest reduces the impact of outliers and noisy data points, resulting in more stable predictions.
   e. **Less Prone to Overfitting than Decision Trees**: While individual decision trees can be prone to overfitting, the ensemble nature of Random Forest makes it less susceptible to this issue.
   f. **Efficient for Large Datasets**: Random Forest can handle large datasets with numerous features more efficiently than trying to build and manage a single complex model.

5. **Disadvantages**:
   a. **Black-Box Nature**: While Random Forest offers better performance, the ensemble approach can make it more challenging to interpret compared to a single decision tree.
   b. **Computationally Intensive**: Constructing multiple decision trees and aggregating their results requires more computational resources and time compared to training a single decision tree.
   c. **Memory Consumption**: Storing multiple trees and their associated data structures can lead to higher memory consumption, especially when dealing with a large number of trees.
   d. **Loss of Interpretability**: While feature importance can be measured, understanding the specific decision logic of each individual tree within the Random Forest can be complex.
   e. **Hyperparameter Tuning**: Random Forest has hyperparameters that need to be tuned, such as the number of trees, maximum depth, and minimum samples per leaf. Tuning these hyperparameters correctly is crucial for optimal performance.
   f. **Not Ideal for Linear Relationships**: Random Forest is designed to capture complex relationships in the data. If the relationships are linear, simpler models like linear regression might perform better and be more interpretable.

Gradient Boosting



1. **Type:** Supervised – Ensemble Boosting Technique
2. **Purpose**: Regression and Classification both
3. **Equations**:

Impurity Criterion

Gini Index

$$I_G = 1 - \sum_{j=1}^{c} p_j^2$$

p_j: proportion of the samples that belongs to class c for a particular node

Entropy

$$I_H = -\sum_{j=1}^{c} p_j log_2(p_j)$$

p_j: proportion of the samples that belongs to class c for a particular node.

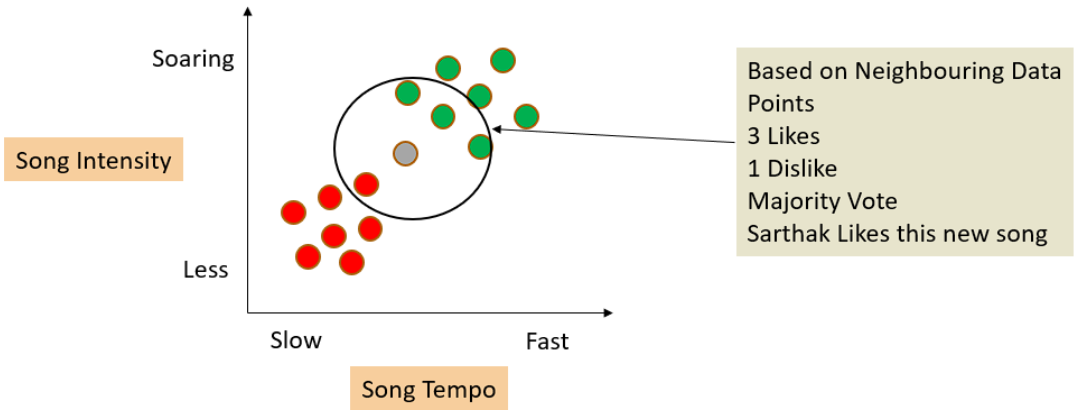
*This is the the definition of entropy for all non-empty classes ($p \neq 0$). The entropy is 0 if all samples at a node belong to the same class.

Gradient Boosting is a powerful ensemble learning algorithm that combines the strengths of multiple weak learners (usually decision trees) to create a strong predictive model. Here are some advantages and disadvantages of the Gradient Boosting algorithm:

4. **Advantages**:
   a. **High Predictive Accuracy**: Gradient Boosting often provides superior predictive accuracy compared to individual decision trees and many other machine learning algorithms. It learns from the mistakes of previous models and focuses on improving where they went wrong.
   b. **Handles Non-linearity**: Gradient Boosting can capture complex non-linear relationships in the data without requiring manual feature engineering.
   c. **Feature Importance**: Like Random Forest, Gradient Boosting provides feature importance scores, helping you identify which features contribute the most to the model's predictions.
   d. **Handles Missing Data**: Gradient Boosting can handle missing data in a flexible manner, imputing missing values in a way that aligns with the learning process.
   e. **Outlier Robustness**: Gradient Boosting is robust to outliers due to the ensemble nature. Outliers in individual trees have a smaller impact on the overall model's predictions.
   f. **Efficient Handling of Large Datasets**: Gradient Boosting can effectively handle large datasets without significant memory and computational overhead.

5. **Disadvantages**:
   a. **Computationally Intensive**: Gradient Boosting can be computationally expensive, especially if the dataset is large and the boosting process involves a high number of iterations.
   b. **Prone to Overfitting**: Without proper hyperparameter tuning and regularization, Gradient Boosting can overfit the training data, leading to poor generalization to new, unseen data.
   c. **Hyperparameter Sensitivity**: There are several hyperparameters that need to be tuned, such as the learning rate, number of trees, and tree depth. Finding the right combination can be challenging.
   d. **Black-Box Nature**: Just like Random Forest, Gradient Boosting can become less interpretable compared to single decision tree

K-Nearest-Neighbours



1. **Type:** Supervised

2. **Purpose**: Regression and Classification both

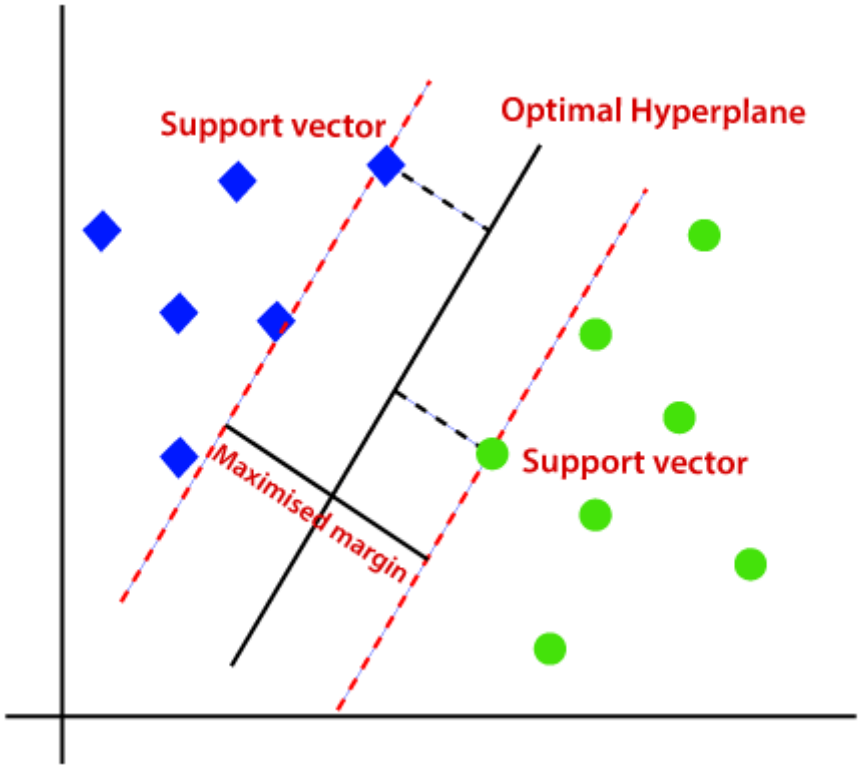
3. **Logic**:
   a. For Regression results are calculated by taking average of nearest neighbours.
   b. For Classification results are calculated by majority vote of classes.

4. **Advantages**:
   a. **Simplicity**: KNN is straightforward and easy to understand. It does not require complex mathematical concepts to implement.
   b. **No Training Phase**: KNN does not require a training phase like other algorithms. It stores the training data and makes predictions directly on new data points.
   c. **Non-parametric**: KNN is non-parametric, which means it does not make any assumptions about the underlying data distribution. This makes it suitable for data with complex patterns.
   d. **Flexibility**: KNN can be used for both classification and regression tasks, making it versatile for various types of predictive modelling.
   e. **Local Patterns**: KNN considers local patterns in the data, which can be useful when the decision boundaries are not linear or easily separable.
   f. **Adaptability to Data Changes**: KNN can adapt to changes in the data over time, making it suitable for scenarios where the data distribution may change.

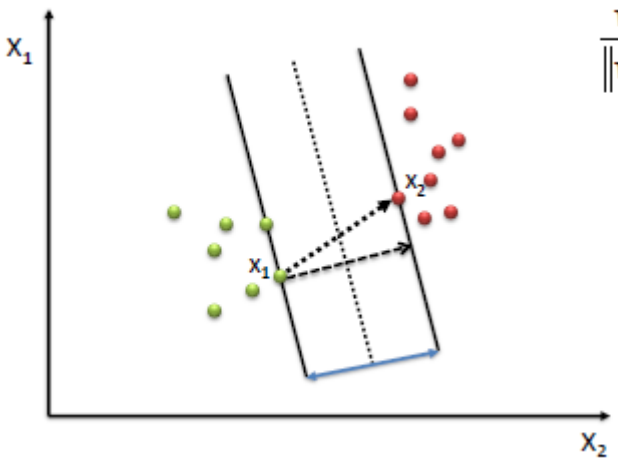
5. **Disadvantages**:
   a. **Computational Complexity**: As the dataset grows, the computational cost of finding nearest neighbors for each prediction can become high, leading to slower prediction times.
   b. **Choice of K**: Selecting the appropriate value of K (the number of neighbors) is crucial. A small K can lead to noisy predictions, while a large K can smooth out decision boundaries and miss local patterns.

c. **Sensitive to Noise and Outliers**: KNN can be sensitive to noisy data and outliers. Outliers can disproportionately affect predictions, especially when K is small.
d. **Memory Intensive**: KNN stores the entire training dataset in memory, which can be memory-intensive for large datasets. This can limit its scalability.
e. **Curse of Dimensionality**: KNN performance can degrade in high-dimensional feature spaces due to the "curse of dimensionality," where distance-based similarity measures become less meaningful in higher dimensions.
f. **Imbalanced Data**: In datasets with imbalanced class distributions, KNN tends to favour the majority class, leading to biased predictions.
g. **Slow for Large Datasets**: Calculating distances between data points becomes computationally expensive in large datasets, leading to slower prediction times.
h. **No Feature Importance**: KNN does not provide feature importance or model interpretability inherently. It may not be suitable if understanding the importance of different features is crucial.

Support Vector Machine (SVM)



1. **Type:** Supervised
2. **Purpose**: Regression and Classification both
3. **Equations:**



$$\frac{w}{\|w\|}\cdot(x_2-x_1)=\text{width}=\frac{2}{\|w\|}$$

$$w\cdot x_2+b=1$$
$$w\cdot x_1+b=-1$$
$$w\cdot x_2+b-w\cdot x_1-b=1-(-1)$$
$$w\cdot x_2-w\cdot x_1=2$$
$$\frac{w}{\|w\|}(x_2-x_1)=\frac{2}{\|w\|}$$

Support Vector Machines (SVM) is a powerful and versatile supervised machine learning algorithm used for classification and regression tasks. It aims to find a hyperplane that best separates data points of different classes while maximizing the margin between the classes. Here are some advantages and disadvantages of the SVM algorithm:

4. **Advantages:**
   a. **Effective in High-Dimensional Spaces**: SVM performs well in high-dimensional feature spaces, making it suitable for datasets with many features, including text and image data.
   b. **Robust to Overfitting**: SVM aims to maximize the margin between classes, which helps in reducing overfitting by promoting a simpler decision boundary.
   c. **Global Optimum**: SVM aims to find the hyperplane that maximizes the margin between classes, resulting in a global optimum solution that generalizes well to new data.
   d. **Kernel Trick**: SVM can efficiently handle non-linear decision boundaries by using kernel functions, which implicitly map the data into higher-dimensional spaces where they can be linearly separated.
   e. **Works Well with Small Datasets**: SVM performs well even when the dataset is small, as it focuses on the support vectors closest to the decision boundary.
   f. **Feature Importance**: SVM provides information about the support vectors, which can help identify important features contributing to the decision boundary.

5. **Disadvantages**:
   a. **Computational Complexity**: Training an SVM can be computationally expensive, especially with large datasets. The time complexity depends on the number of data points and the dimensionality of the feature space.
   b. **Sensitivity to Noise**: SVM is sensitive to noisy data, outliers, and mislabelled examples, which can impact the location of the decision boundary and margin.
   c. **Choice of Kernel and Parameters**: Selecting the appropriate kernel function and tuning hyperparameters can be challenging. The choice of the kernel function and parameters can significantly affect the performance of the SVM.
   d. **Memory Intensive**: In the case of large datasets, SVM can be memory-intensive, particularly when dealing with non-linear kernels that require the storage of kernel matrices.
   e. **Limited Interpretability**: The decision boundary generated by SVM can be difficult to interpret, especially when non-linear kernels are used. The model does not provide direct insights into the importance of individual features.
   f. **Class Imbalance**: SVM can be biased towards the majority class in imbalanced datasets, leading to suboptimal performance on the minority class.
   g. **Slow Prediction**: The prediction phase can be slower compared to other algorithms, especially for large datasets, as it requires calculating distances from new data points to support vectors.

Classification Metrics

Classification metrics

In a context of a binary classification, here are the main metrics that are important to track in order to assess the performance of the model.

❒ **Confusion matrix** — The confusion matrix is used to have a more complete picture when assessing the performance of a model. It is defined as follows:

		Predicted class +	**Predicted** class -
Actual class	+	**TP** True Positives	**FN** False Negatives Type II error
	-	**FP** False Positives Type I error	**TN** True Negatives

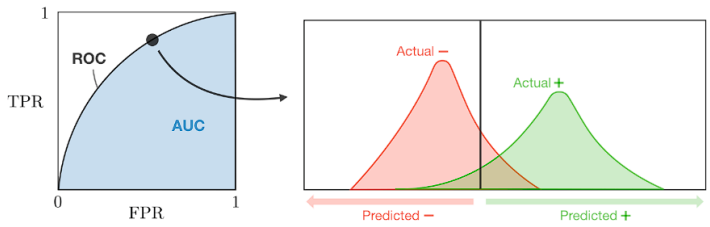
❒ **Main metrics** — The following metrics are commonly used to assess the performance of classification models:

Metric	Formula	Interpretation
Accuracy	$\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$	Overall performance of model
Precision	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$	How accurate the positive predictions are
Recall Sensitivity	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$	Coverage of actual positive sample
Specificity	$\frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}$	Coverage of actual negative sample
F1 score	$\frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$	Hybrid metric useful for unbalanced classes

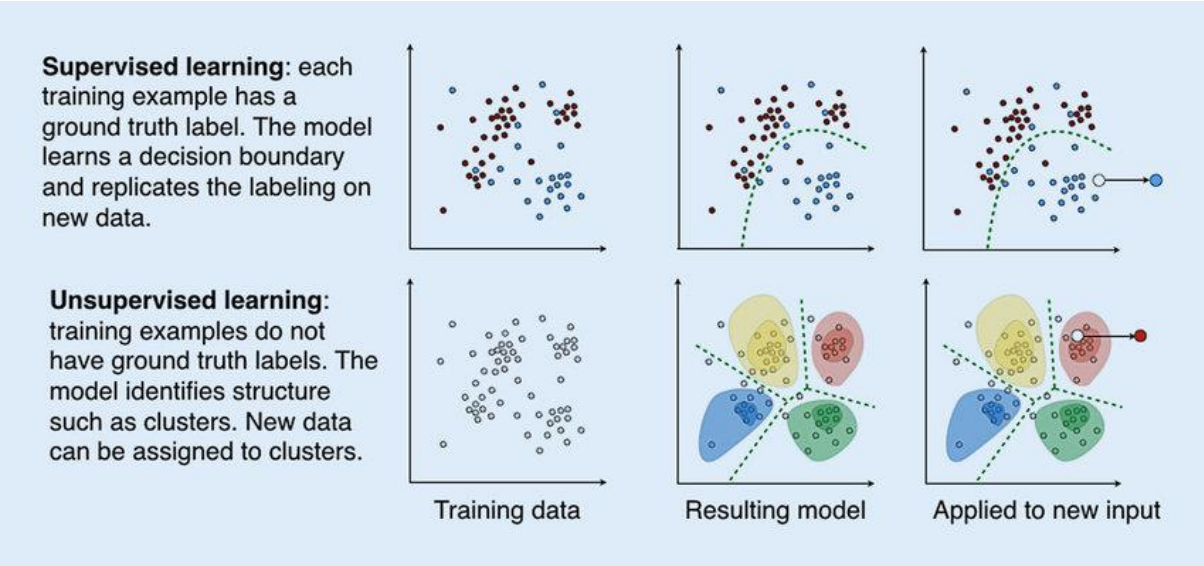
❒ **ROC** — The receiver operating curve, also noted ROC, is the plot of TPR versus FPR by varying the threshold. These metrics are are summed up in the table below:

Metric	Formula	Equivalent
True Positive Rate TPR	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$	Recall, sensitivity
False Positive Rate FPR	$\frac{\mathrm{FP}}{\mathrm{TN}+\mathrm{FP}}$	1-specificity

❒ **AUC** — The area under the receiving operating curve, also noted AUC or AUROC, is the area below the ROC as shown in the following figure:



Unsupervised Machine Learning

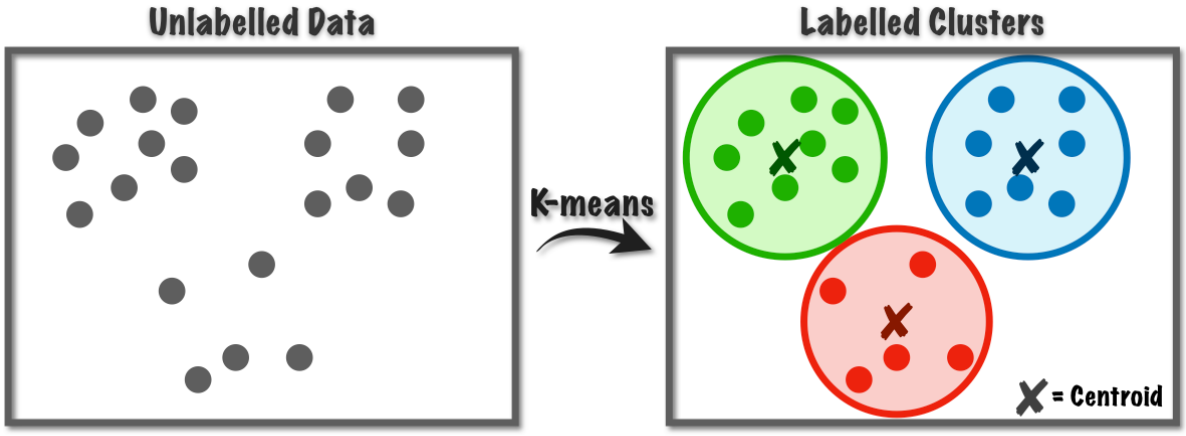


Unsupervised machine learning is a branch of machine learning where the goal is to find patterns, structures, and relationships within a dataset without explicit guidance from labelled outputs. Unlike supervised learning, where the algorithm learns from labelled examples, unsupervised learning involves working with unlabelled data and focuses on uncovering inherent structures within the data.

Unsupervised learning has several applications across various fields:

1. **Market Segmentation**: Clustering can be used to segment customers into groups with similar behaviours or preferences, aiding in targeted marketing strategies.
2. **Anomaly Detection**: Identifying outliers or anomalies within data can be crucial in fraud detection, network security, or quality control.
3. **Topic Modelling**: Unsupervised learning can help uncover topics within text data, making it useful for tasks like document categorization or sentiment analysis.
4. **Image Compression**: Dimensionality reduction techniques can be used to reduce the size of image data while preserving important features.
5. **Genomics**: Clustering can help identify patterns in gene expression data, leading to insights about disease subtypes and genetic relationships.
6. **Recommendation Systems**: Unsupervised learning can be used to group users with similar preferences, aiding in building personalized recommendation systems.
7. **Data Preprocessing**: Unsupervised techniques can be used to preprocess and clean data, removing redundant features and improving the quality of input for subsequent analysis.

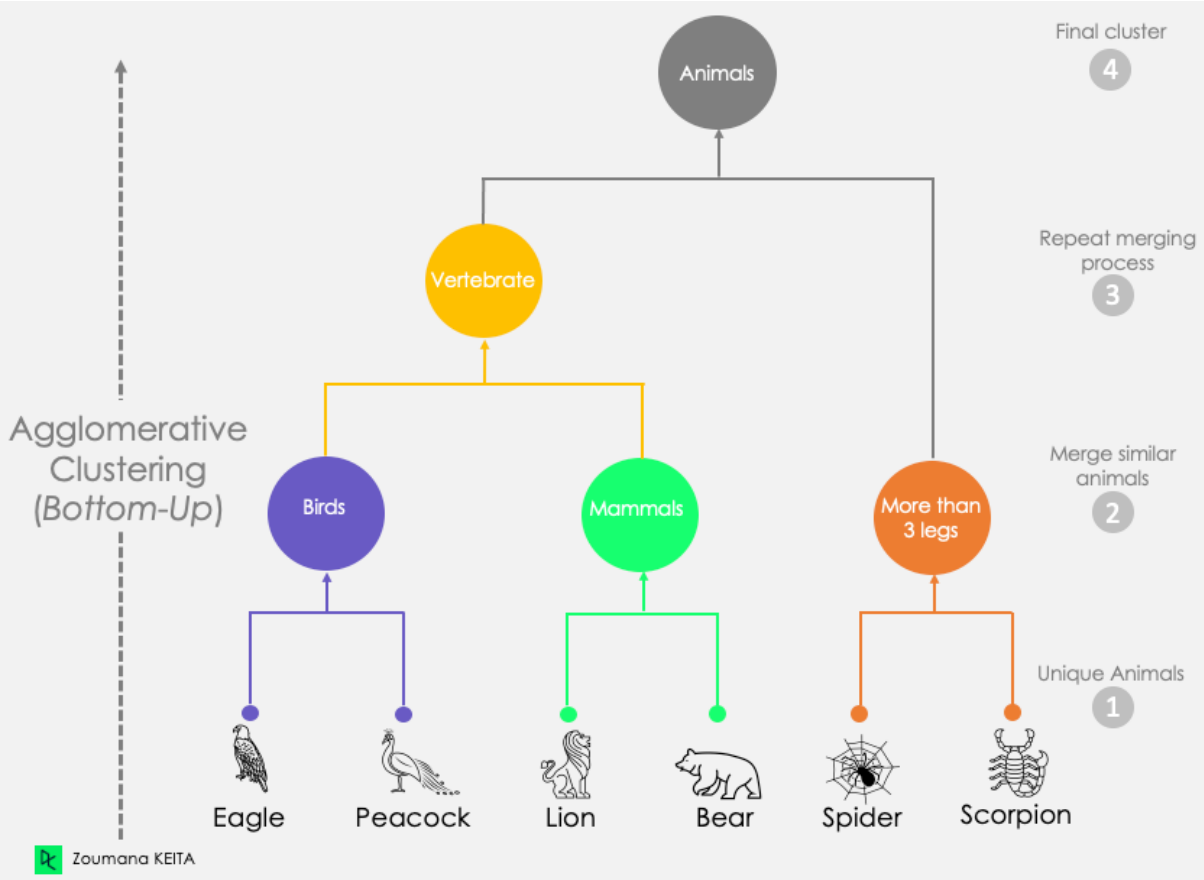
K-means Clustering



K-Means clustering is a popular unsupervised machine learning algorithm used for partitioning a dataset into distinct groups or clusters based on similarity patterns in the data. The algorithm aims to find K centroids, each representing a cluster, and assigns data points to the nearest centroid. It iteratively refines the centroids until convergence, minimizing the sum of squared distances between data points and their assigned centroids. Here is a brief summary of how K-Means clustering works:

1. **Initialization**: Choose K initial centroids either randomly or using a specific strategy, such as selecting data points from the dataset.
2. **Assignment Step**: Assign each data point to the nearest centroid. This is done by calculating the distance (usually Euclidean distance) between the data point and each centroid and assigning the data point to the cluster of the closest centroid.
3. **Update Step**: Recalculate the centroids of each cluster by computing the mean of all data points assigned to that cluster.
4. **Iteration**: Repeat the assignment and update steps iteratively until the centroids stabilize (convergence) or a specified number of iterations is reached.
5. **Final Clusters**: Once the algorithm converges, the final clusters are formed based on the centroids.

Hierarchical Clustering



Hierarchical Clustering is an unsupervised machine learning algorithm used to create a hierarchical representation of data by iteratively merging or dividing clusters based on similarity. It builds a tree-like structure (dendrogram) that shows the relationships between data points and clusters at different levels of granularity. Here is a concise summary of how Hierarchical Clustering works:

1. **Initialization**: Begin with each data point as its own individual cluster.
2. **Distance Calculation**: Compute a distance or dissimilarity metric between pairs of clusters or data points. Common metrics include Euclidean distance, Manhattan distance, or correlation.
3. **Merging or Dividing**: Merge the two closest clusters or divide an existing cluster into smaller clusters based on the distance metric. This creates a hierarchy of clusters.
4. **Dendrogram Construction**: Represent the merging/dividing process as a dendrogram, where the vertical axis represents the distance and the horizontal axis represents the data points or clusters.
5. **Repeat**: Continue merging/dividing until all data points belong to a single cluster or until a predefined stopping criterion is met.
6. **Cutting the Dendrogram**: To determine the number of clusters, you can cut the dendrogram at a certain height. Each horizontal line intersected by the dendrogram corresponds to a potential number of clusters.

Association Rule Mining

Transaction 1	🍎 🍺 🍚 🍗
Transaction 2	🍎 🍺 🍚
Transaction 3	🍎 🍺
Transaction 4	🍎 🍐
Transaction 5	🍼 🍺 🍚 🍗
Transaction 6	🍼 🍺 🍚
Transaction 7	🍼 🍺
Transaction 8	🍼 🍐

$$Support = \frac{frq(X,Y)}{N}$$

$$Rule: X \Rightarrow Y \rightarrow Confidence = \frac{frq(X,Y)}{frq(X)}$$

$$Lift = \frac{Support}{Supp(X) \times Supp(Y)}$$

Association Rule Mining is a data mining technique used to discover interesting relationships or patterns within large datasets. The Apriori algorithm is one of the most well-known algorithms for association rule mining. It focuses on finding frequent item sets and

deriving association rules from them. Here's a concise summary of how the Apriori algorithm works:

1. **Support and Confidence**: The algorithm relies on two key metrics: support and confidence. Support measures how often a specific itemset appears in the dataset, while confidence measures the likelihood that an association rule is true.
2. **Frequent Itemset Generation**: The algorithm starts by identifying individual items that meet a minimum support threshold. These are considered 1-item frequent itemsets.
3. **Join and Prune**: The algorithm iteratively generates larger k-item frequent itemsets by joining smaller (k-1)-item frequent itemsets and then pruning those that don't meet the minimum support threshold.
4. **Association Rule Generation**: From the frequent itemsets, association rules are generated by considering all possible combinations of items in each itemset. The confidence of each rule is calculated.
5. **Rule Selection**: The generated association rules are filtered based on the minimum confidence threshold. Rules that satisfy the confidence criterion are considered interesting and relevant.
6. **Iteration**: The algorithm repeats the process of generating larger frequent itemsets, creating association rules, and filtering them until no more frequent itemsets or rules can be generated.