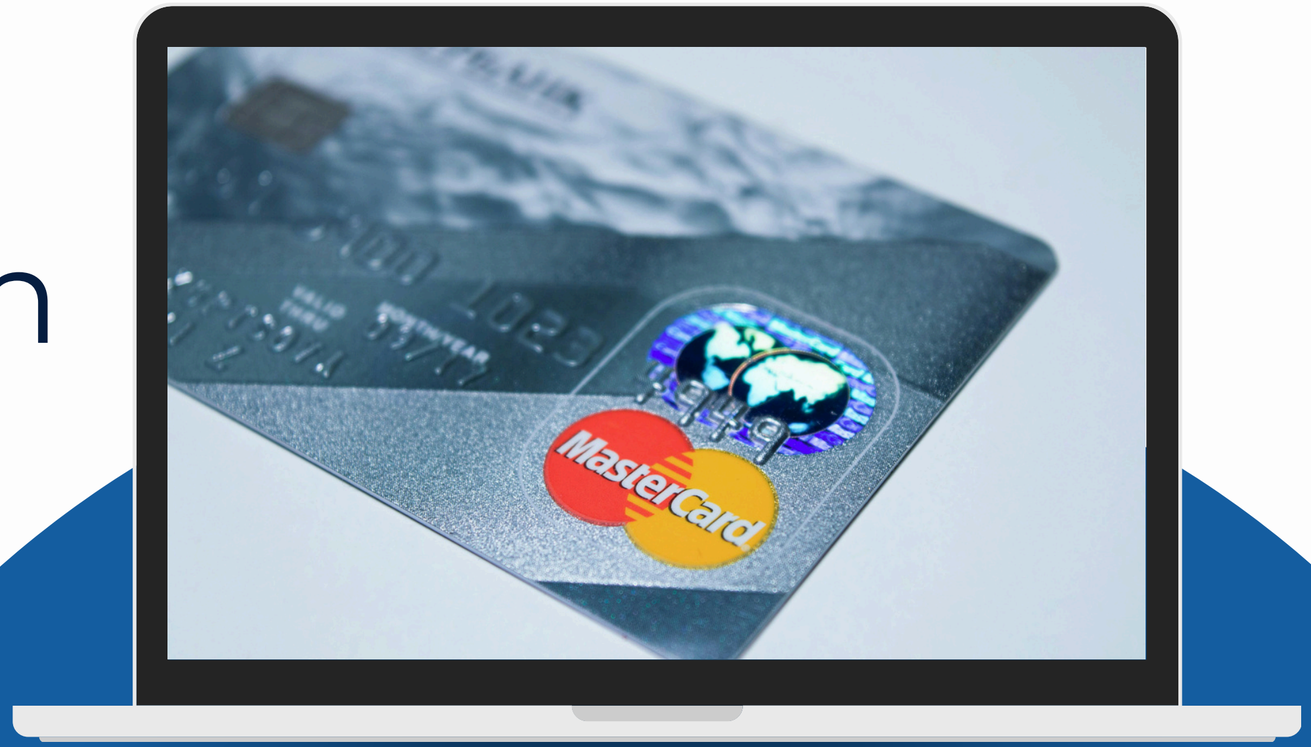
IEEE

Credit Card Fraud Detection

Using Machine learning



By:
Fatma Salah
Youssef Mohamed
Hany Tamer

Overview

Problem definition

What is credit card fraud

Definition

Credit card fraud is the unauthorized use of a credit card or credit card information to make fraudulent purchases or withdrawals. This can involve the theft of the physical card or the theft of card details like account numbers, expiration dates, and security codes, which are then used to make unauthorized transactions online, over the phone, or in person.

It’s Impact

In business:

- Chargebacks
- reputation damage
- increased security costs

To consumers :

- Identity theft
- financial stress
- damaged credit scores

How fraudulent transactions occur

Card-Not-Present CNP Fraud

Online and phone transactions using stolen card information

Skimming and Shimming

Devices steal card data at ATMs and point-of-sale terminals

Phishing and Social Engineering

Tricking consumers into revealing card information

“449,032 reports of credit card fraud in 2024"
- Bankrate, 2025

The Dataset

The dataset info

We got project's dataset from Kaggle.
The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days
It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues
it has
284807 rows
31 columns

Dataset columns

V1
V2
V3
:
V28

PCA

Amount

Time

Class (our target)

Dataset Challenges

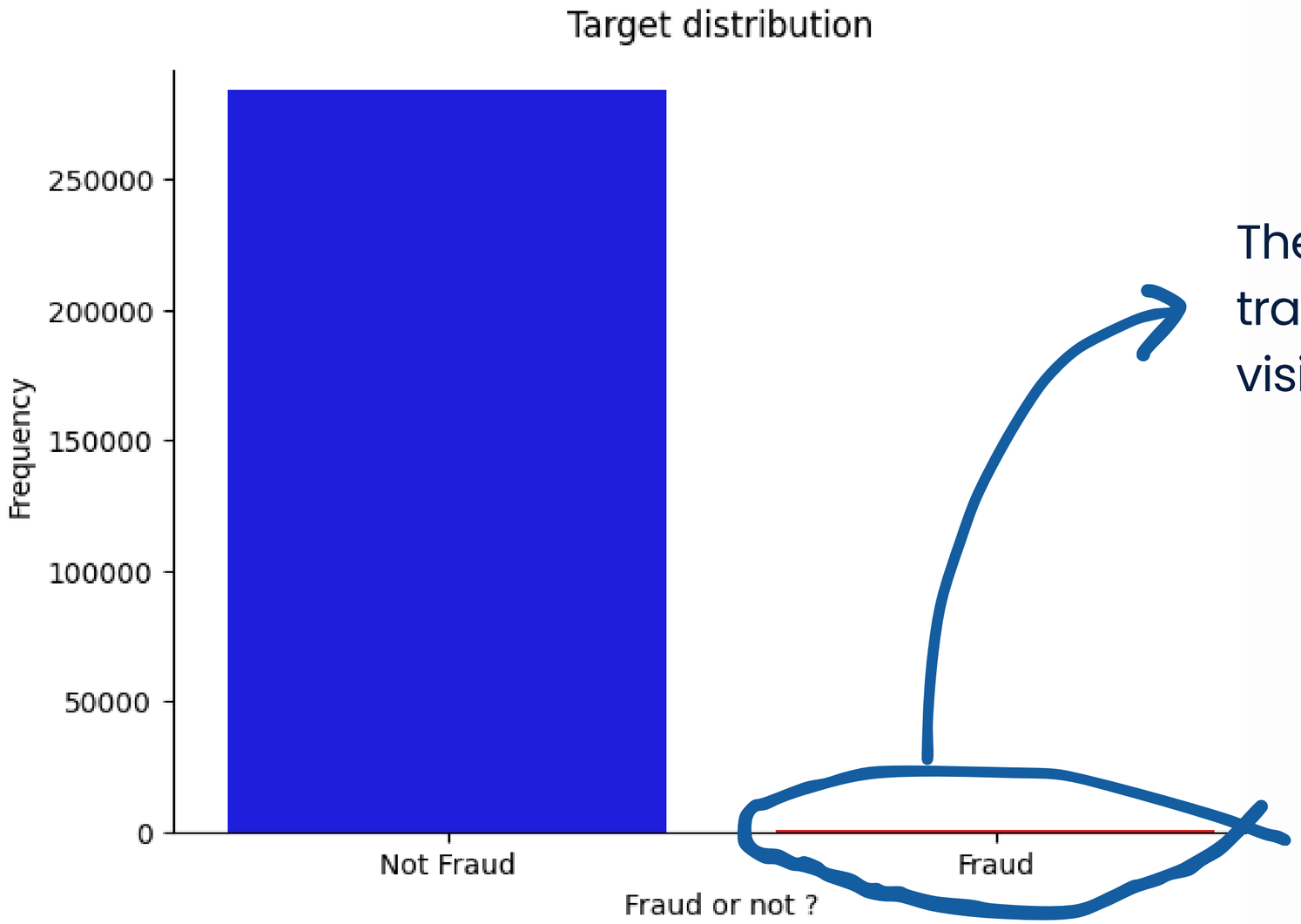
We have here extreme class imbalance

99.83%	VS	0.17%
Legitimate		Fraudulent

We have (in our dataset) 492 fraud transaction out of 284,807

Which is very bad during ML modeling, as it will definitely lead to a biased model

Visualization from our code



The frequency of fraudulent transactions is so small that it's barely visible in the graph

The Cost of Bias in ML

Standard ML techniques such as **Decision Trees** and **Logistic Regression** are biased towards the majority class, which means they tend to predict most transactions as real (non-fraud), even if some of them are actually fraudulent.

Also, it will cause **overfitting**, which means that the model's accuracy and score will be too high during training, but in real-life scenarios, its performance will be much lower

The solution

There are several ways to solve imbalance in data, we used the Resampling Techniques

Resampling Techniques has two types:

- **Oversampling:** Increase the number of instances in the minority class by randomly duplicating samples or generating synthetic samples. like SMOTE (Synthetic Minority Over-sampling Technique)
- **Undersampling:** Reduce the number of instances in the majority class by randomly removing samples

For Example

If we have 492 rows only for fraud transaction
we will have 10,000

and if we have 284,315 rows for real transactions we will have only 10,000

The solution

We experimented with 3 versions of the resampled dataset:

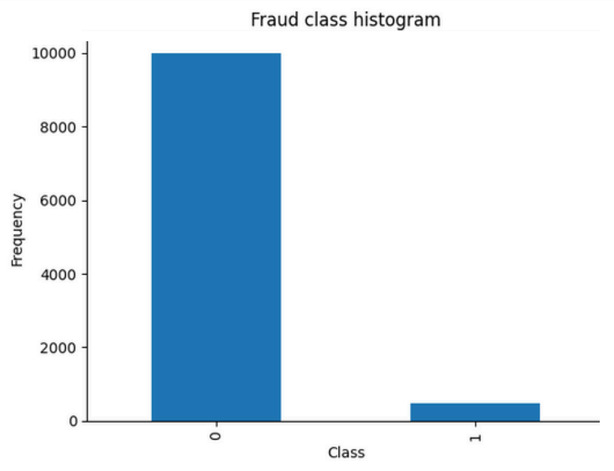
resampled_df ----> 10,000 not-fraud transactions, 492 fraud transactions

resampled_df1 ----> 10,000 not-fraud transactions, 10,000 fraud transactions

resampled_df2 ----> 5000 not-fraud transactions, 2500 fraud transactions

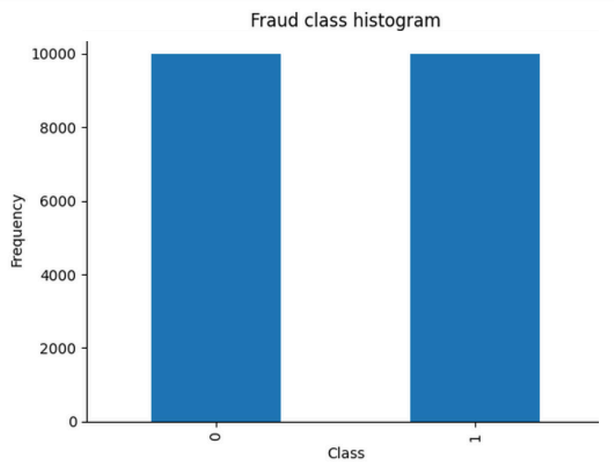
Freq. Visualizations

resampled_df



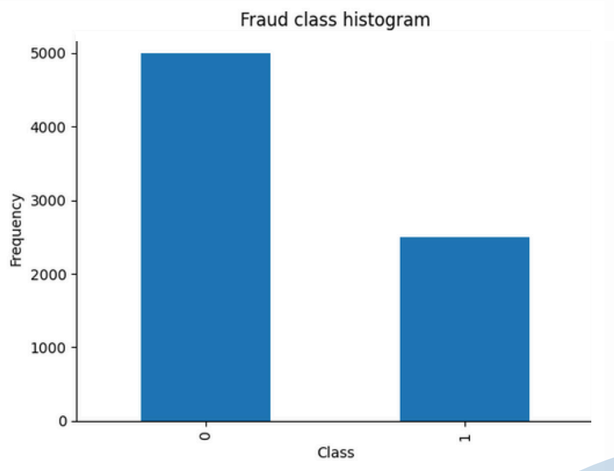
undersampling only

resampled_df1



undersampling and oversampling

resampled_df2



undersampling and oversampling

Feature selection

What is feature selection?

Feature selection is the process of choosing the most relevant input features (columns) from your dataset to use in building a machine learning model.

Why use feature selection?

- Improve model performance (accuracy, speed).
- Reduce overfitting (less noise = better generalization).
- Simplify the model (easier to understand and maintain).
- Reduce computation time (especially with large datasets).

Common feature selection methods:

- Filter methods – Use statistical tests (e.g., correlation, chi-square).
- Embedded methods – Done during model training (tree-based models).

Feature selection

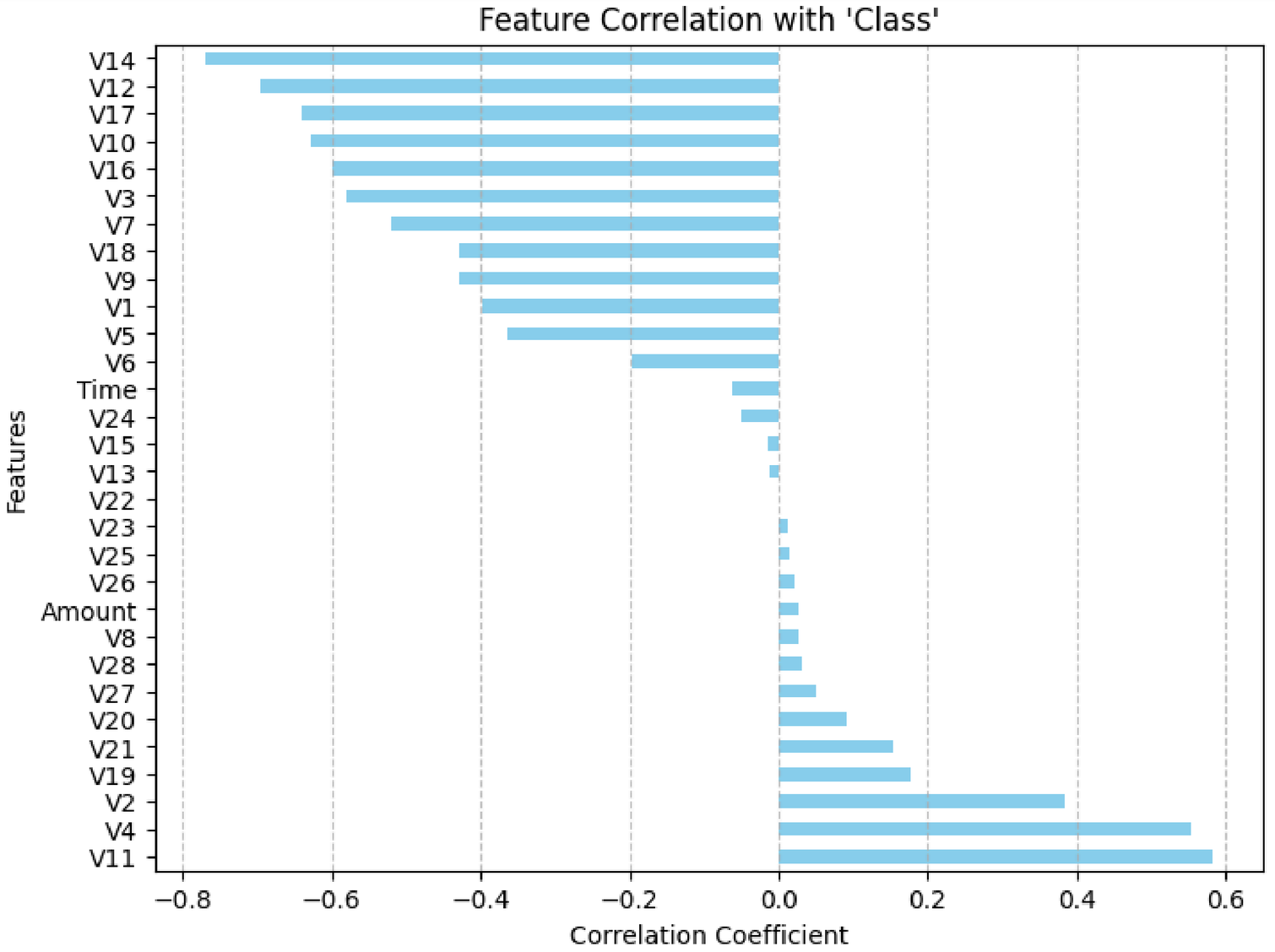
We used **correlation coeff.** between class and other features (columns), here is the **results:**

V11 ---> 0.582041
V4 ---> 0.554010
V2 ---> 0.384676
V19 ---> 0.177410
V21 ---> 0.153526
V20 ---> 0.091902
V27 ---> 0.050299
V28 ---> 0.031129
V8 ---> 0.026139
V14 ---> -0.769769

Amount ---> 0.025746
V26 ---> 0.021026
V25 ---> 0.015100
V23 ---> 0.012316
V22 ---> 0.001369
V13 ---> -0.012100
V15 ---> -0.013670
V24 ---> -0.049205
Time ---> -0.062150
V12 ---> -0.696186

V6 ---> -0.196730
V5 ---> -0.363579
V1 ---> -0.398566
V9 ---> -0.429177
V18 ---> -0.429504
V7 ---> -0.521254
V3 ---> -0.579336
V16 ---> -0.598549
V10 ---> -0.628774
V17 ---> -0.639332

Correlation visualization



Feature selection

After seeing the relation between class (our target cloumn) and all features we decided to **drop (remove) Time** column as their correlation with class column is too low
corr between class and time ---> -0.06

Note:

A correlation coefficient close to -1 or +1 indicates a strong relationship, whether negative or positive.
So, for example, -0.8 is a stronger correlation than +0.06, even though it's negative.

Feature scaling

What is feature scaling?
Feature scaling is a fundamental preprocessing step in machine learning aimed at ensuring that numerical features have a similar scale. This is pivotal because many machine learning algorithms perform better or converge faster when the input numerical features are on a similar scale.

For example (from our data):
The V3 column has a range between -31.1 and 3.8,
while the Amount column ranges from 0 to 7879.
This large difference in numerical scales can bias the model toward features with larger values, which may lead to incorrect predictions.

The solution

There are several ways to solve skewness in data, we used the Standardization technique (standard scaler)

What **Standardization Techniques** do?

- **Definition:** each feature is transformed to have a mean of 0 and a standard deviation of 1
- **formula:** $Z = (x - \mu) / \sigma$

For Example

If the maximum value of Amount column before scaling is 7879

after scaling the max. value of amount column is 33

ML model building

choosing our model

Before building our model, we need to clearly understand what we're trying to achieve.
Our target column, labeled 'Class', contains only two values: 0 and 1.
This means we are dealing with a binary classification problem, so we need models that can predict one of two possible outcomes.

We will experiment with the following models:

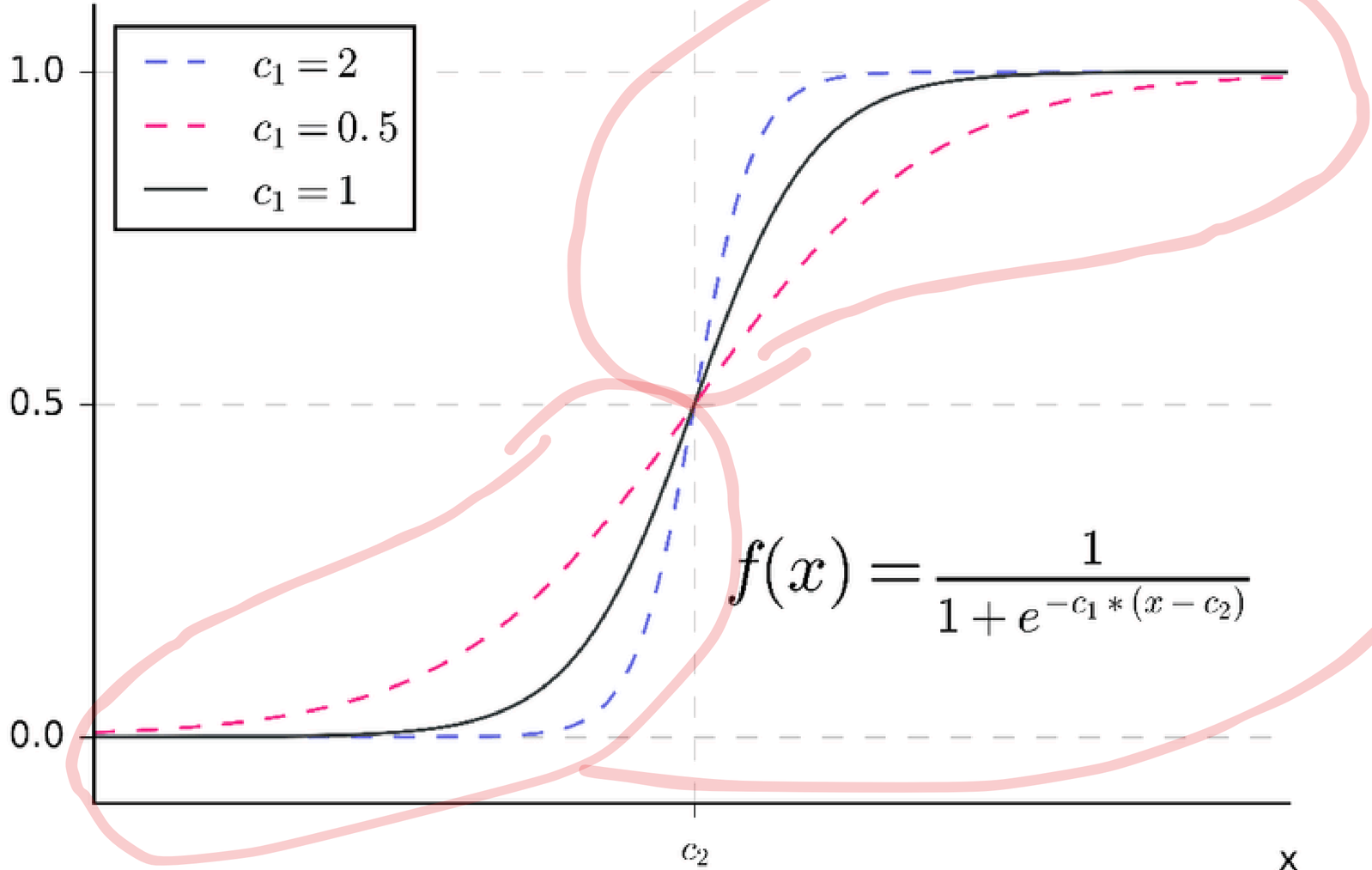
- **Logistic regression**
- **SVC (Support Vector Classifier)**
- **KNN (K-Nearest Neighbors)**
- **Decision Tree**
- **Random Forest**
- **Gradient Boosting Classifier**

Logistic regression

Definition:

Logistic Regression is a supervised learning algorithm used for binary classification problems. It estimates the probability that a given input belongs to a particular class by applying the logistic (sigmoid) function to a linear combination of the input features. The output is always between 0 and 1, representing the predicted probability. If the probability is greater than a threshold (usually 0.5), it predicts class 1, otherwise class 0. Although it's called "regression", it is used for classification tasks. It's best suited when the data is linearly separable, meaning a straight line (or plane in higher dimensions) can divide the two classes.

It’s shape



if probability > 0.5 so classification is 1

As you see it’s range always between 0 and 1 (binary classification)

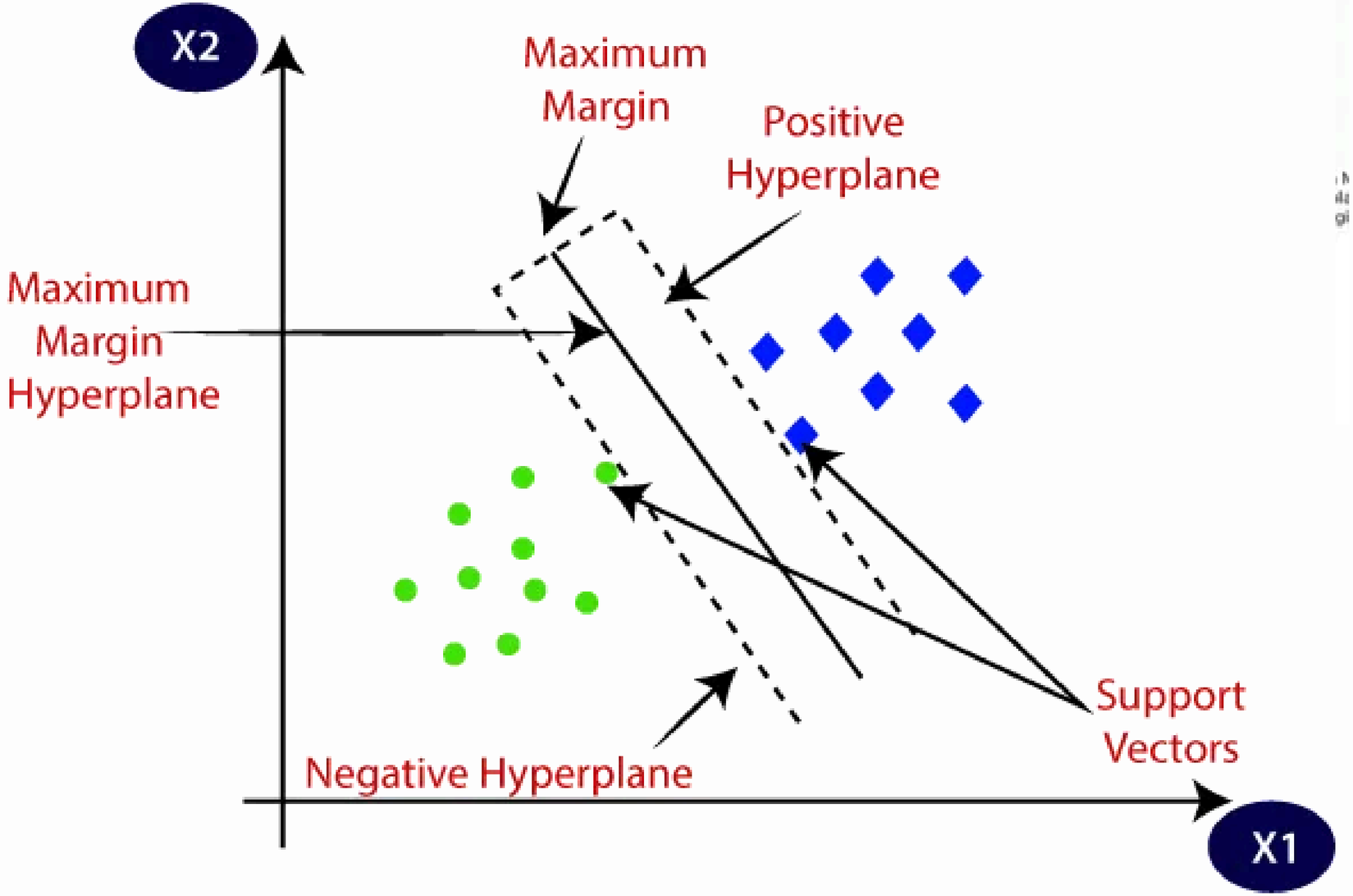
if probability < 0.5 so classification is 0

SVC (Support Vector Classifier)

Definition:

Support Vector Classifier is a powerful classification algorithm that tries to find the optimal hyperplane that best separates the data points into two classes. A hyperplane is simply a decision boundary. The unique aspect of SVC is that it tries to maximize the margin, the distance between the hyperplane and the nearest data points from each class (called support vectors). If the data is not linearly separable, SVC can use kernel functions (like RBF or polynomial) to project the data into a higher-dimensional space where a linear separator may exist. This makes SVC especially useful for complex, non-linear datasets.

Visualization

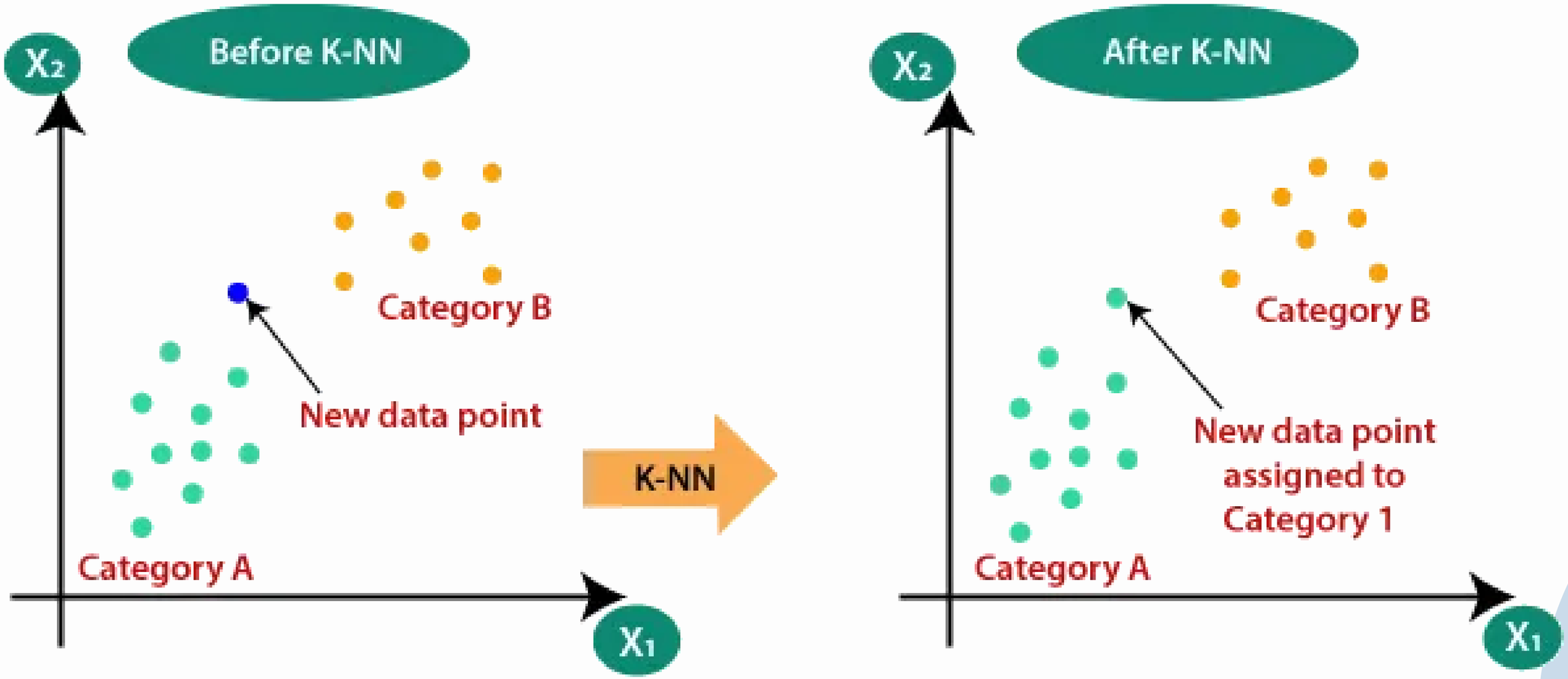


KNN (K-Nearest Neighbors)

Definition:

KNN is a non-parametric, instance-based learning algorithm used for both classification and regression, but more commonly for classification. It does not make assumptions about the data distribution. Instead of learning an explicit model during training, KNN simply stores all the training data. When it’s time to classify a new instance, the algorithm finds the K closest training samples (neighbors) in the feature space, based on distance metrics like Euclidean distance, and assigns the most common class among them. It’s intuitive and works well for small, clean datasets, but struggles with high-dimensional or noisy data.

Visualization

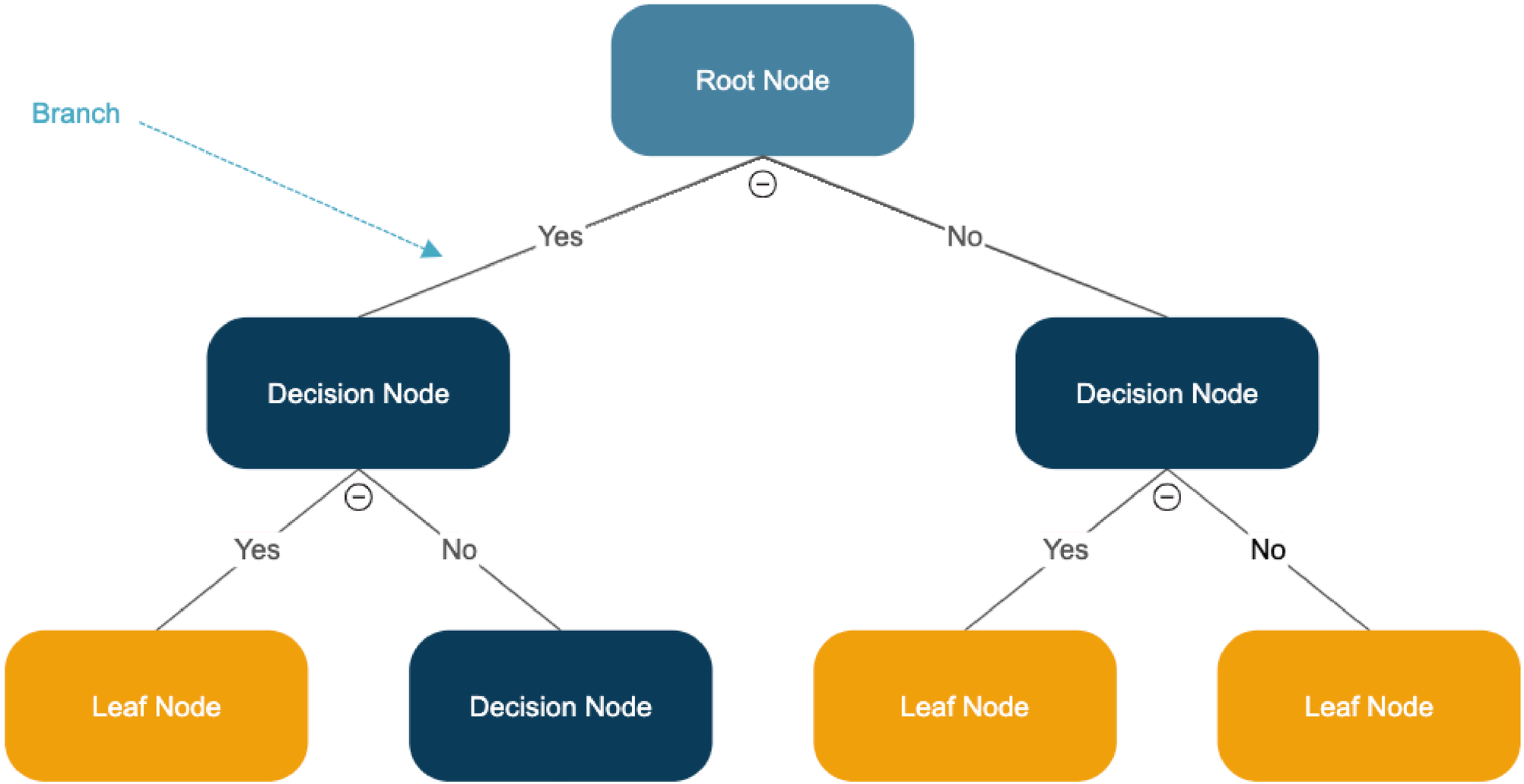


Decision Tree

Definition:

A Decision Tree is a tree-structured model used for classification and regression. It splits the dataset into subsets based on feature values through a series of questions (conditions). Each node represents a feature and each branch represents a decision rule. The process continues until it reaches a leaf node, which represents the final class label. It builds the tree using criteria like Gini Impurity or Entropy (Information Gain). Decision Trees are very easy to interpret and explain because they mimic human decision-making, but without constraints they can become very complex and overfit the data.

Visualization

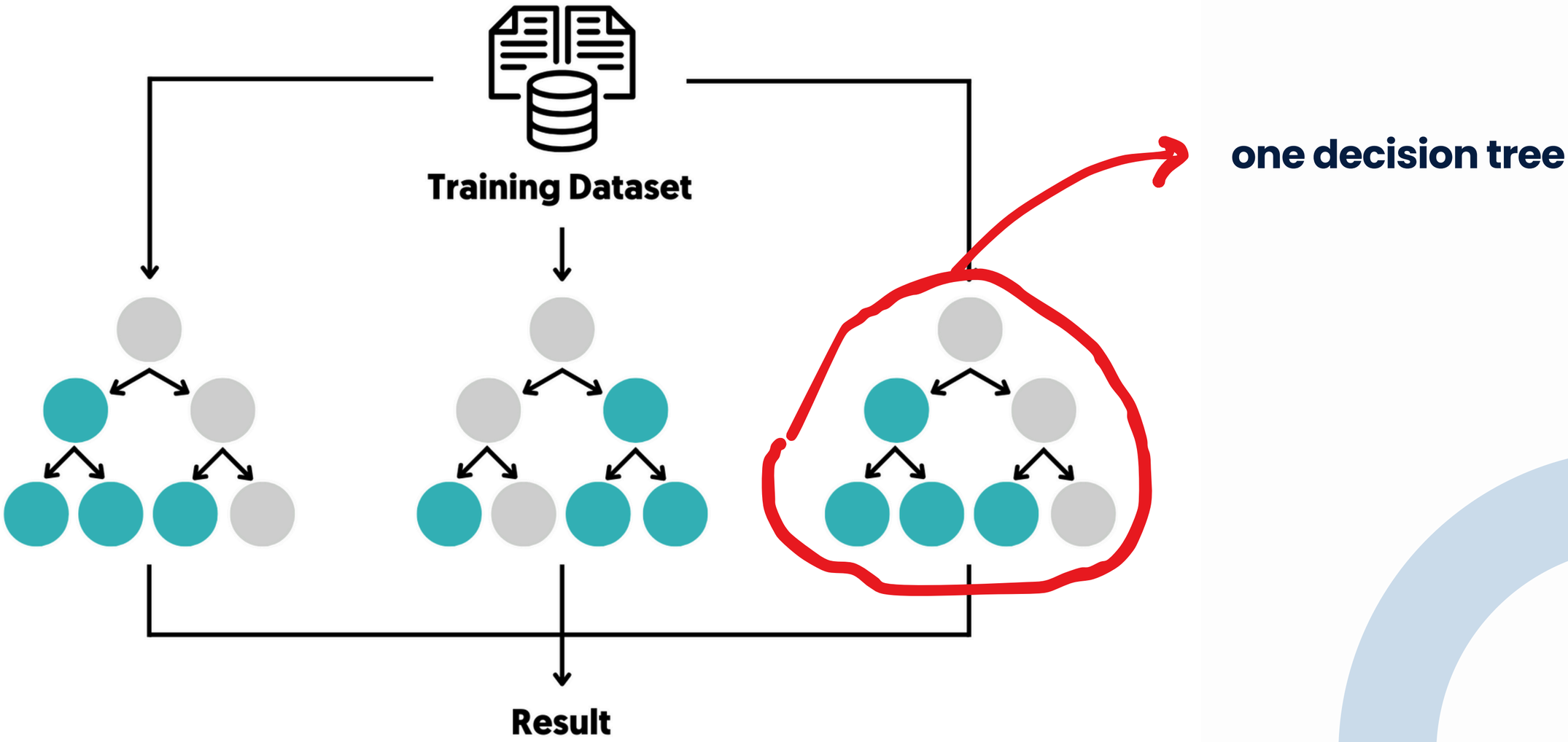


Random Forest

Definition:

Random Forest is an ensemble learning method that builds a large number of independent Decision Trees during training and combines their outputs using majority voting (for classification) or averaging (for regression). Each tree is trained on a random subset of the data and a random subset of features, which introduces diversity and reduces the risk of overfitting. The final prediction is made by aggregating the predictions of all trees. Random Forest is known for its robustness and accuracy, especially when dealing with large and complex datasets, or datasets with missing values and non-linear relationships.

Visualization

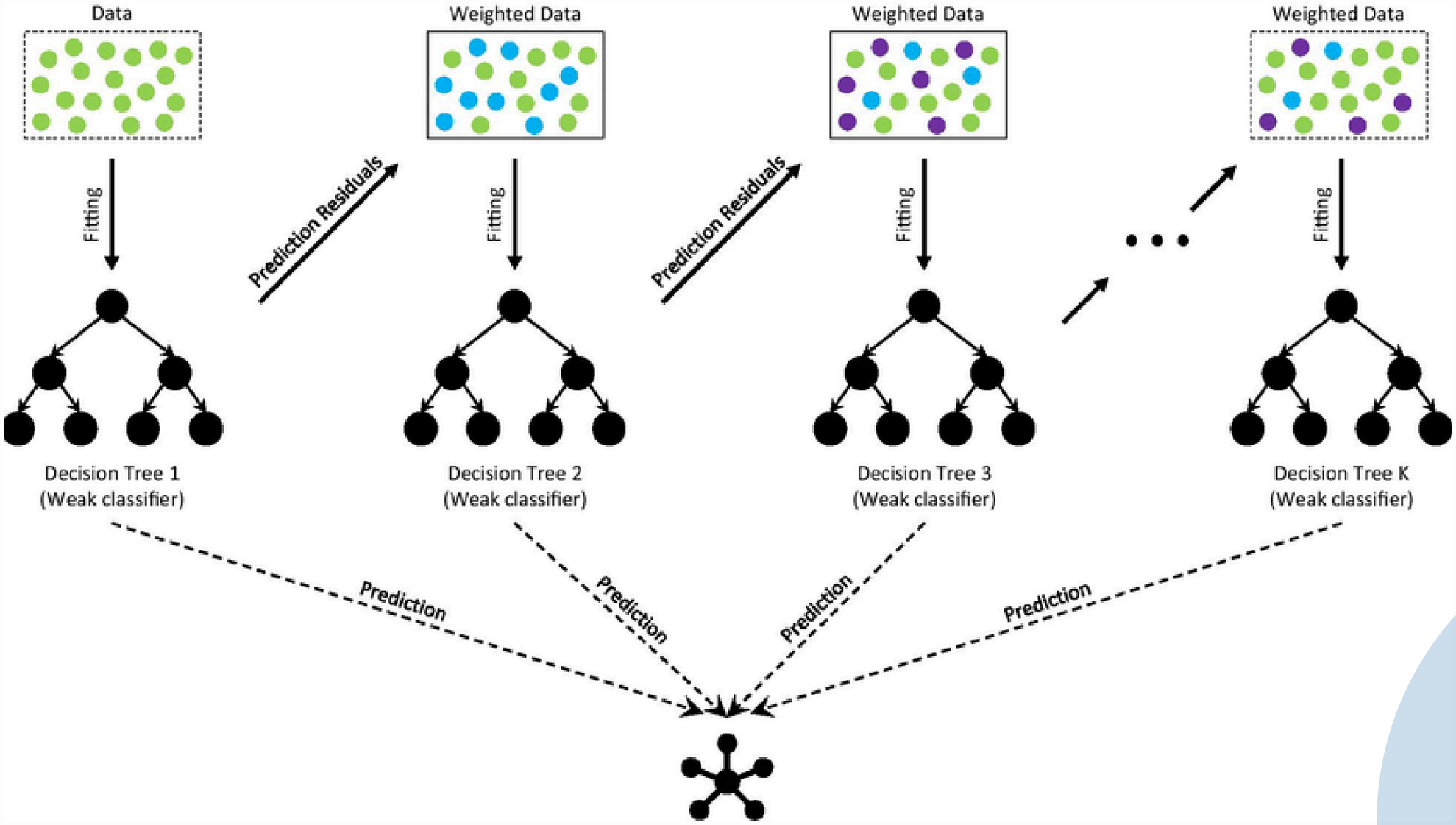


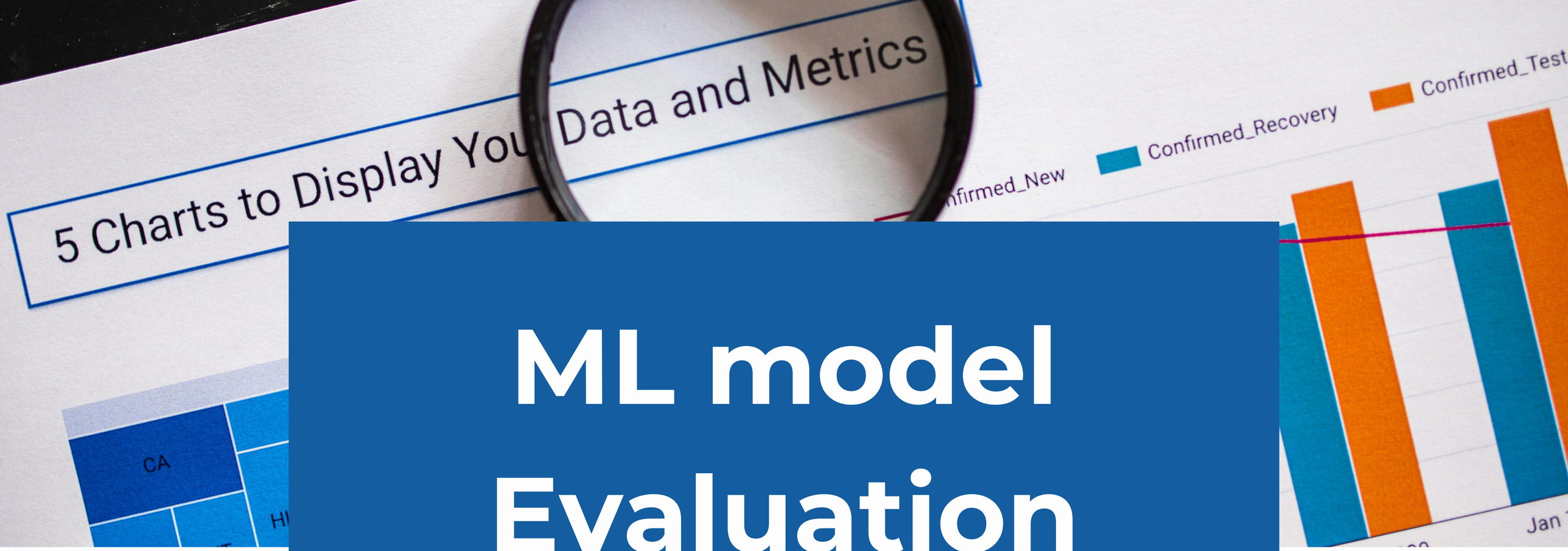
Gradient Boosting Classifier

Definition:

Gradient Boosting Classifier is an advanced boosting ensemble technique that builds a model sequentially in stages, where each new model is trained to correct the errors (residuals) of the previous one. It typically uses shallow Decision Trees (weak learners) as base models. Each step minimizes a loss function using gradient descent, hence the name "gradient boosting." Unlike Random Forest (which builds trees independently), Gradient Boosting focuses on improving accuracy step-by-step. It's very powerful and often used in winning solutions for machine learning competitions (like Kaggle), especially when tuned properly.

Visualization



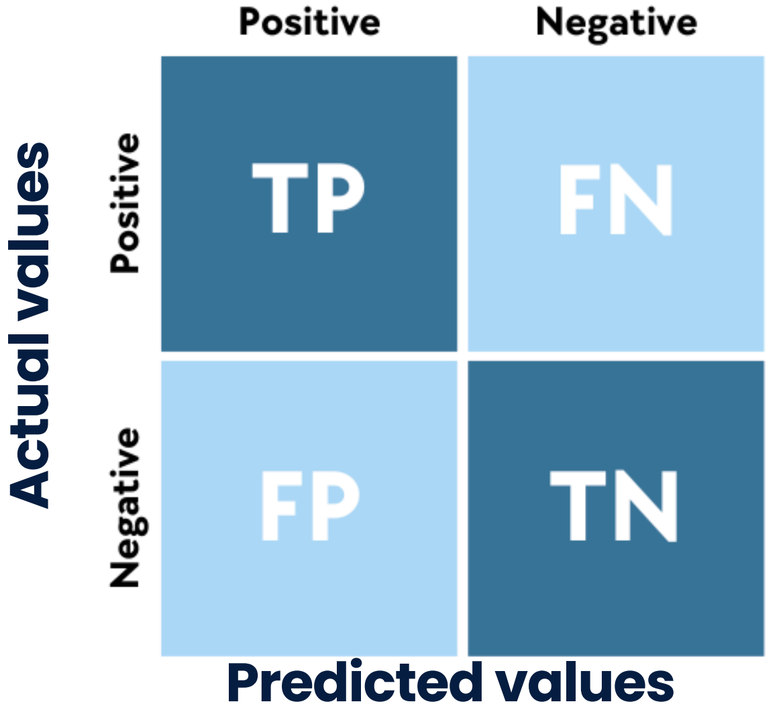


ML model Evaluation

Classification Model Evaluation

Before discussing classification model evaluation and the metrics we used, it's important to first explain a key concept: **the confusion matrix.**

The confusion matrix is a table that helps evaluate the performance of a classification model by comparing the predicted labels to the actual labels.



Classification Model Evaluation

There are several metrics to evaluate our models, we used:

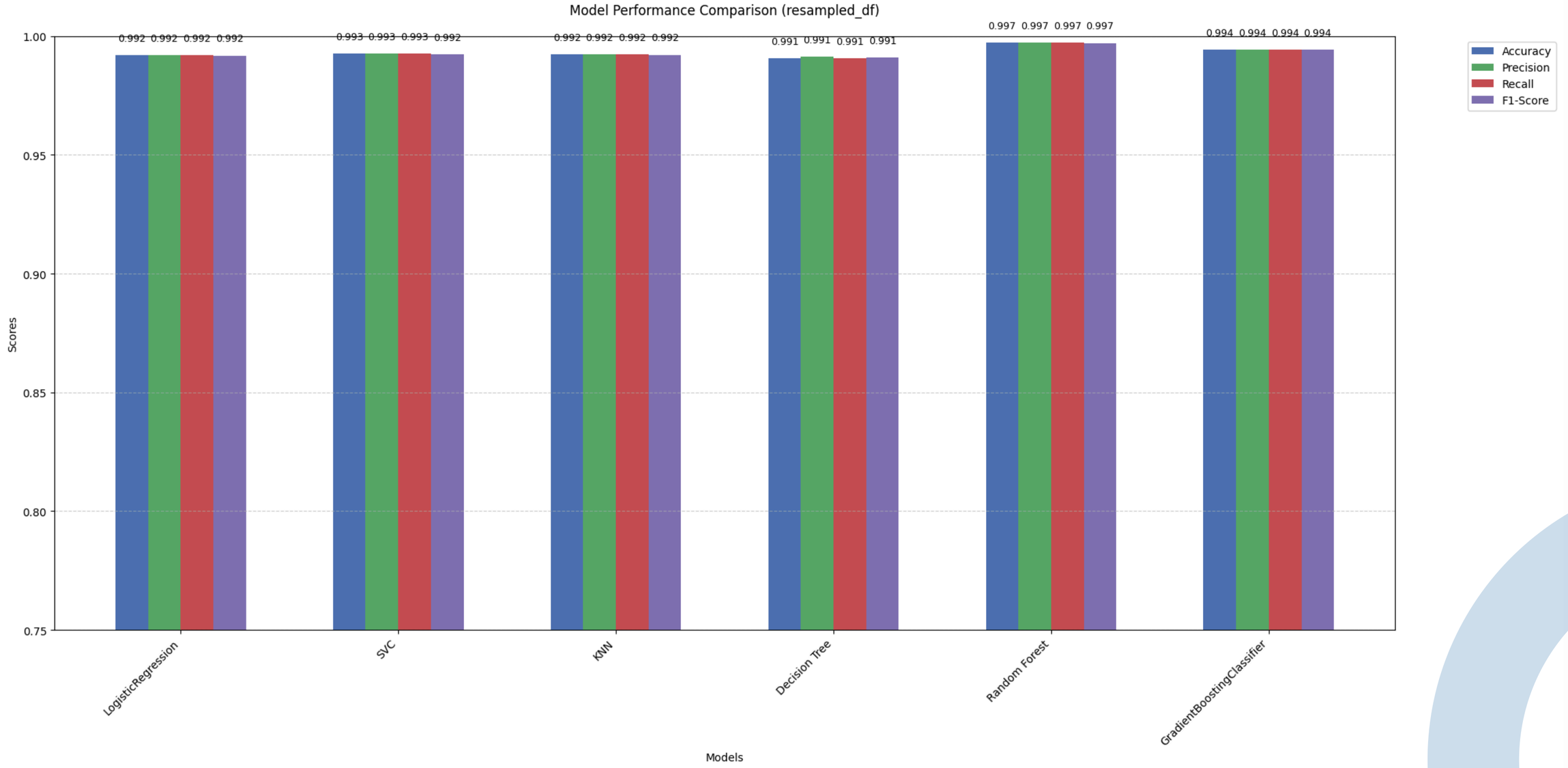
- **Accuracy:** How many predictions were correct overall
  It's formula → **(TP+TN) / (TP + TN + FP + FN)**

- **Precision:** Of all predicted positives, how many were actually positive
  It's formula → **(TP) / (TP + FP)**

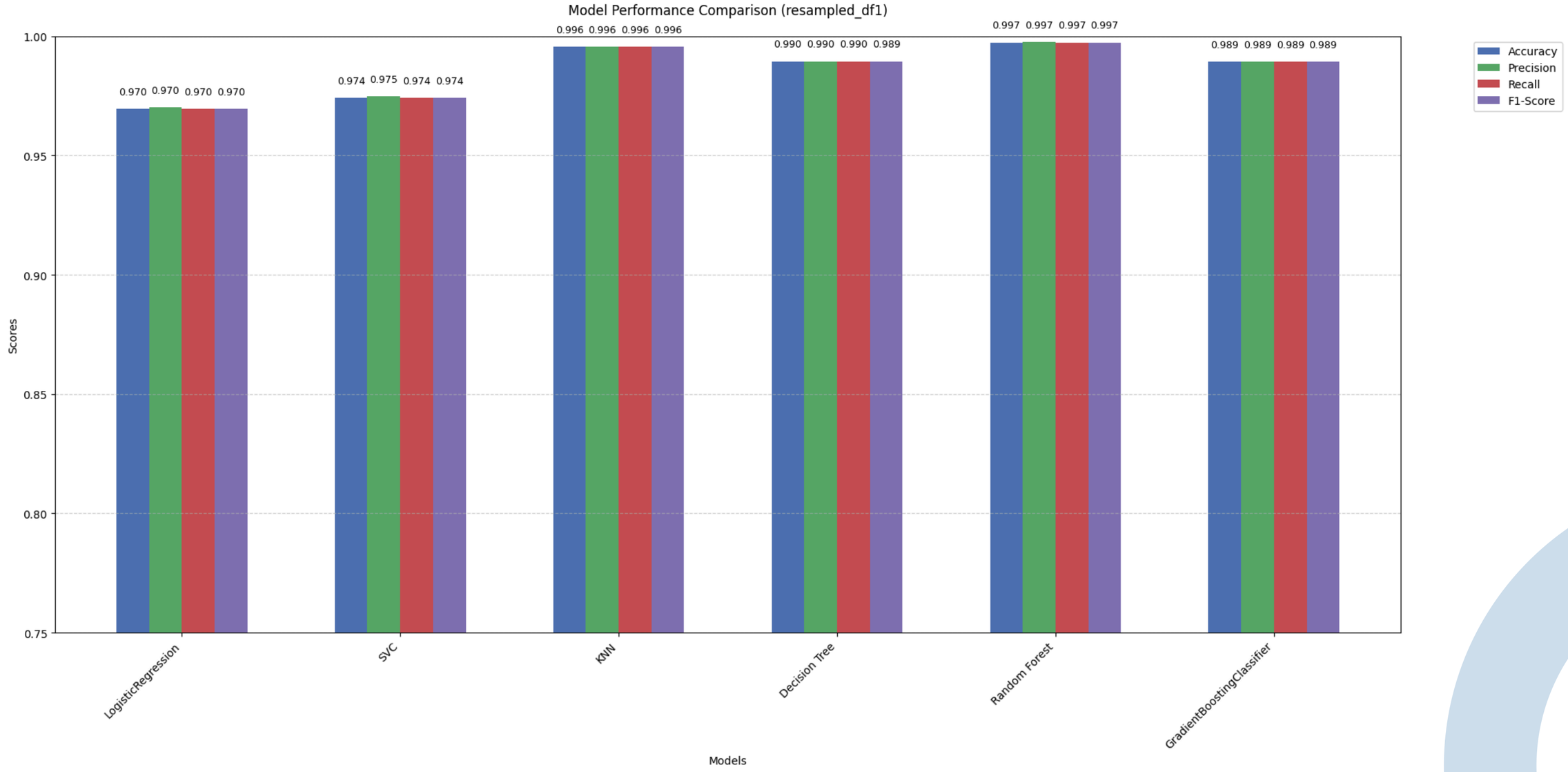
- **Recall:** Of all actual positives, how many did we correctly predict
  It's formula → **(TP) / (TP + FN)**

- **F1-Score:** Harmonic mean of precision and recall. Balances both.
  It's formula → **(2 × Precision × Recall) / (Precision + Recall)**

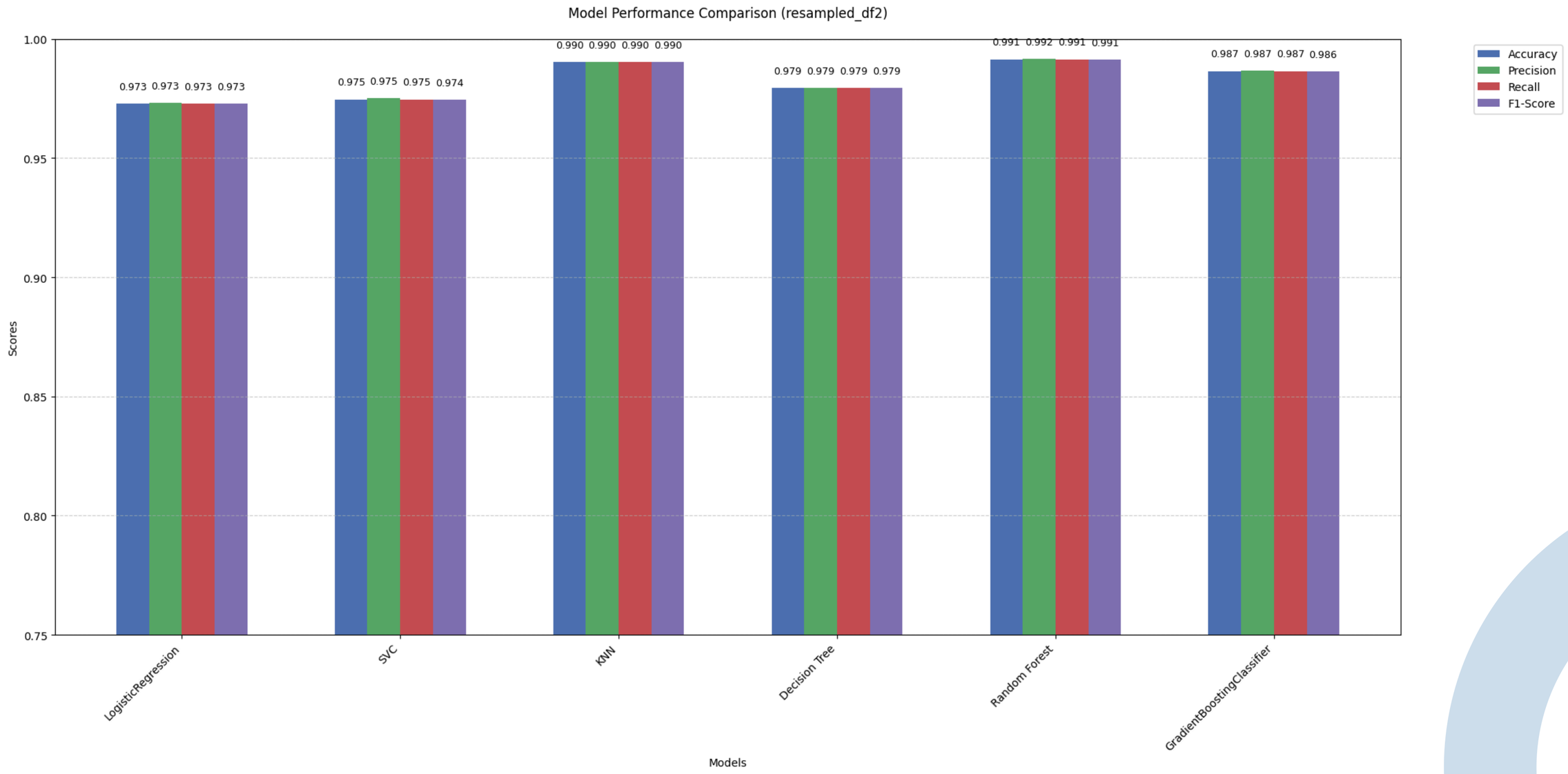
Model Comparison on resampled_ df



Model Comparison on resampled_ df1



Model Comparison on resampled_ df2



Hyperparameter tuning

Hyperparameter Tuning is the process of selecting the best combination of predefined settings (hyperparameters) for a machine learning model to achieve the best possible performance.

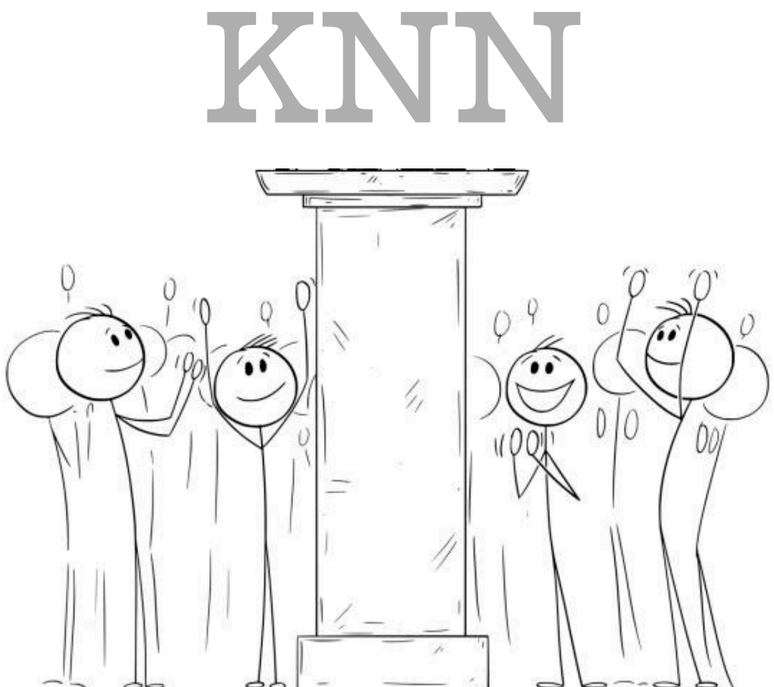
In our case
we used one of the most common hyperparameter tuning techniques, **Grid Search**, and evaluated each combination using **Cross-Validation** to ensure reliable results.

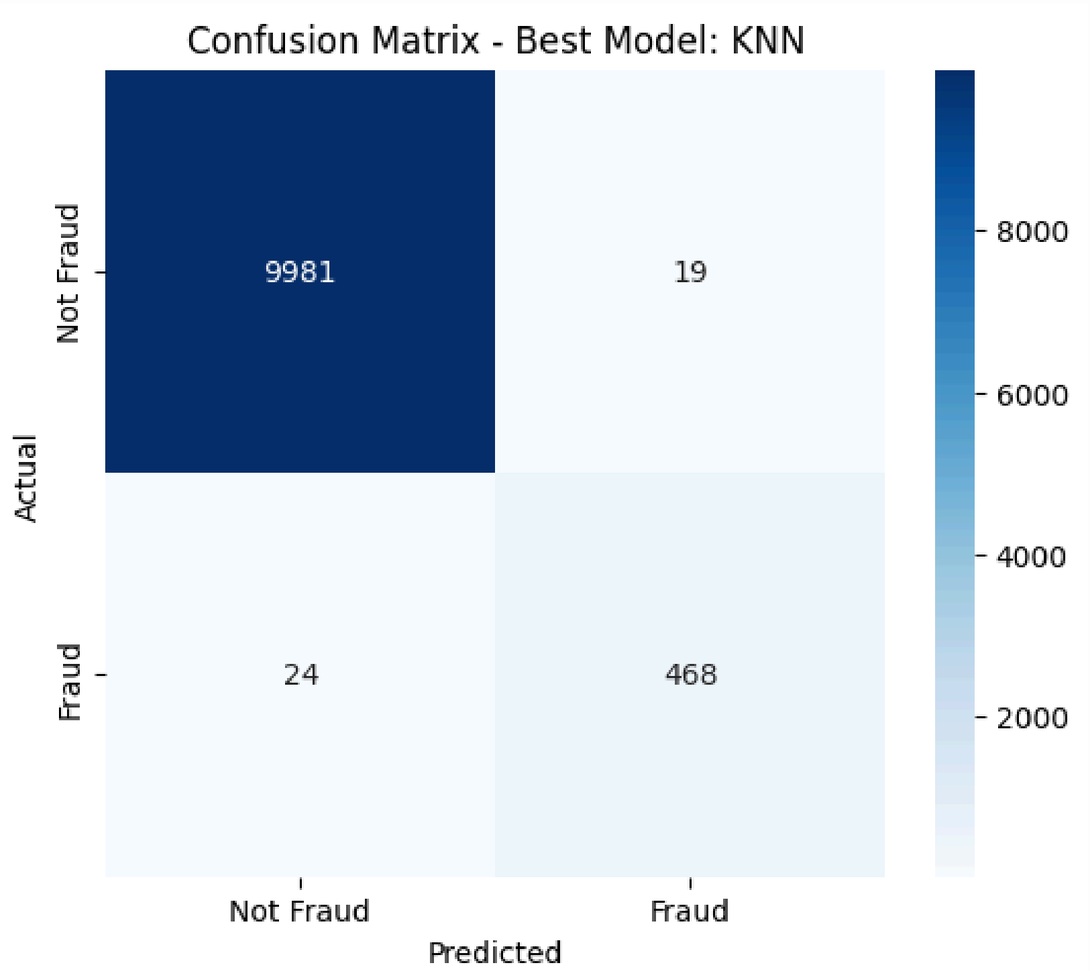
- **Grid search:** is a hyperparameter tuning method that tries all possible combinations of given hyperparameter values to find the best-performing setup.

- **Cross-validation:** is a model evaluation technique where the data is split into parts (folds); the model is trained and tested multiple times to ensure reliable performance on unseen data.

Hyperparameter tuning

After applying Grid Search and Cross-Validation, the **best hyperparameters** were found for the **KNN model.**





ML model Deployment

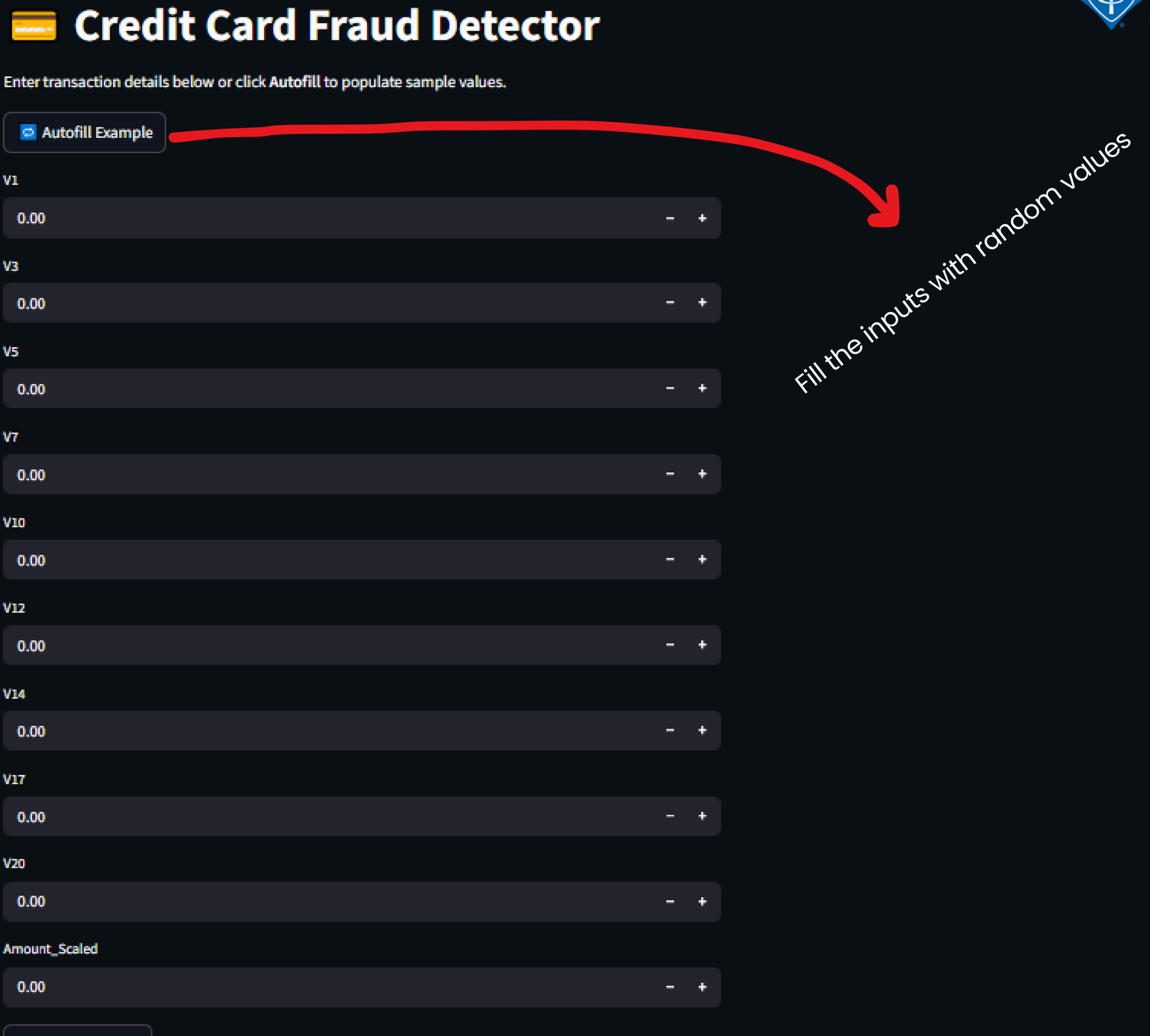
Model deployment

Model deployment means bringing the model to life, finally 🤩
To achieve this, we can use libraries like **Streamlit** or **Gradio** to make the model interactive and accessible. **In our case, we used the Streamlit library.**
Our app takes user **input** (features) and returns a **prediction**: Fraud or Not Fraud.
Since the dataset contains many features, we selected the following key features to be used as inputs in our interface.

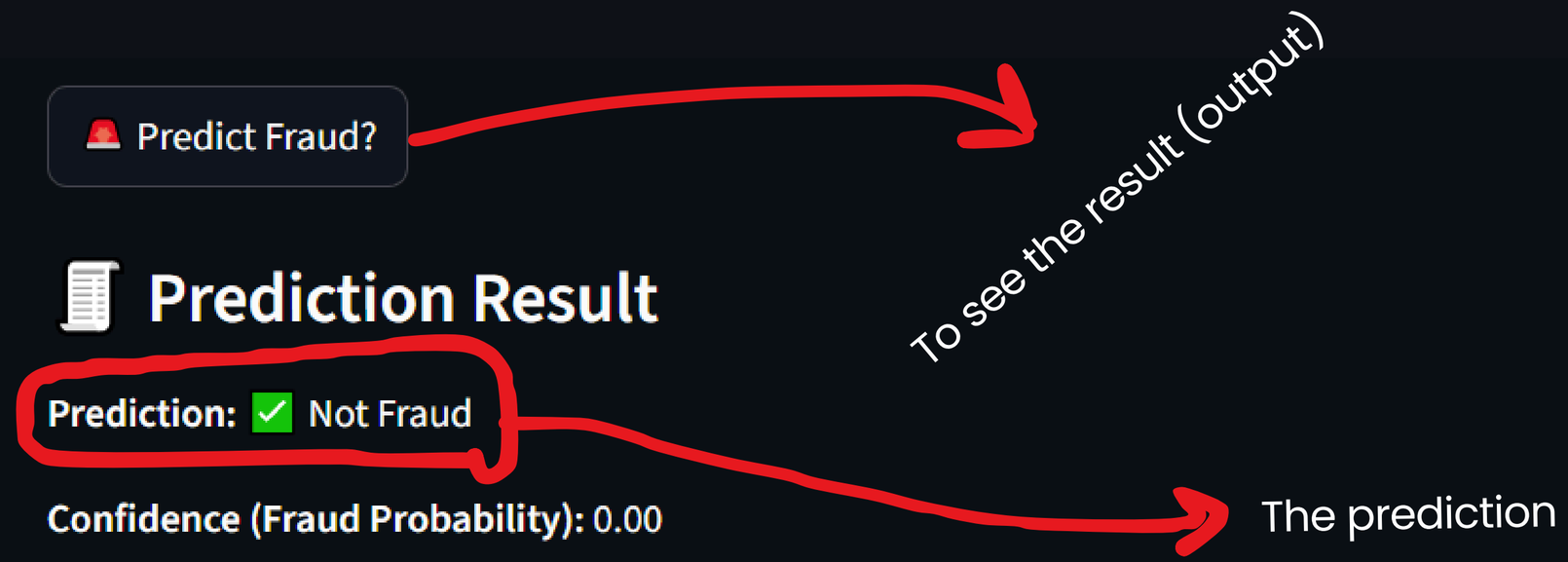
- V1
- V3
- V5
- V7
- V10
- V12
- V14
- V17
- V20
- Scaled_amount

Now... let's take a look at the GUI 😍

Inputs


Credit Card Fraud Detector
Enter transaction details below or click Autofill to populate sample values.
Autofill Example
V1
0.00
V3
0.00
V5
0.00
V7
0.00
V10
0.00
V12
0.00
V14
0.00
V17
0.00
V20
0.00
Amount_Scaled
0.00
Fill the inputs with random values

Output (The result)



🔍 Your Input Data:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
0	0	0	0	0	0	0	0	0	0	0	0	0	0

Summary

To sum up

Data processing

- We addressed the class imbalance in the dataset using a resampling technique.
- The Time column was dropped due to its low correlation with the target variable (Class).
- we scaled the Amount column using the Standard Scaler to normalize its values.

modeling

- We experimented with several models — Logistic Regression, SVC, KNN, Decision Tree, Random Forest, and Gradient Boosting Classifier, to determine which one would deliver the best performance.
- To evaluate each model, we used accuracy and recall as our main metrics
- We also applied Grid Search for hyperparameter tuning, and evaluated the results using cross-validation.

Model deployment

- Because the dataset contains many columns, we selected only 10 features to be used as inputs, and one column (the Class column) as the output.

THANK YOU!

by: Card Guardians team