Machine Learning
(IT312)

Prepared by

Mr. Umesh B. Sangule

Assistant Professor

Department of Information Technology

Unit-III

CLASSIFICATION

Course Objectives : *To explore the different types of Classification algorithm.*

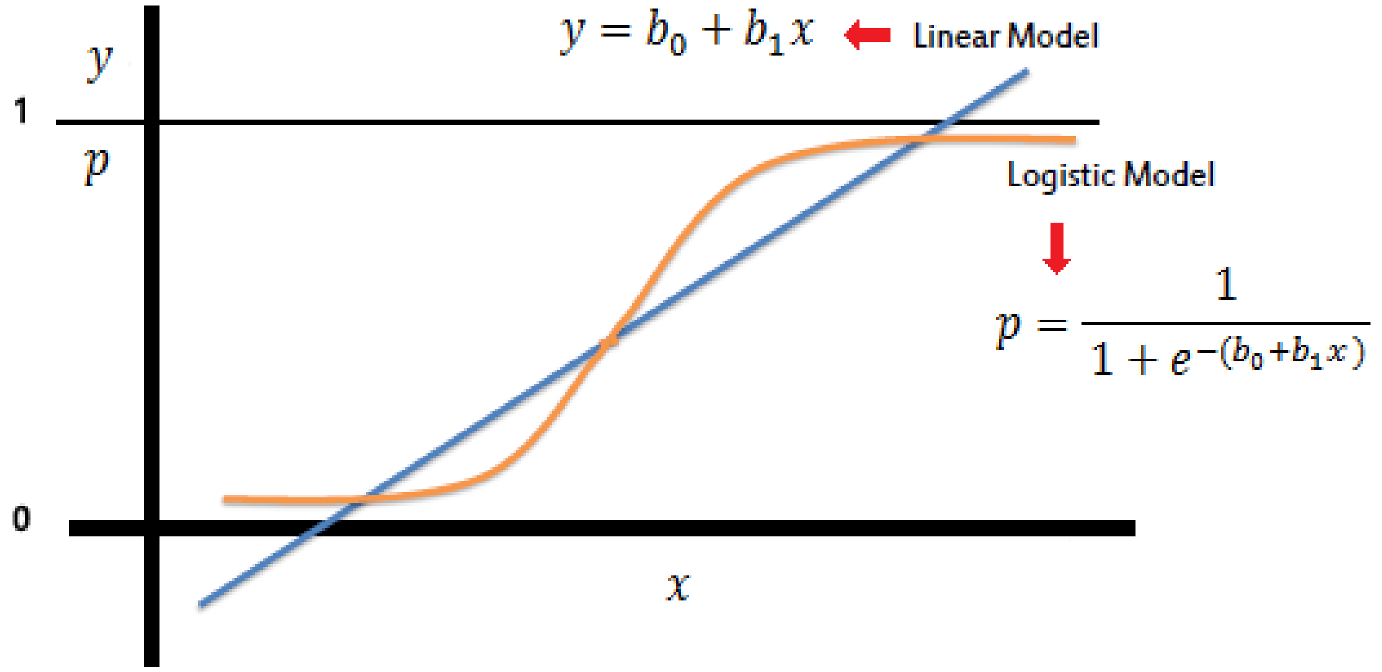
Course Outcome(CO3) : *Apply different classification algorithms for various machine learning applications,*

Classification

Sigmoid Function,

- You must be wondering how logistic regression squeezes the output of linear regression between 0 and 1.
- Formula of logistic function:

$$Y = \frac{1}{1 + e^{-(\beta_0+\beta_1 x)}}$$



Classification

Characteristics of the Sigmoid Function

- **S-Shaped Curve**: The most prominent characteristic of the sigmoid function is its S-shaped curve, which makes it suitable for modeling the probability of binary outcomes.
- **Bounded Output**: The sigmoid function always produces values between 0 and 1, which is ideal for representing probabilities.
- **Symmetry**: The sigmoid function is symmetric around its midpoint at x=0.5.
- **Differentiability**: The sigmoid function is differentiable, allowing for the calculation of gradients necessary for optimization algorithms like gradient descent.

Classification Algorithms

Decision Tree:

- A decision tree is a non-parametric supervised learning algorithm for classification and regression tasks.
- It has a hierarchical tree structure consisting of a root node, branches, internal nodes, and leaf nodes.
- The name itself suggests that it uses a flowchart like a tree structure to show the predictions that result from a series of feature-based splits.
- It starts with a root node and ends with a decision made by leaves.

Classification Algorithms

Decision Tree Terminologies:

- **Root Node**: The initial node at the beginning of a decision tree, where the entire population or dataset starts dividing based on various features or conditions.
- **Decision Nodes**: Nodes resulting from the splitting of root nodes are known as decision nodes. These nodes represent intermediate decisions or conditions within the tree.
- **Leaf Nodes**: Nodes where further splitting is not possible, often indicating the final classification or outcome. Leaf nodes are also referred to as terminal nodes.

Classification Algorithms

Decision Tree Terminologies:

- **Sub-Tree**: Similar to a subsection of a graph being called a sub-graph, a sub-section of a decision tree is referred to as a sub-tree. It represents a specific portion of the decision tree.
- **Pruning**: The process of removing or cutting down specific nodes in a decision tree to prevent overfitting and simplify the model.
- **Branch / Sub-Tree**: A subsection of the entire decision tree is referred to as a branch or sub-tree. It represents a specific path of decisions and outcomes within the tree.

Classification Algorithms

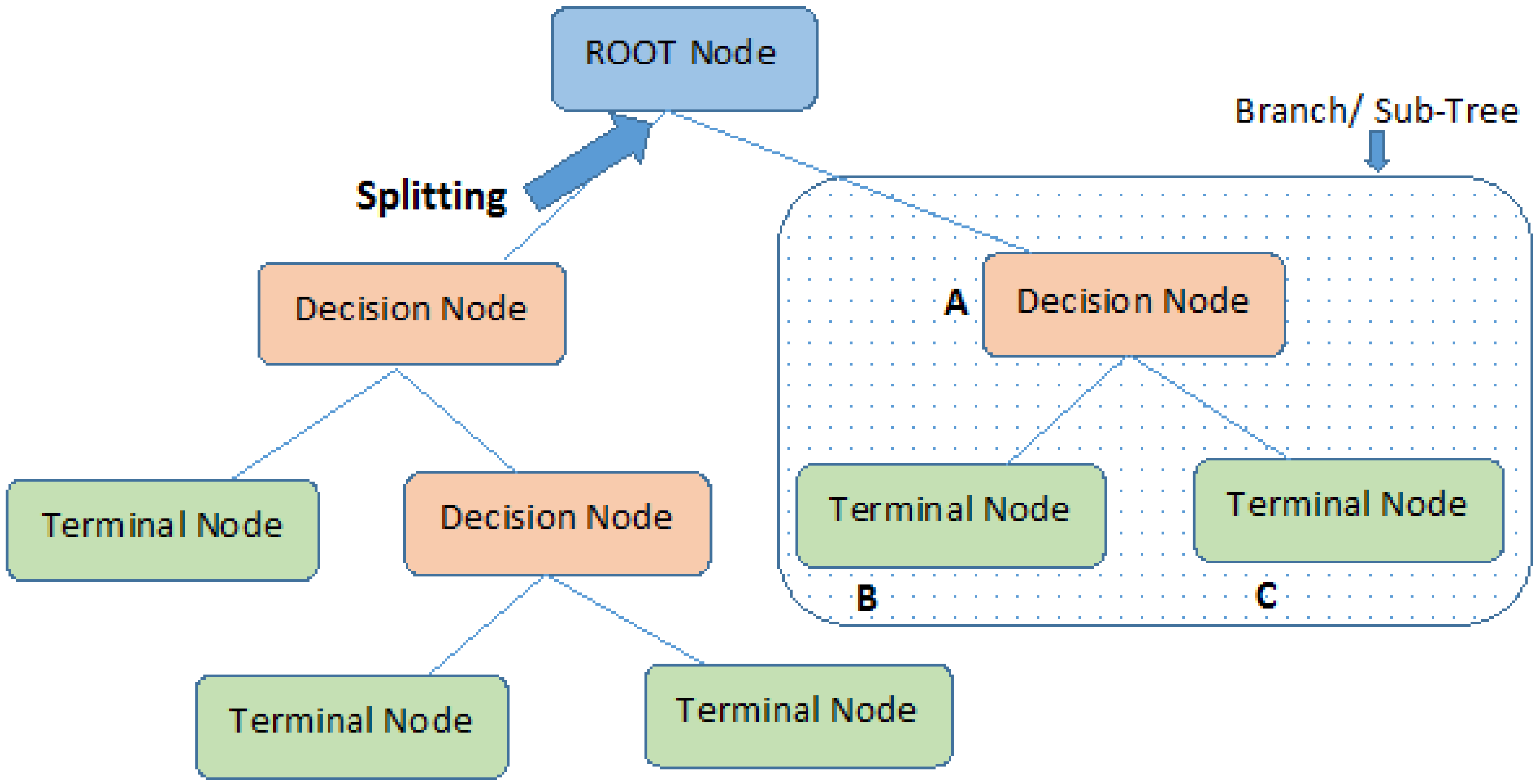
Decision Tree Terminologies:

- **Parent and Child Node**: In a decision tree, a node that is divided into sub-nodes is known as a parent node, and the sub-nodes emerging from it are referred to as child nodes.

The parent node represents a **decision or condition**, while the child nodes represent the **potential outcomes or further decisions** based on that condition.

Classification Algorithms

Decision Tree Terminologies:



Classification Algorithms

How decision tree algorithms work?

- **Starting at the Root**: The algorithm begins at the top, called the "root node," representing the entire dataset.
- **Asking the Best Questions**: It looks for the most important feature or question that splits the data into the most distinct groups. This is like asking a question at a fork in the tree.
- **Branching Out**: Based on the answer to that question, it divides the data into smaller subsets, creating new branches. Each branch represents a possible route through the tree.
- **Repeating the Process**: The algorithm continues asking questions and splitting the data at each branch until it reaches the final "leaf nodes," representing the predicted outcomes or classifications.

Classification Algorithms

Decision Tree Assumptions

- **Binary Splits:** Decision trees typically make binary splits, meaning each node divides the data into two subsets based on a single feature or condition. This assumes that each decision can be represented as a binary choice.
- **Recursive Partitioning:** Decision trees use a recursive partitioning process, where each node is divided into child nodes, and this process continues until a stopping criterion is met. This assumes that data can be effectively subdivided into smaller, more manageable subsets.
- **Feature Independence:** Decision trees often assume that the features used for splitting nodes are independent. In practice, feature independence may not hold, but decision trees can still perform well if features are correlated.

Classification Algorithms

Decision Tree Assumptions

- **Top-Down Greedy Approach:** Decision trees are constructed using a top-down, greedy approach, where each split is chosen to maximize information gain or minimize impurity at the current node. This may not always result in the globally optimal tree.
- **Categorical and Numerical Features:** Decision trees can handle both categorical and numerical features. However, they may require different splitting strategies for each type.
- **Impurity Measures:** Decision trees use impurity measures such as Gini impurity or entropy to evaluate how well a split separates classes. The choice of impurity measure can impact tree construction.

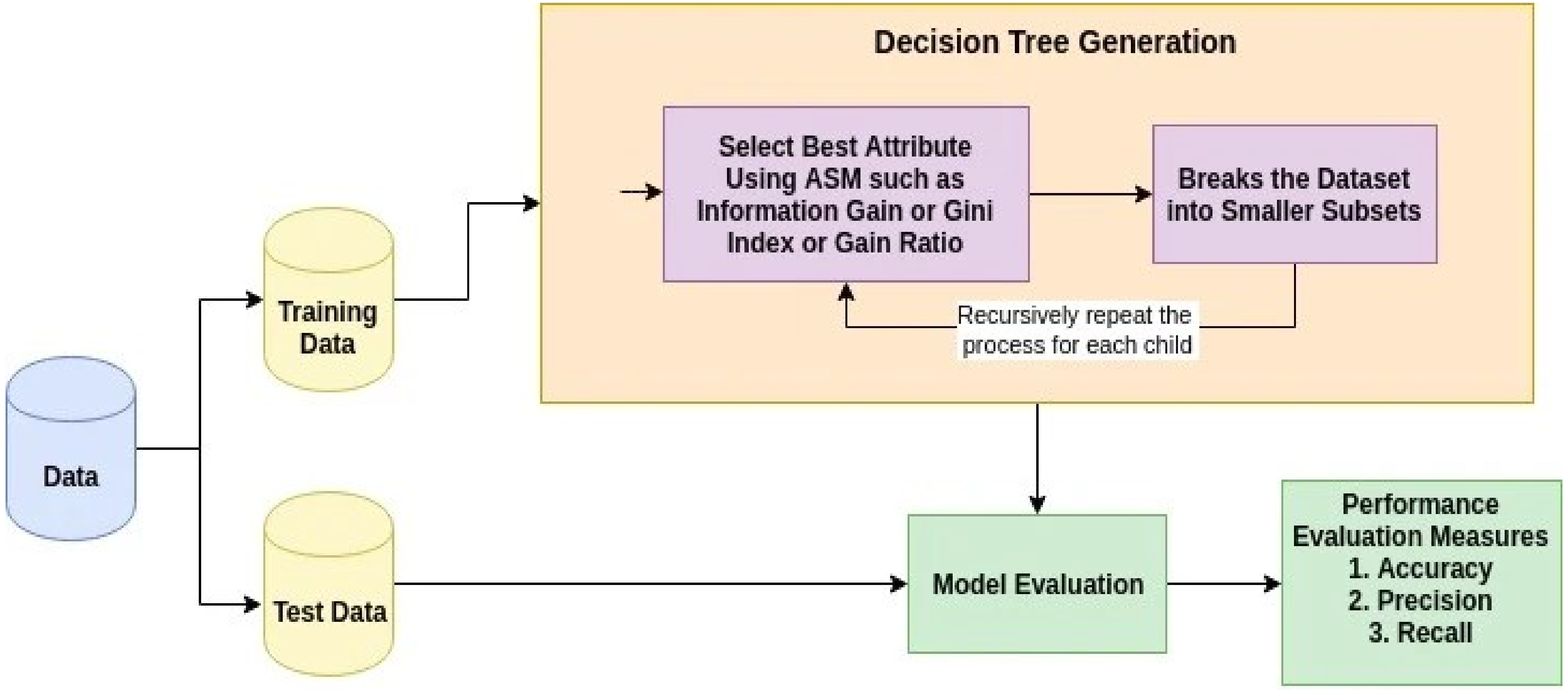
Classification Algorithms

Decision Tree Assumptions

- **No Missing Values:** Decision trees assume that there are no missing values in the dataset or that missing values have been appropriately handled through imputation or other methods.
- **Equal Importance of Features:** Decision trees may assume equal importance for all features unless feature scaling or weighting is applied to emphasize certain features.
- **No Outliers:** Decision trees are sensitive to outliers, and extreme values can influence their construction. Preprocessing or robust methods may be needed to handle outliers effectively.

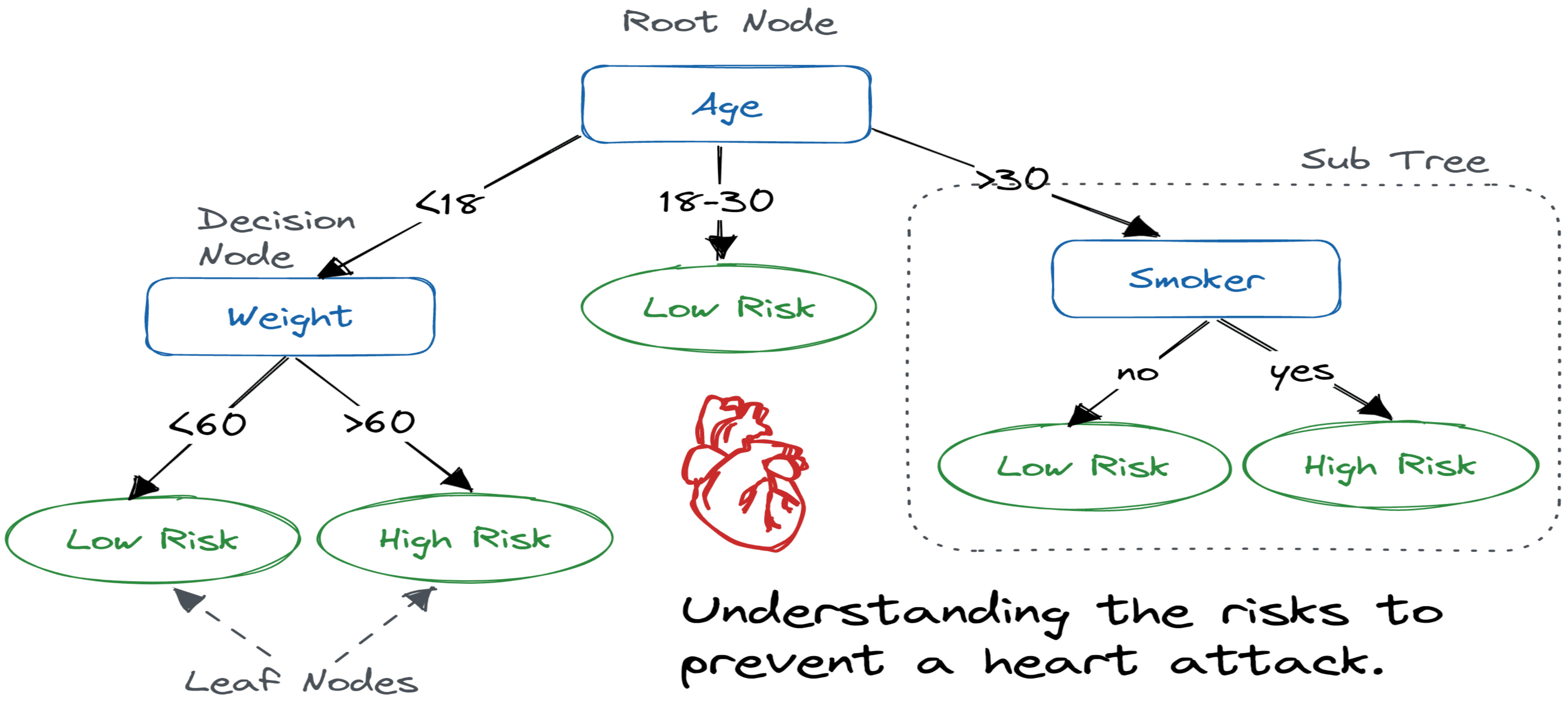
Classification Algorithms

Decision Tree Algorithm:


Decision Tree Generation
Select Best Attribute Using ASM such as Information Gain or Gini Index or Gain Ratio
Breaks the Dataset into Smaller Subsets
Recursively repeat the process for each child
Training Data
Data
Test Data
Model Evaluation
Performance Evaluation Measures
1. Accuracy
2. Precision
3. Recall

Classification Algorithms

Decision Tree Algorithm:



Understanding the risks to prevent a heart attack.

Classification Algorithms

Attribute Selection Measures(ASM):

Attribute selection measure is a heuristic for selecting the splitting criterion that partitions data in the best possible manner

➢ ***Entropy and Information Gain:***

1) Entropy is nothing but the uncertainty in our dataset or measure of disorder.
2) Entropy is a measure of the randomness in the information being processed.
3) The higher the entropy, the harder it is to draw any conclusions from that information.

Attribute Selection Measures(ASM):

Entropy and Information Gain:

- Thus, a node with more variable composition, such as 2Pass and 2 Fail would be considered to have higher Entropy than a node which has only pass or only fail.
- The maximum level of entropy or disorder is given by 1 and minimum entropy is given by a value 0.
- Leaf nodes which have all instances belonging to 1 class would have an entropy of 0.
- Whereas, the entropy for a node where the classes are divided equally would be 1.

Attribute Selection Measures(ASM):

Entropy and Information Gain:

Entropy is measured by the formula:

$$E = -\sum_{i=1}^{n} p_i \, log_2(p_i)$$

Where the pi is the probability of randomly selecting an example in class i.

Attribute Selection Measures(ASM):

Entropy and Information Gain:

- Now essentially what a Decision Tree does to determine the root node is to calculate the entropy for each variable and its potential splits.
- For this we have to calculate a potential split from each variable, calculate the average entropy across both or all the nodes and then the change in entropy vis a vis the parent node.
- This change in entropy is termed **Information Gain** and represents how much information a feature provides for the target variable.

$$Information\ Gain = Entropy_{parent} - Entropy_{children}$$

Attribute Selection Measures(ASM):

Entropy and Information Gain:

1. Probability of pass and fail at each node, i.e, the Pi:

$$P_i = \frac{\#Pass\ or\ Fail}{\#\ subnode}$$

2. Entropy:

$$E = -(P_{pass}\, log_2(P_{pass}) + P_{fail}\, log_2(P_{fail}))$$

3. Average Entropy at child nodes:

$$\text{Average Entropy} = \frac{(n_{subnode\ 1})}{n\_parent} E\_subnode1 + \frac{(n_{subnode\ 2})}{n_{parent}} E\_subnode2$$

Attribute Selection Measures(ASM):

Gini Index:

- The other way of splitting a decision tree is via the Gini Index. The Entropy and Information Gain method focuses on purity and impurity in a node.
- The Gini Index or Impurity measures the probability for a random instance being misclassified when chosen randomly.
- The formula for Gini Index

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

Where **c** represents the no. of classes in the target variable: Pass and Fail

P(i) represents the ratio of Pass/Total no. of observations in node.

Classification Algorithms

Decision Tree:

- ➢ Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes.
- ➢ The algorithm selection is also based on the type of target variables. Let us look at some algorithms used in Decision Trees:

ID3 → (extension of D3)

CART → (Classification And Regression Tree)

CHAID → (Chi-square automatic interaction detection Performs multi-level splits when computing classification trees)

Classification Algorithms

Decision Tree:

When to Stop Splitting?

- We can set the maximum depth of our decision tree using the **max_depth** parameter. The more the value of max_depth, the more complex your tree will
- Another way is to set the minimum number of samples for each spilt. It is denoted by **min_samples_split.** Here we specify the minimum number of samples required to do a spilt.

Classification Algorithms(Decision Tree:)

Pruning:

- Pruning is another method that can help us avoid overfitting. It helps in improving the performance of the Decision tree by cutting the nodes or sub-nodes which are not significant. Additionally, it removes the branches which have very low importance.
- There are mainly 2 ways for pruning:
- **Pre-pruning** – we can stop growing the tree earlier, which means we can prune/remove/cut a node if it has low importance while growing the tree.
- **Post-pruning** – once our tree is built to its depth, we can start pruning the nodes based on their significance.

Classification Algorithms(Decision Tree:)

```
# Load libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']
# load dataset
pima = pd.read_csv("diabetes.csv", names=col_names)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

Classification Algorithms(Decision Tree:)

# Create Decision Tree classifier object

```
clf = DecisionTreeClassifier()
clf = clf.fit(X_train,y_train)
```

#Predict the response for test dataset

```
y_pred = clf.predict(X_test)
```

# Model Accuracy, how often is the classifier correct?

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Classification Algorithms(Random Forest)

Random Forest:

- Random Forest Algorithm widespread popularity stems from its user-friendly nature and adaptability, enabling it to tackle both classification and regression problems effectively.
- The algorithm’s strength lies in its ability to handle complex datasets and mitigate overfitting, making it a valuable tool for various predictive tasks in machine learning.
- One of the most important features of the Random Forest Algorithm is that it can handle the data set containing **continuous variables**, as in the case of regression, and **categorical variables**, as in the case of classification

Classification Algorithms(Random Forest)

Working of Random Forest:

- Before understanding the working of the random forest algorithm in machine learning, we must look into the **Ensemble learning technique**.
- **Ensemble** simply means **combining multiple models**. Thus a collection of models is used to make predictions rather than an individual model.
- Ensemble uses two types of methods:

  1. **Bagging**
  2. **Boosting**

Classification Algorithms(Random Forest)

Bagging:

- ➢ It creates a different training subset from sample training data with replacement & the final output is based on majority voting.

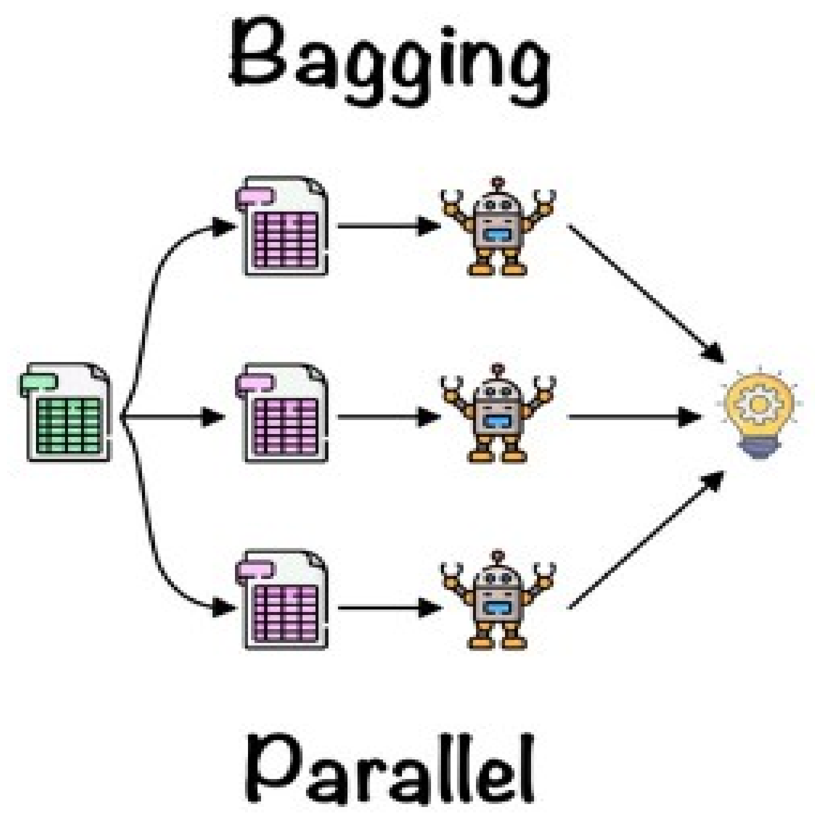
For example, Random Forest.

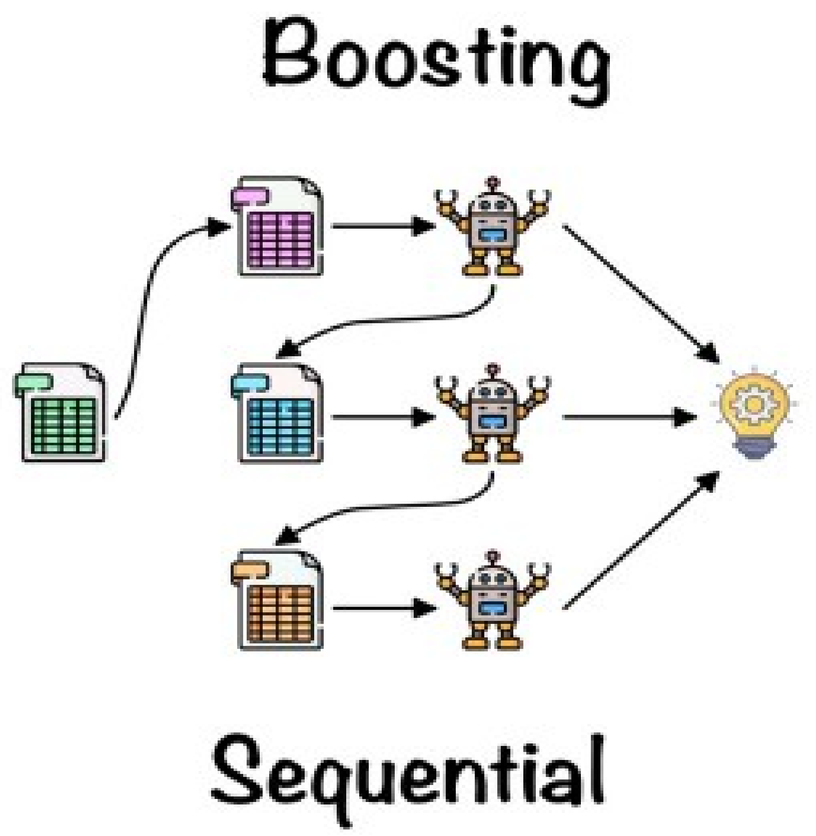
Boosting

- ➢ It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy.

For example, ADA BOOST, XG BOOST.

Classification Algorithms(Random Forest)

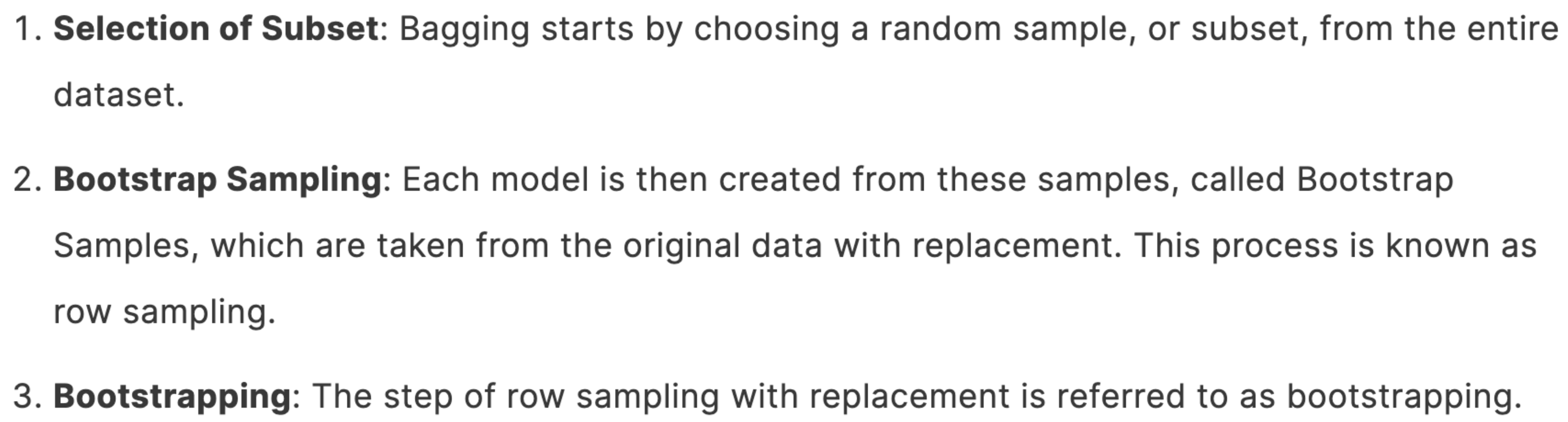




Classification Algorithms(Random Forest)

Bagging:

- Bagging, also known as ***Bootstrap Aggregation***, serves as the ensemble technique in the Random Forest algorithm. Here are the steps involved in Bagging:



1. **Selection of Subset**: Bagging starts by choosing a random sample, or subset, from the entire dataset.
2. **Bootstrap Sampling**: Each model is then created from these samples, called Bootstrap Samples, which are taken from the original data with replacement. This process is known as row sampling.
3. **Bootstrapping**: The step of row sampling with replacement is referred to as bootstrapping.

Classification Algorithms(Random Forest)

Bagging:

Here are the steps involved in Bagging:

4. **Independent Model Training**: Each model is trained independently on its corresponding Bootstrap Sample. This training process generates results for each model.

5. **Majority Voting**: The final output is determined by combining the results of all models through majority voting. The most commonly predicted outcome among the models is selected.

6. **Aggregation**: This step, which involves combining all the results and generating the final output based on majority voting, is known as aggregation.

Classification Algorithms(Random Forest)

Boosting:

- Boosting is one of the techniques that use the concept of ensemble learning. A boosting algorithm combines multiple simple models (also known as weak learners or base estimators) to generate the final output.
- It is done by building a model by using weak models in series.
- There are several boosting algorithms; **AdaBoost** was the first really successful boosting algorithm that was developed for the purpose of binary classification.
- AdaBoost is an abbreviation for Adaptive Boosting and is a prevalent boosting technique that combines multiple "weak classifiers" into a single "strong classifier."

Classification Algorithms(Random Forest)

Steps Involved in Random Forest Algorithm:

- **Step 1:** In the Random forest model, a subset of data points and a subset of features is selected for constructing each decision tree. Simply put, n random records and m features are taken from the data set having k number of records.
- **Step 2:** Individual decision trees are constructed for each sample.
- **Step 3:** Each decision tree will generate an output.
- **Step 4:** Final output is considered based on ***Majority Voting or Averaging*** for Classification and regression, respectively.

Classification Algorithms(Random Forest)

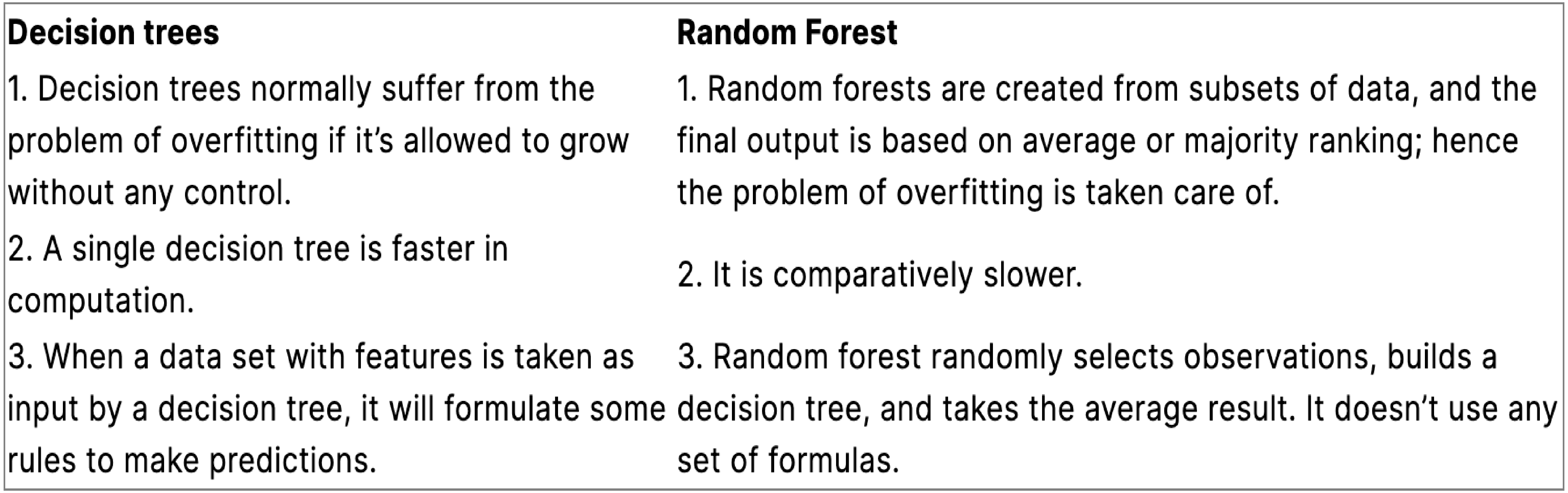
Important Features of Random Forest

- **Diversity:** Not all attributes/variables/features are considered while making an individual tree; each tree is different.
- **Immune to the curse of dimensionality:** Since each tree does not consider all the features, the feature space is reduced.
- **Parallelization:** Each tree is created independently out of different data and attributes. This means we can fully use the CPU to build random forests.
- **Train-Test split:** In a random forest, we don't have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.
- **Stability:** Stability arises because the result is based on majority voting/ averaging.

Classification Algorithms(Random Forest)

Difference Between Decision Tree and Random Forest

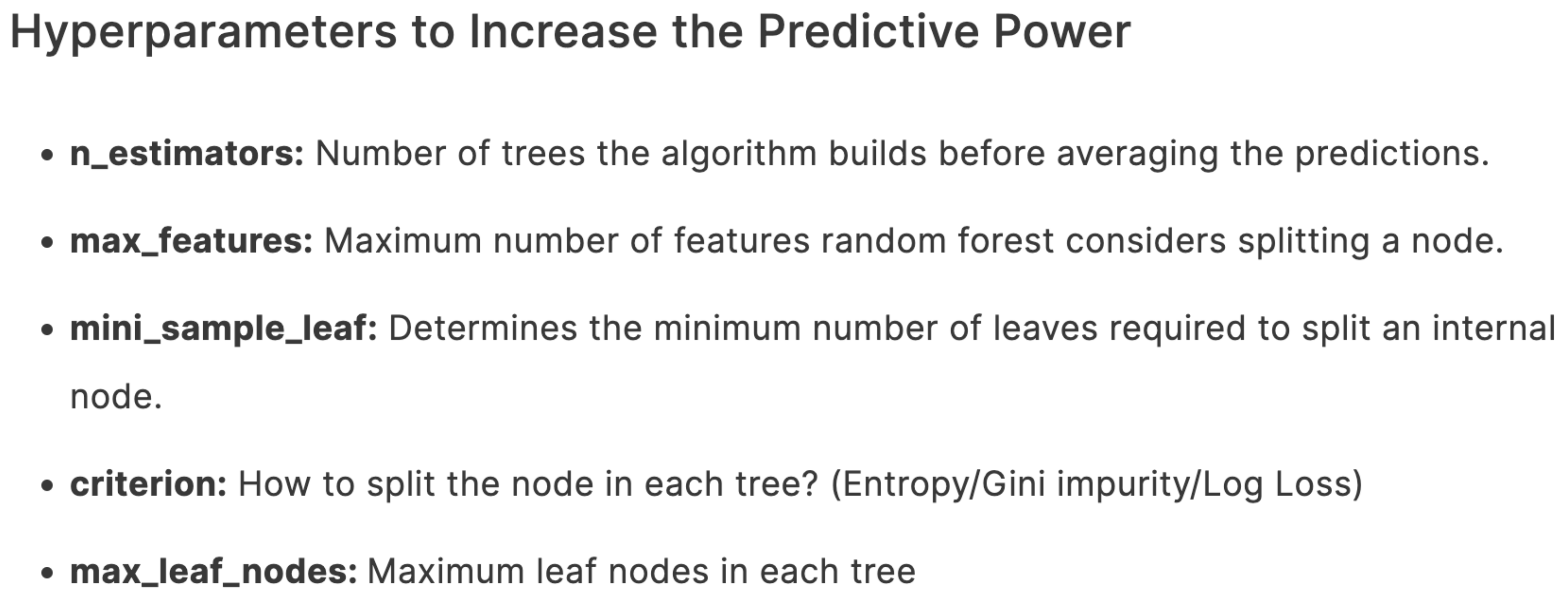
Random forest is a collection of decision trees; still, there are a lot of differences in their behavior.



Decision trees	Random Forest
1. Decision trees normally suffer from the problem of overfitting if it's allowed to grow without any control.	1. Random forests are created from subsets of data, and the final output is based on average or majority ranking; hence the problem of overfitting is taken care of.
2. A single decision tree is faster in computation.	2. It is comparatively slower.
3. When a data set with features is taken as input by a decision tree, it will formulate some rules to make predictions.	3. Random forest randomly selects observations, builds a decision tree, and takes the average result. It doesn't use any set of formulas.

Classification Algorithms(Random Forest)

Important Hyperparameters in Random Forest

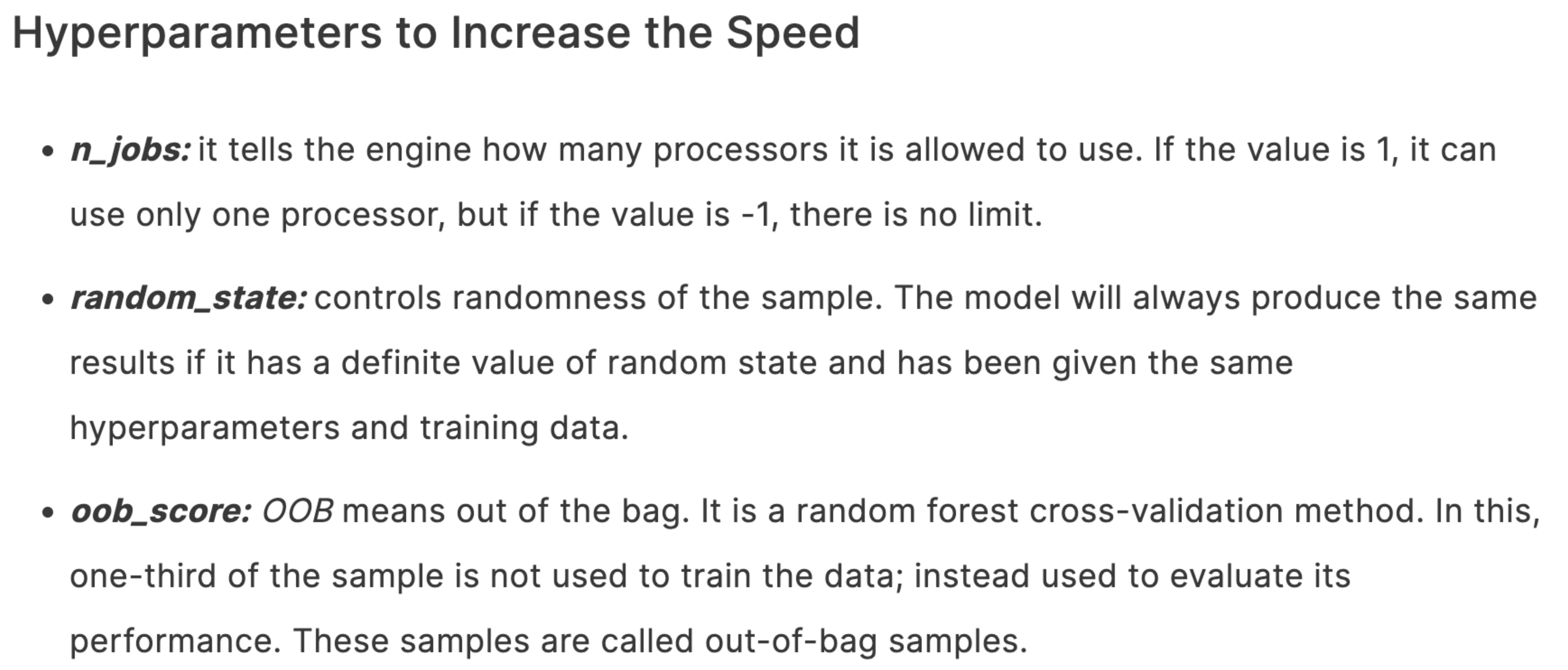


Hyperparameters to Increase the Predictive Power

- **n_estimators:** Number of trees the algorithm builds before averaging the predictions.
- **max_features:** Maximum number of features random forest considers splitting a node.
- **mini_sample_leaf:** Determines the minimum number of leaves required to split an internal node.
- **criterion:** How to split the node in each tree? (Entropy/Gini impurity/Log Loss)
- **max_leaf_nodes:** Maximum leaf nodes in each tree

Classification Algorithms(Random Forest)

Important Hyperparameters in Random Forest



Hyperparameters to Increase the Speed

- ***n_jobs:*** it tells the engine how many processors it is allowed to use. If the value is 1, it can use only one processor, but if the value is -1, there is no limit.
- ***random_state:*** controls randomness of the sample. The model will always produce the same results if it has a definite value of random state and has been given the same hyperparameters and training data.
- ***oob_score:*** *OOB* means out of the bag. It is a random forest cross-validation method. In this, one-third of the sample is not used to train the data; instead used to evaluate its performance. These samples are called out-of-bag samples.

Classification Algorithms(Random Forest)

Random Forest using Python:

```
# Data Processing

import pandas as pd

import numpy as np


# Modelling

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, confusion_matrix, precision_score,
recall_score, ConfusionMatrixDisplay

from sklearn.model_selection import train_test_split
```

Classification Algorithms(Random Forest)

Random Forest using Python:

# Split the data into training and test sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Classification Algorithms

Adaboost(Adaptive Boosting) Algorithm:

- Adaboost (short for Adaptive Boosting) is a popular boosting algorithm used in machine learning to improve the accuracy of **binary classification models**.
- **Boosting** is a technique that combines multiple weak learning algorithms to create a strong learner that can make accurate predictions,
- Adaboost works by **sequentially** training a series of weak classifiers on weighted versions of the training data. The weights are adjusted after each iteration to give more importance to misclassified data points.
- The final model **combines** the **weak classifiers** using a weighted sum to make predictions on new data.

Classification Algorithms

Adaboost(Adaptive Boosting) Algorithm:

- It focuses on misclassified examples and **adjusts** the weights of the training data to **prioritize** these examples.
- By doing so, the algorithm can learn from the mistakes made by the weak classifiers and produce a strong model that can generalize well on new data.
- Each weak classifier is trained to classify the examples as either positive or negative. After each iteration, the weights of the misclassified examples are increased to give them more importance in the next iteration.
- This process continues until a predetermined number of weak classifiers are trained or a specified threshold is reached.

Classification Algorithms

Adaboost(Adaptive Boosting) Algorithm:

- The key idea is to choose weak classifiers that perform slightly better than random guessing.
- The biggest advantage of this algorithm is its ability to handle **noisy data** and **outliers**.
- It is also **less prone to overfitting** than other algorithms such as support vector machines.
- Additionally, the algorithm is easy to interpret and can be used to generate insights about the data.

Classification Algorithms

Adaboost Algorithm Step-by-Step Process:

1. **Initialize the weights of the training examples:** The first step is to assign equal weights to all the training examples. These weights are used to emphasize the misclassified examples in the subsequent iterations.

2. **Train a weak classifier:** A weak classifier is trained on the weighted training data in each iteration. The weak classifier aims to classify the examples as positive or negative. A variety of weak classifiers can be used, such as decision trees, linear models, or support vector machines.

3. **Evaluate the performance of the weak classifier:** Once the weak classifier is trained, its performance is evaluated on the training data. The misclassified examples are given higher weights to prioritize them in the next iteration.

Classification Algorithms

Adaboost Algorithm Step-by-Step Process:

4. **Update the weights of the training examples:** The weights of the training examples are updated based on their classification by the weak classifier. The weights of the misclassified examples are increased, while the weights of the correctly classified examples are decreased. This ensures that the weak classifier focuses on the challenging examples.

5. **Repeat steps 2-4 for a predetermined number of iterations:** The previous steps are repeated for a specified number of iterations, or until a threshold is reached. The weights of the training examples are adjusted in each iteration to prioritize the misclassified examples and learn from the mistakes made by the weak classifier.

6. **Combine the weak classifiers into a strong model:** After the specified number of iterations, the weak classifiers are combined into a strong model using a weighted sum of their outputs. The final model can make accurate predictions on new, unseen data.

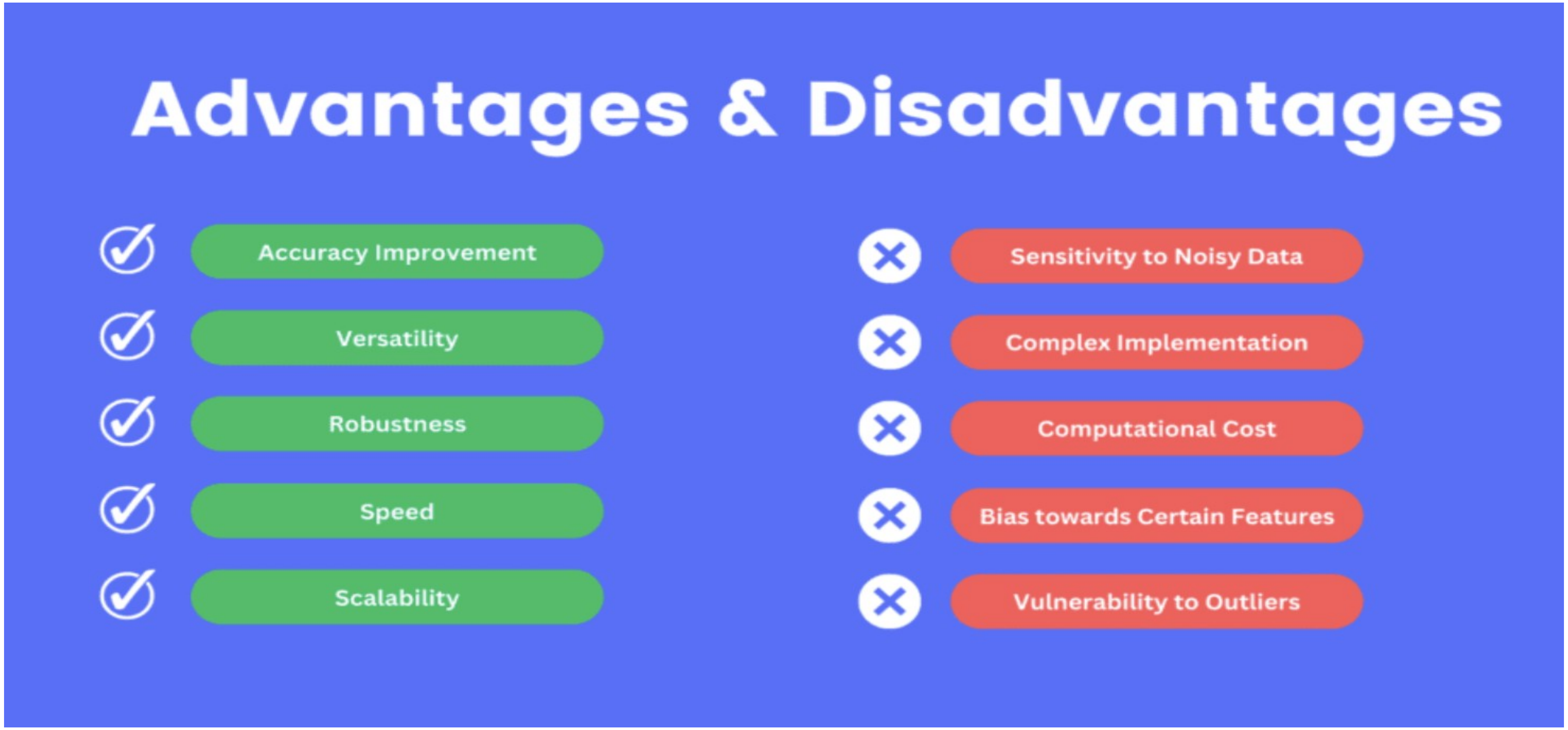
Classification Algorithms

Adaboost Algorithm Applications:

Adaboost Applications
4
Image Classification
Text Classification
Medical Diagnosis

Classification Algorithms

Advantages and Disadvantages of the Adaboost Algorithm:



Advantages & Disadvantages

✓ Accuracy Improvement	✗ Sensitivity to Noisy Data
✓ Versatility	✗ Complex Implementation
✓ Robustness	✗ Computational Cost
✓ Speed	✗ Bias towards Certain Features
✓ Scalability	✗ Vulnerability to Outliers

Classification Algorithms

Gradient Boosting Algorithm:

- The key idea behind Gradient Boosting is to fit a sequence of weak learners (e.g., decision trees) **to the residuals of the previous learners.**
- The algorithm starts by training a weak learner using the original data. Then compute the residuals (that is, the difference between the predicted and actual values) and fit another weak learner to the residuals.
- This process is repeated for a predefined number of iterations, with each new learner trying to minimize the leftovers of the previous learner. The final prediction is the sum of all weak learner predictions.
- Gradient boosting approach can use for both the regression and classification problems.

Classification Algorithms

How Gradient Boosting Works

➢ Three components are involved in gradient boosting:

1. An optimized loss function.

2. A lousy indicator learner.

3. An additive model that reduces failure cases.

1. *Loss Function*:

➢ The benefit of the gradient booster framework is that for each loss function, you may decide to use, the new booster algorithm does not need to extract.

➢Instead, the framework is general enough to enable any differentiable loss function.

Classification Algorithms

How Gradient Boosting Works

2. *Weak Learner*:

- Decision trees are used in gradient boosting as weak learners.

3. *Additive Model*:

- Trees are introduced one at a time. The current trees in the model are not updated. A gradient descent technique minimizes losses when adding trees.
- Traditionally, gradient descent minimizes the number of parameters, such as the regression equation coefficients,
- After measuring the error or loss, the values are updated to minimize the error.

Classification Algorithms

Naive Bayes Classifier:

- Naive Bayes classifier is a straightforward and powerful algorithm for the classification task. Even if we are working on a data set with millions of records with some attributes, it is suggested to try Naive Bayes approach.
- Naive Bayes classifier gives great results when we use it for **textual data analysis**. Such as Natural Language Processing(NLP).
- Naive Bayes is a kind of classifier which uses the **Bayes Theorem**.
- Bayes Theorem works on **conditional probability**. Conditional probability is the probability that **something will happen**, given that something else has **already occurred**. Using the conditional probability, we can calculate the probability of an event using its prior knowledge.

Classification Algorithms

Naive Bayes Classifier:

- It predicts membership probabilities for each class such as the probability that given record or data point belongs to a particular class.
- The class with the **highest probability** is considered as the most likely class.
- Naive Bayes classifier assumes that all the features are unrelated to each other. Presence or absence of a feature does not influence the presence or absence of any other feature.
- This approach is based on the assumption that the features of the input data are conditionally independent given the class, allowing the algorithm to make predictions quickly and accurately.

Classification Algorithms

Types of Naive Bayes Algorithm

Gaussian Naive Bayes

When attribute values are continuous, an assumption is made that the values associated with each class are distributed according to Gaussian i.e., Normal Distribution.

MultiNomial Naive Bayes

MultiNomial Naive Bayes is preferred to use on data that is multinomially distributed. It is one of the standard classic algorithms. Which is used in text categorization (classification). Each event in text classification represents the occurrence of a word in a document.

Bernoulli Naive Bayes

Bernoulli Naive Bayes is used on the data that is distributed according to multivariate Bernoulli distributions.i.e., multiple features can be there, but each one is assumed to be a binary-valued (Bernoulli, boolean) variable. So, it requires features to be binary valued.

Classification Algorithms

Advantages and Disadvantage of Naive Bayes classifier

Advantages

- Naive Bayes Algorithm is a fast, highly scalable algorithm.
- Naive Bayes can be use for Binary and Multiclass classification. It provides different types of Naive Bayes Algorithms like GaussianNB, MultinomialNB, BernoulliNB.
- It is a simple algorithm that depends on doing a bunch of counts.
- Great choice for Text Classification problems. It's a popular choice for spam email classification.
- It can be easily train on small dataset

Disadvantages

- It considers all the features to be unrelated, so it cannot learn the relationship between features.

Classification Algorithms

Naive Bayes classifier using Python:

```
# Import libraries
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.30)
from sklearn.naive_bayes import GaussianNB
# Build a Gaussian Classifier
model = GaussianNB()
# Model training
model.fit(X_train, y_train)
```

Classification Algorithms

Naive Bayes classifier using Python:

```
# Predict Output
predicted = model.predict([X_test[6]])
print("Actual Value:", y_test[6])
print("Predicted Value:", predicted[0])
# Evaluation Metrics
from sklearn.metrics import (accuracy_score,confusion_matrix, f1_score,)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_pred, y_test)
F1 = f1_score(y_pred, y_test, average="weighted")
```

Classification Algorithms(SVM)

Support Vector Machine(SVM)

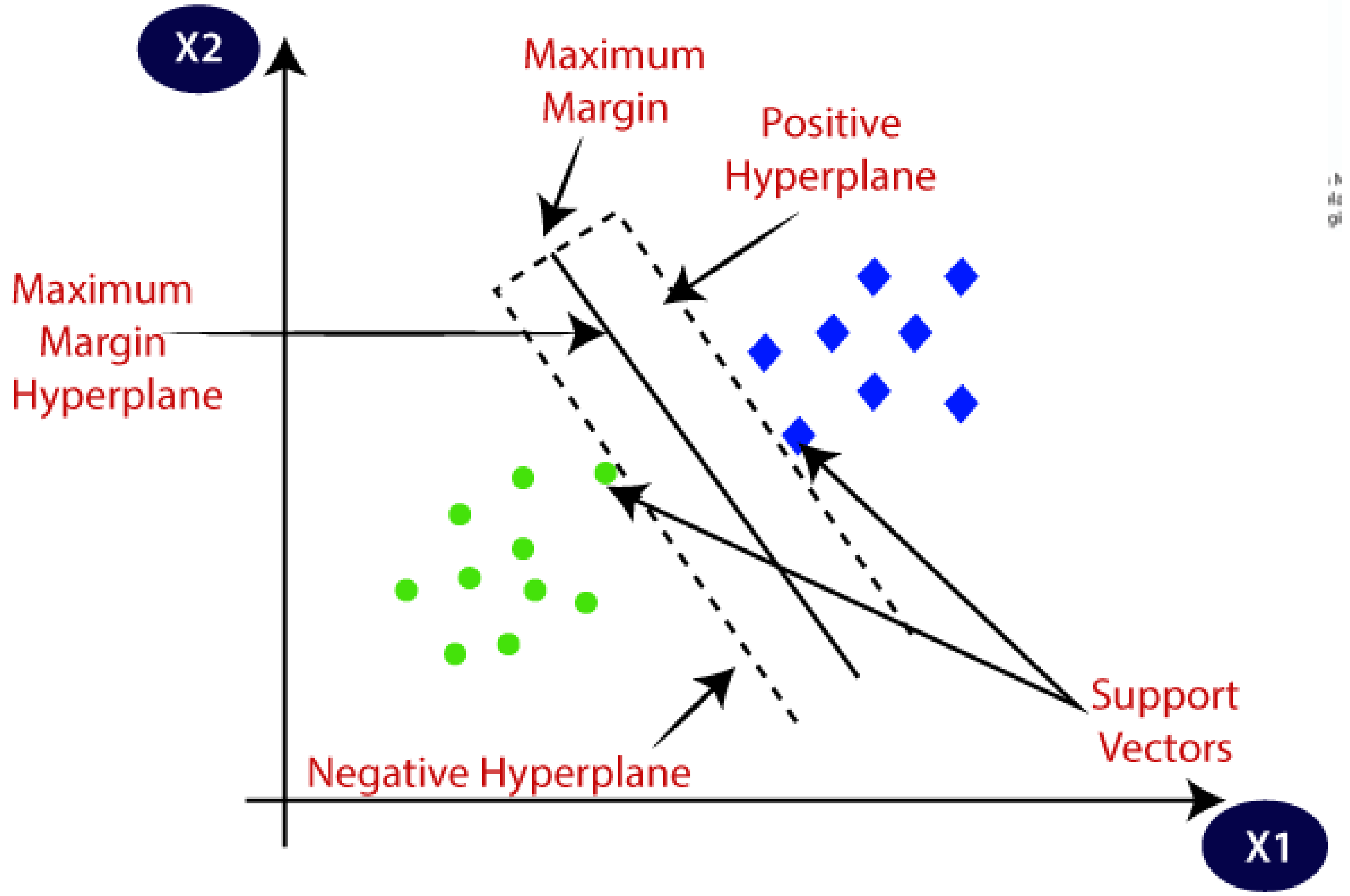
- It is a supervised machine learning problem where we try to find a **hyperplane** that **best separates the two classes**.
- SVM selects best hyperplane by finding the maximum margin between the hyperplanes that means maximum distances between the two classes.

Important Terms

- **Support Vectors:** These are the points that are closest to the hyperplane. A separating line will be defined with the help of these data points.
- **Margin**: it is the distance between the hyperplane and the observations closest to the hyperplane (support vectors). In SVM large margin is considered a good margin. There are two types of margins hard margin and soft margin.

Classification Algorithms(SVM)

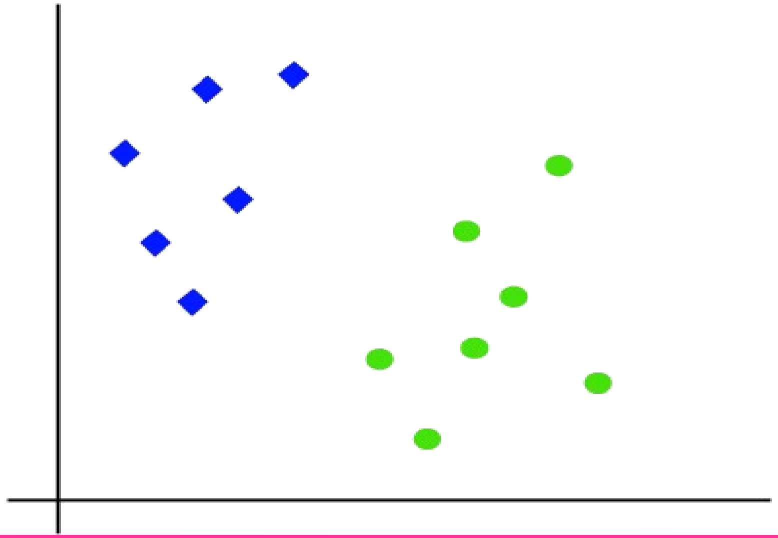
Support Vector Machine(SVM)



Classification Algorithms(SVM)

How Does SVM Work?

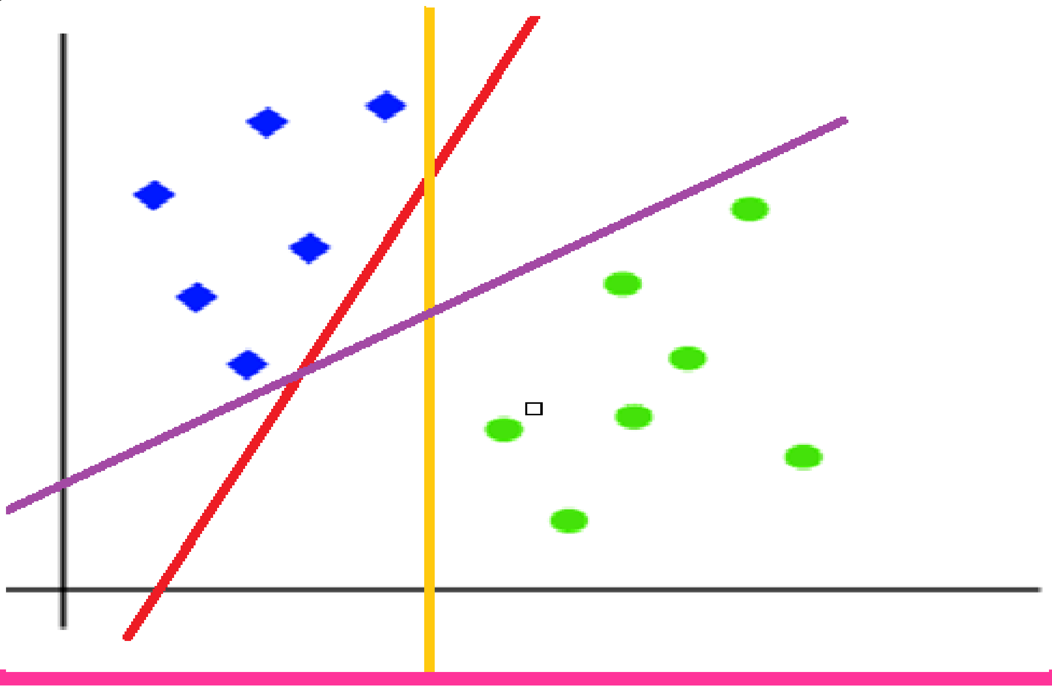
- SVM is defined such that it is defined in terms of the support vectors only, we don’t have to worry about other observations since the margin is made using the points which are closest to the hyperplane (support vectors),
- Suppose we have a dataset that has two classes (green and blue). We want to classify that the new data point as either blue or green.



Classification Algorithms(SVM)

How Does SVM Work?

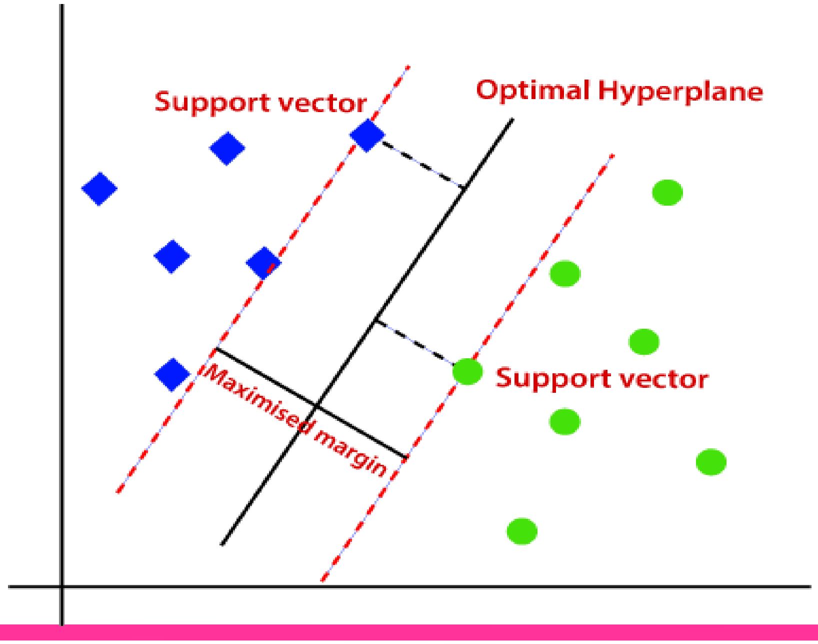
- To classify these points, we can have many decision boundaries,
- Since we are plotting the data points in a 2-dimensional graph we call this **decision boundary** a **straight line** but if we have more dimensions, we call this decision boundary a "**hyperplane**"



Classification Algorithms(SVM)

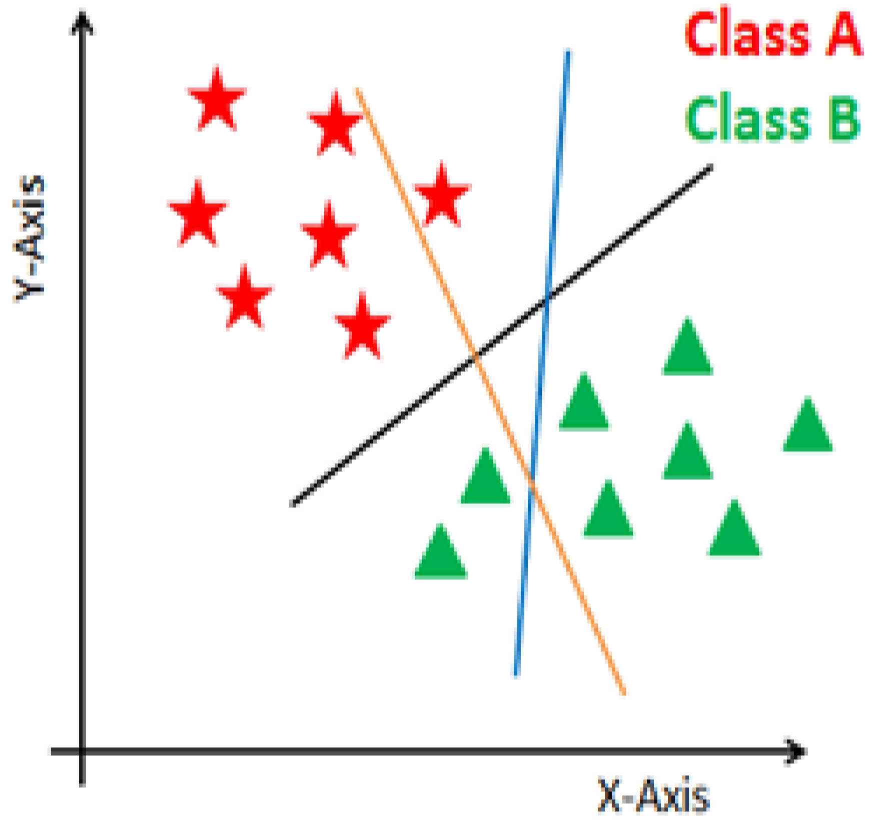
How Does SVM Work?

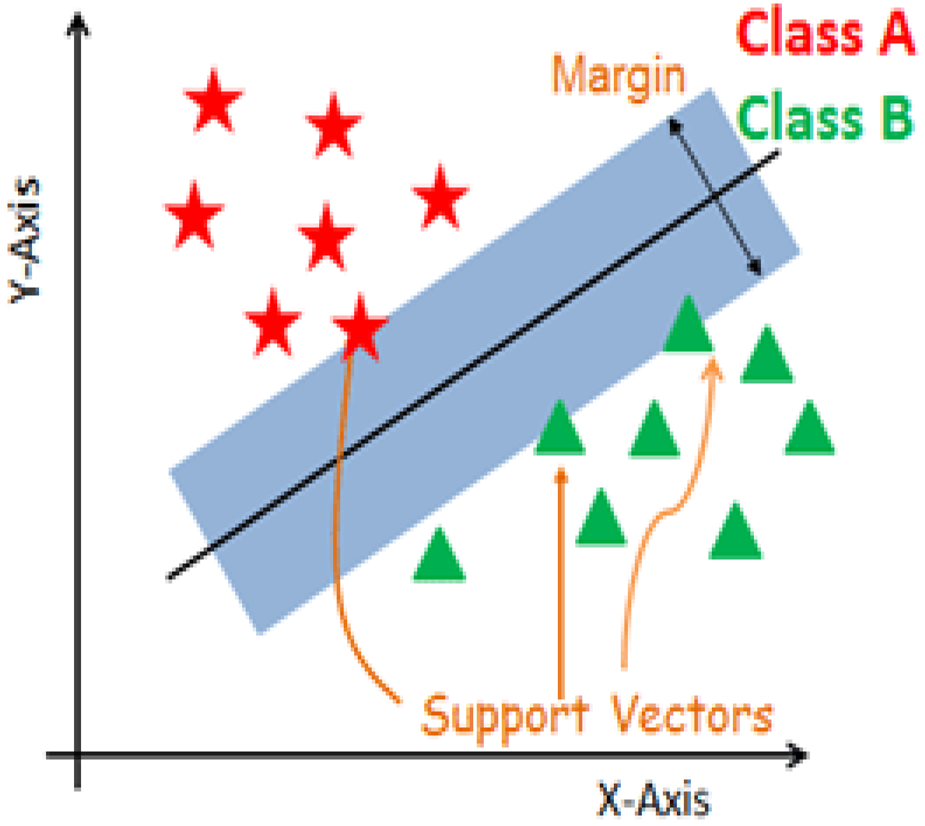
- The best hyperplane is that plane that has the maximum distance from both the classes, and this is the main aim of SVM.
- This is done by finding different hyperplanes which classify the labels in the best way then it will choose the one which is farthest from the data points or the one which has a maximum margin.



Classification Algorithms(SVM)

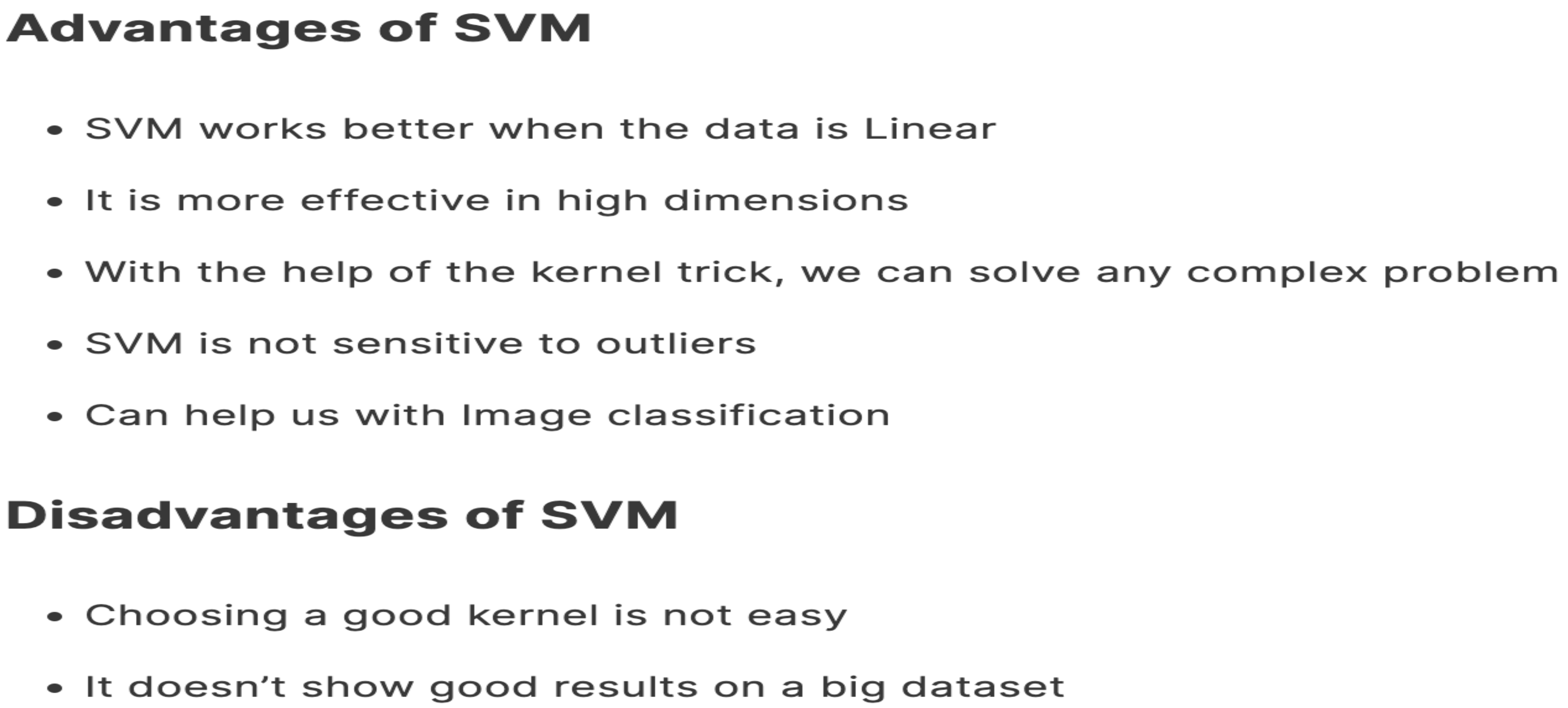
How Does SVM Work?





Classification Algorithms(SVM)

Advantages and Disadvantages of SVM:



Advantages of SVM

- SVM works better when the data is Linear
- It is more effective in high dimensions
- With the help of the kernel trick, we can solve any complex problem
- SVM is not sensitive to outliers
- Can help us with Image classification

Disadvantages of SVM

- Choosing a good kernel is not easy
- It doesn't show good results on a big dataset

Classification Algorithms Evaluation Metrics

Evaluation Metrics:

There are many ways for measuring classification performance. Accuracy, confusion matrix, log-loss, and AUC-ROC are some of the most popular metrics. Precision-recall is a widely used metrics for classification problems.

Accuracy:

- Accuracy simply measures how often the classifier correctly predicts. We can define accuracy as the ratio of the number of correct predictions and the total number of predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Classification Algorithms Evaluation Metrics

Confusion Matrix:

- Confusion Matrix is a performance measurement for the machine learning classification problems where the output can be two or more classes. It is a table with combinations of predicted and actual values.
- A confusion matrix is defined as ***the table that is often used to describe the performance of a classification model on a set of the test data for which the true values are known.***
- A confusion matrix helps us gain insight into how correct our predictions were and how they hold up against the actual values.
- The confusion matrix provides more insight into not only the performance of a predictive model, but also which classes are being predicted correctly, which incorrectly, and what type of errors are being made.

Classification Algorithms Evaluation Metrics

Confusion Matrix:

True/False Positives and Negatives:

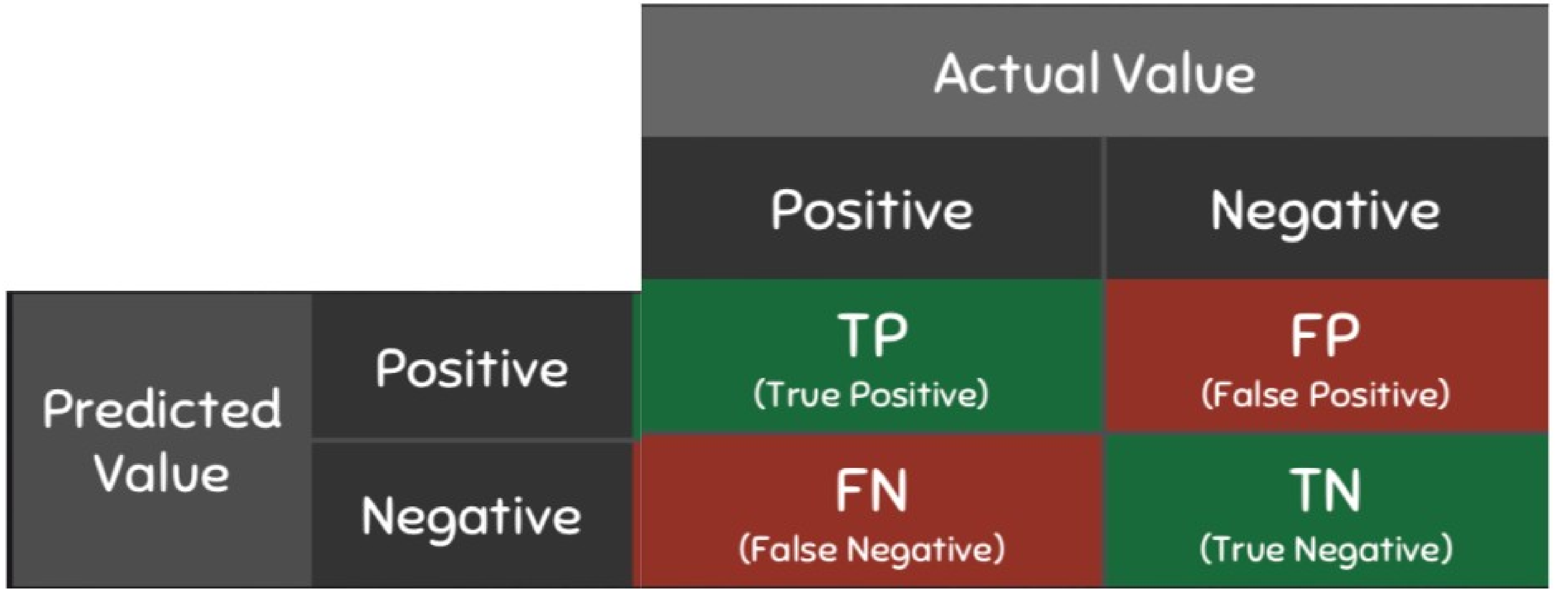
- A binary classifier can be viewed as classifying instances as positive or negative:
- **Positive**: The instance is classified as a member of the class the classifier is trying to identify.

  For example, a classifier looking for cat photos would classify photos with cats as positive (when correct).
- **Negative**: The instance is classified as not being a member of the class we are trying to identify.

  For example, a classifier looking for cat photos should classify photos with dogs (and no cats) as negative.

Classification Algorithms Evaluation Metrics

Confusion Matrix:

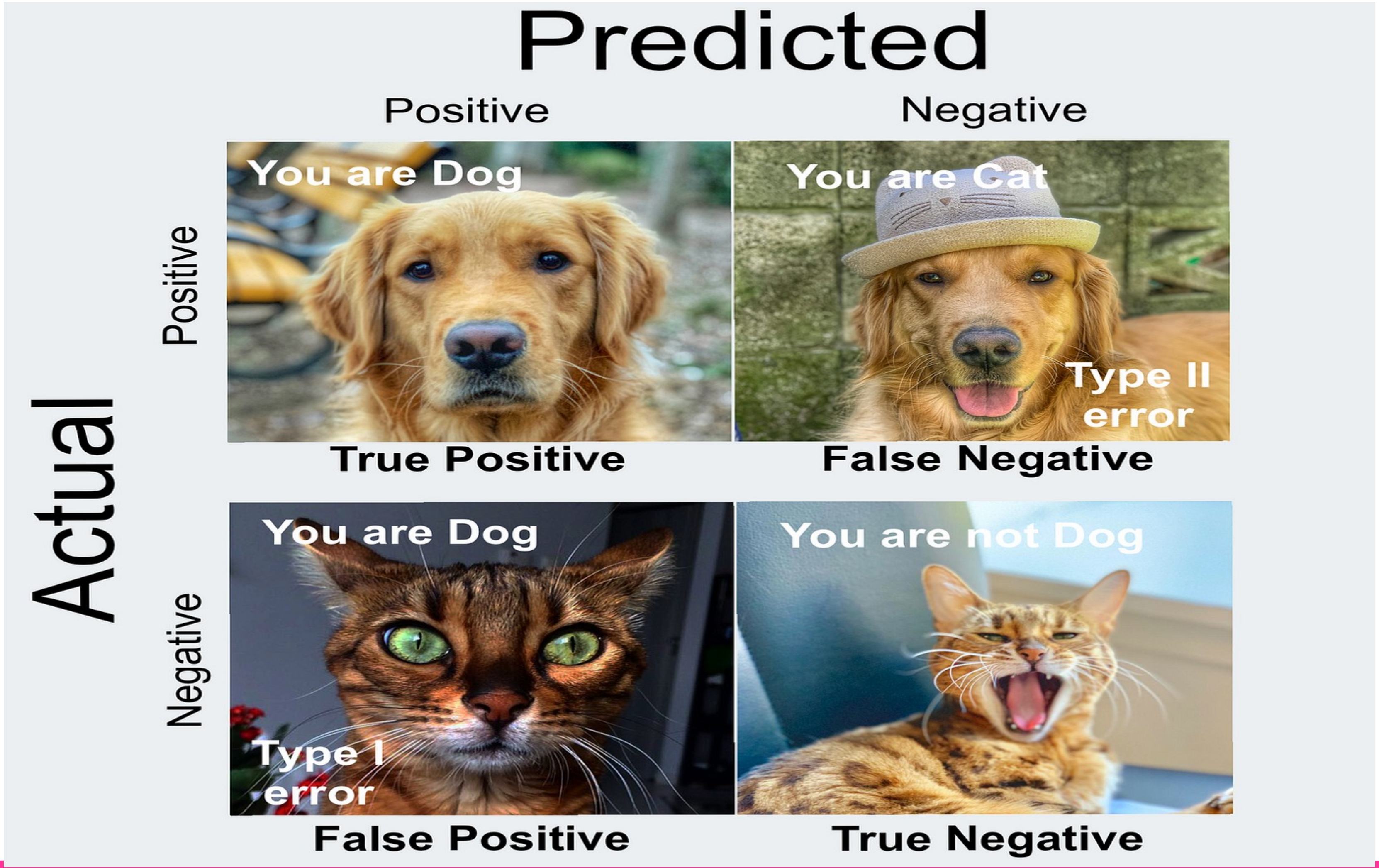


		Actual Value	
		Positive	Negative
Predicted Value	Positive	TP (True Positive)	FP (False Positive)
	Negative	FN (False Negative)	TN (True Negative)

- True Positive (TP) : Observation is positive, and is predicted to be positive.
- False Negative (FN) : Observation is positive, but is predicted negative.
- True Negative (TN) : Observation is negative, and is predicted to be negative.
- False Positive (FP) : Observation is negative, but is predicted positive.

Classification Algorithms Evaluation Metrics

Confusion Matrix:



Classification Algorithms Evaluation Metrics

Precision :

- It explains how many of the correctly predicted cases actually turned out to be positive.
- Precision is useful in the cases where False Positive is a higher concern than False Negatives.
- The importance of Precision is in music or video recommendation systems, e-commerce websites, etc.

Precision for a label is defined as the number of true positives divided by the number of predicted positives.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

Classification Algorithms Evaluation Metrics

Recall (Sensitivity)

- It explains how many of the actual positive cases we were able to predict correctly with our model.
- Recall is a useful metric in cases where False Negative is of higher concern than False Positive.
- It is important in medical cases where it doesn't matter whether we raise a false alarm but the actual positive cases should not go undetected!

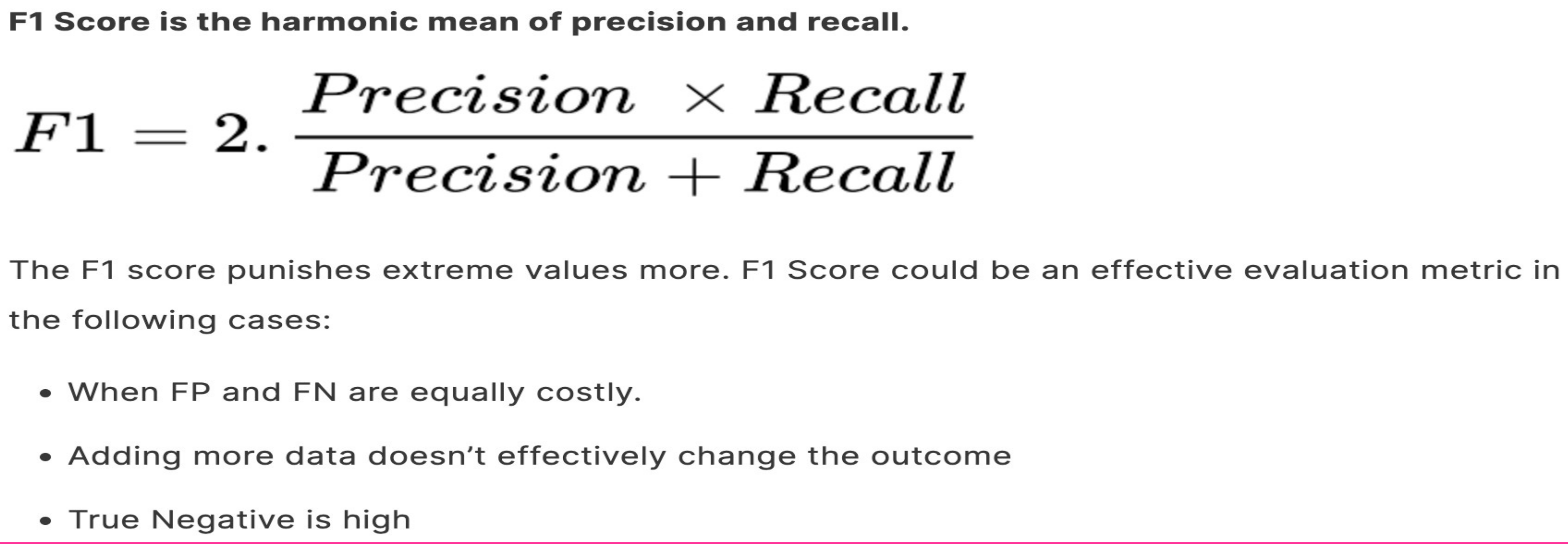
Recall for a label is defined as the number of true positives divided by the total number of actual positives.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

Classification Algorithms Evaluation Metrics

F1 Score:

- It gives a combined idea about Precision and Recall metrics. It is maximum when Precision is equal to Recall.



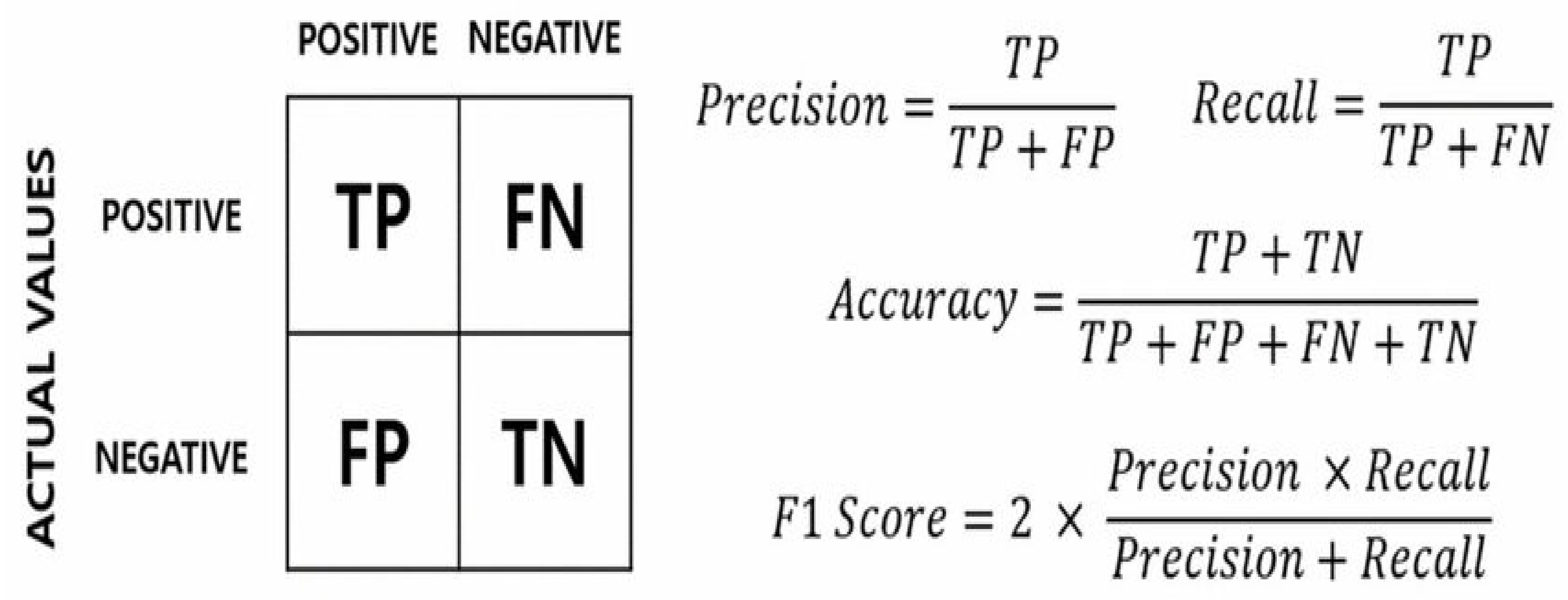
F1 Score is the harmonic mean of precision and recall.

$$F1 = 2.\frac{Precision \times Recall}{Precision + Recall}$$

The F1 score punishes extreme values more. F1 Score could be an effective evaluation metric in the following cases:

- When FP and FN are equally costly.
- Adding more data doesn't effectively change the outcome
- True Negative is high

Classification Algorithms Evaluation Metrics



		POSITIVE	NEGATIVE
ACTUAL VALUES	POSITIVE	TP	FN
	NEGATIVE	FP	TN

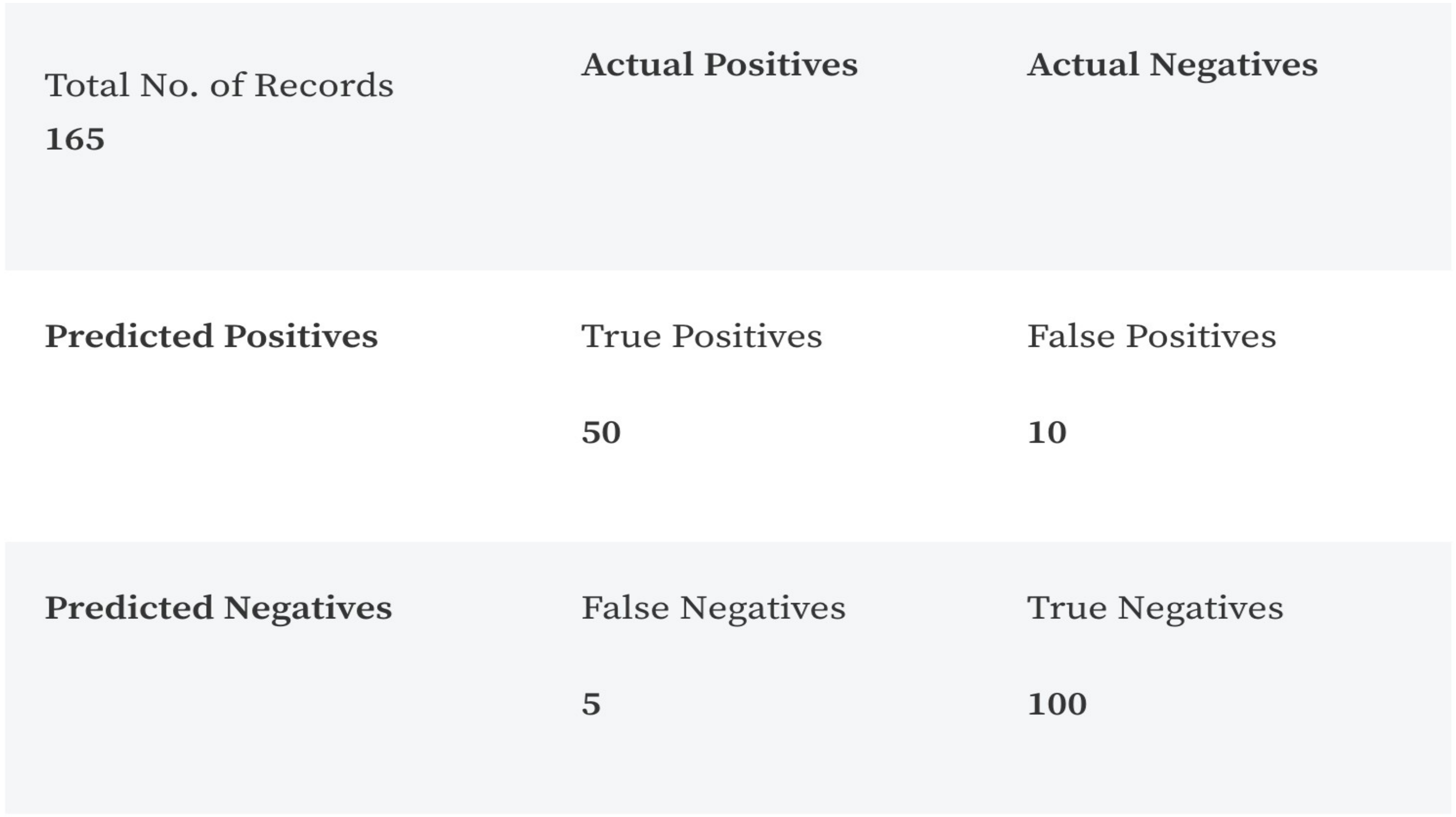
$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$F1\,Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Classification Algorithms Evaluation Metrics

- Suppose you have a machine learning model to predict if a credit card transaction was fraudulent or not. The following confusion matrix summarizes the prediction made by the model. Calculate **Accuracy** and **F1 Score** for classification model.



Total No. of Records 165	**Actual Positives**	**Actual Negatives**
Predicted Positives	True Positives 50	False Positives 10
Predicted Negatives	False Negatives 5	True Negatives 100

Classification Algorithms Evaluation Metrics

AUC-ROC:

- The **Receiver Operator Characteristic** (ROC) is a probability curve that plots the TPR(True Positive Rate) against the FPR(False Positive Rate) at various threshold values and separates the 'signal' from the 'noise'.
- The **Area Under the Curve (AUC)** is the measure of the ability of a classifier to distinguish between classes.
- When **AUC is equal to 1**, the classifier is able to perfectly distinguish between all Positive and Negative class points.
- **When AUC is equal to 0,** the classifier would be predicting all Negatives as Positives and vice versa.

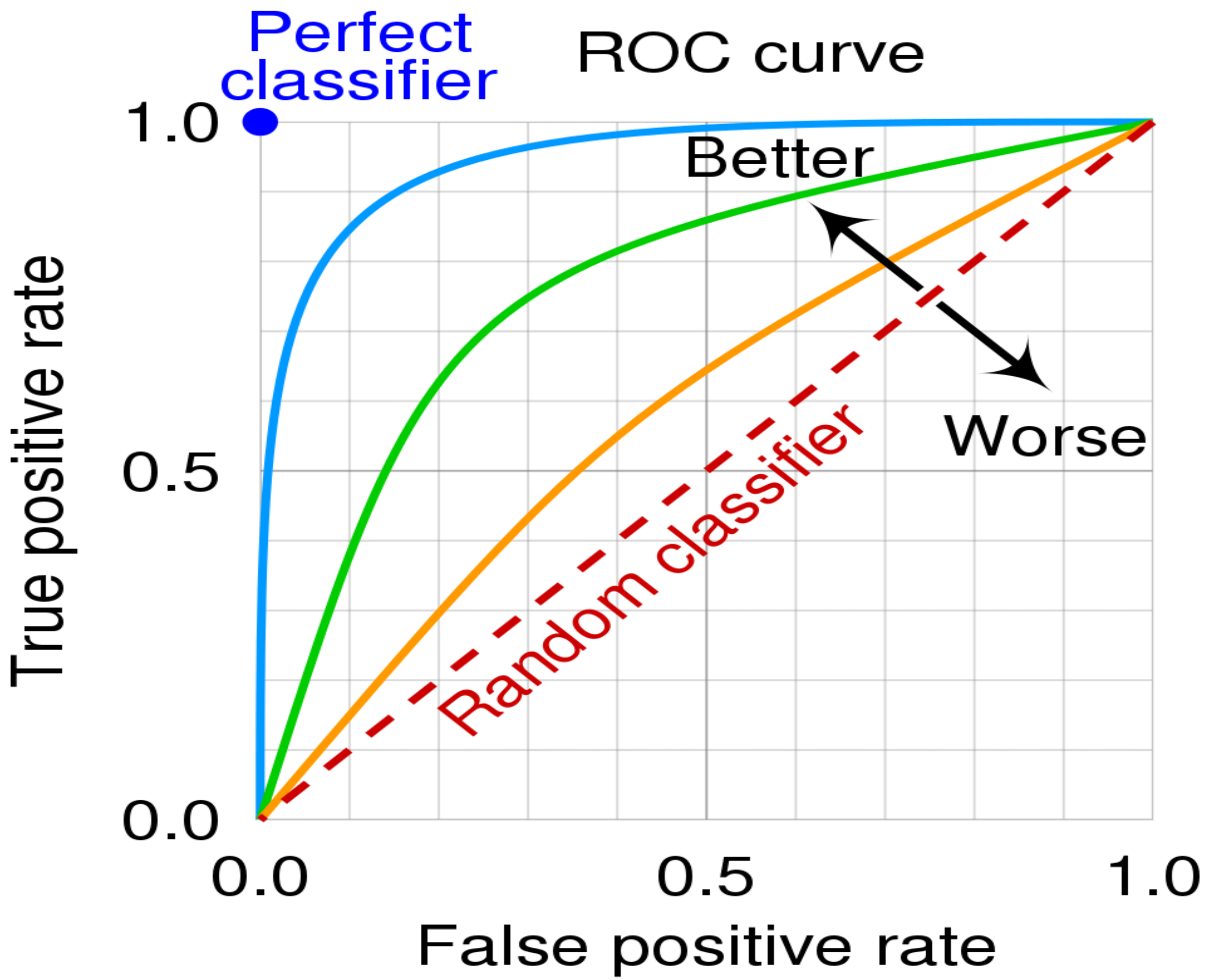
Classification Algorithms Evaluation Metrics

Working of AUC-ROC:

- In a ROC curve, the X-axis value shows False Positive Rate (FPR), and Y-axis shows True Positive Rate (TPR).
- Higher the value of X means higher the number of False Positives(FP) than True Negatives(TN), while a higher Y-axis value indicates a higher number of TP than FN.
- So, the choice of the threshold depends on the ability to balance between FP and FN.
- Models with a **high AUC** are called **models with good skills.**

Classification Algorithms Evaluation Metrics

AUC-ROC:



THANK YOU