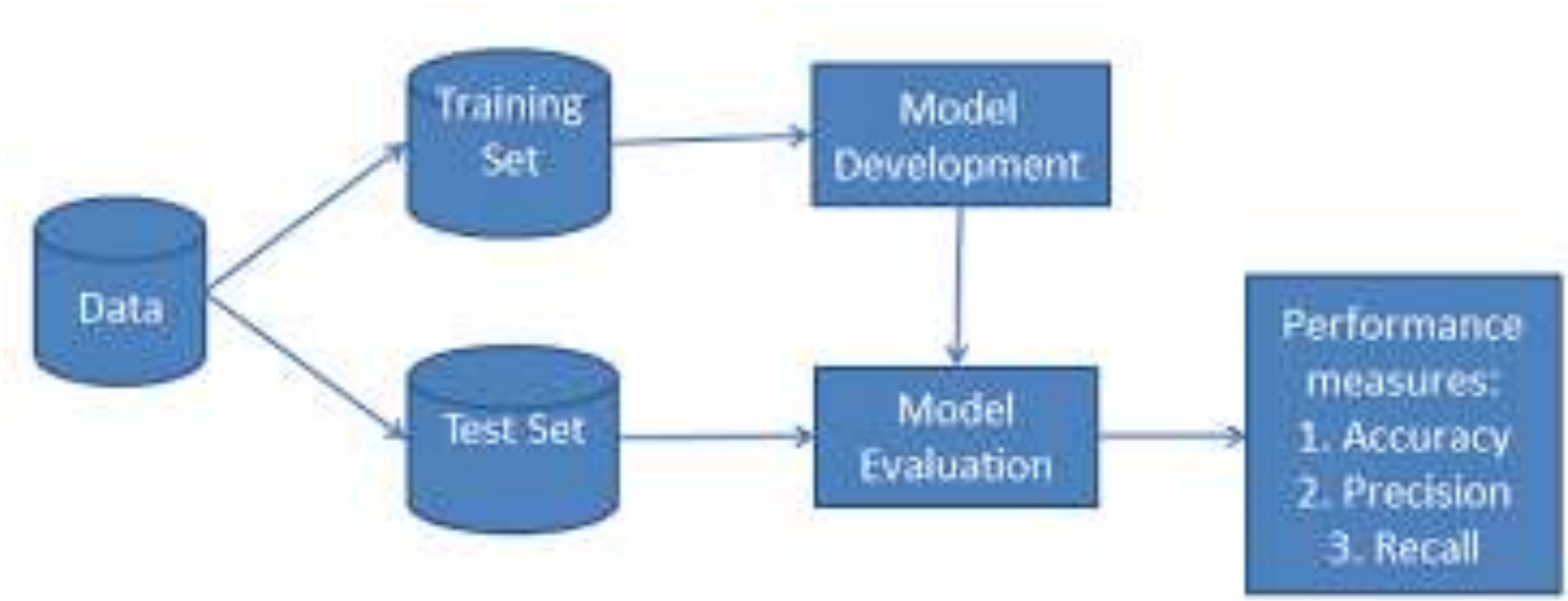
Machine Learning

Samatrix Consulting Pvt Ltd

Samatrix.io

Project – Predict Credit Card Fraud


Data
Training Set
Test Set
Model Development
Model Evaluation
Performance measures:
1. Accuracy
2. Precision
3. Recall

Samatrix.io

Project - Introduction

Project - Finance

- Predict whether a transaction is normal transaction or fraud.

Project Steps Followed

- Define Project Goals/Objective
- Data Retrieval
- Data Cleansing
- Exploratory Data Analysis
- Data Modeling
- Result Analysis

Samatrix.io

Project - Introduction

- One of the most critical issues that the finance sector faces is fraud. The fraud impacts the bottom line of a financial institution.
- It is estimated that a typical financial institution loses 5% of its revenue to fraud. If we apply this estimate to the Gross World Product of $79.6, the global loss during 2017 was $4 trillion (more than the GDP of India)
- Machine learning models can detect such Fraud.
- The machine learning models can detect anomalies in the transactions and detect cases that might be prone to fraud.
- The machine learning models can compute faster as compared to the traditional rule-based approaches.
- Machine learning models can map the data collected from various sources to the trigger points and discover the rate of defaulting or fraud propensity for each potential customer and transaction

Project - Introduction

- Define Research Goals
  - The goal of the project is to detect whether a transaction is a normal payment or a fraud.
- Data Set
  - The Data set can be downloaded
  - The dataset contains two-day transactions by European cardholders during September 2013.
  - The dataset contains 284,807 transactions out of which 492 were fraud cases
  - Due to the privacy reasons, the dataset has been anonymized. The feature names have also been changed (V1, V2, V3, etc.). Hence, you will not gain much insights from visualization
  - By the end of the project, the learners will be able to learn the approaches required for fraud modeling

Python Packages

- The first step is to load the python library for data loading, data analysis, data preparation, model evaluation, and model tuning.
- The most commonly used Python Libraries are
- **NumPy**
  - Numpy is an extensive collection of mathematical functions. It is used for data analysis of large, multidimensional arrays.
- **Pandas**
  - Pandas library is used for data manipulation and analysis. Pandas offers data structures to handle tables and the tools to manipulate them.
- **SciPy**
  - SciPy is a combination of NumPy, Pandas, and Matplotlib Libraries. SciPy is extensively used for solving mathematics, science, and engineering problems

Python Packages

- **Matplotlib**
  - Matplotlib is a plotting library that allows the creation of 2D charts and plots
- **Seaborn**
  - Seaborn is a data visualization library that is based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- **Scikit-learn (or sklearn)**
  - A machine learning library offering a wide range of algorithms and utilities.
- **StatsModels**
  - A Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests and statistical data exploration.

Main challenges involved in credit card fraud detection are:

- Enormous Data is processed every day and the model build must be fast enough to respond to the scam in time.
- Imbalanced Data i.e most of the transactions (99.8%) are not fraudulent which makes it really hard for detecting the fraudulent ones
- Data availability as the data is mostly private.
- Misclassified Data can be another major issue, as not every fraudulent transaction is caught and reported.
- Adaptive techniques used against the model by the scammers.

How to tackle these challenges?

- The model used must be simple and fast enough to detect the anomaly and classify it as a fraudulent transaction as quickly as possible.
- Imbalance can be dealt with by properly using some methods which we will talk about in the next paragraph
- For protecting the privacy of the user the dimensionality of the data can be reduced.
- A more trustworthy source must be taken which double-check the data, at least for training the model.
- We can make the model simple and interpretable so that when the scammer adapts to it with just some tweaks we can have a new model up and running to deploy.

Import Libraries and Load the Data

Import the Libraries

```
import numpy as np
import pandas as pd
```

Load the data

```
Fraud_data = pd.read_csv('../creditcard.csv.zip')
```

Understanding Data

- The most important step of model development is understanding the dataset. Generally, we follow the following steps to understand the data:
  - View the raw data
  - Dimensions of the dataset
  - Data Types of the attributes
  - Presence of Null Values in the dataset
  - Statistical Analysis

View the raw data

```
fraud_data.head()
```

```
Time      V1     V2     V3     V4   ...    V26          V27    V28   Amount
Class
0    0.0 -1.36 -0.07  2.54  1.38  ... -0.19  1.34e-01 -0.02  149.62
0
1    0.0  1.19  0.27  0.17  0.45  ...  0.13 -8.98e-03  0.01    2.69
0
2    1.0 -1.36 -1.34  1.77  0.38  ... -0.14 -5.54e-02 -0.06  378.66
0
3    1.0 -0.97 -0.19  1.79 -0.86  ... -0.22  6.27e-02  0.06  123.50
0
4    2.0 -1.16  0.88  1.55  0.40  ...  0.50  2.19e-01  0.22   69.99
0
```

Dimension of the Data

```
fraud_data.shape
```

(153758, 31)

We get the dimension of the dataset.

Data Type

fraud_data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 153758 entries, 0 to 153757
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    153758 non-null  int64
 1   V1      153758 non-null  float64
 2   V2      153758 non-null  float64
 3   V3      153758 non-null  float64
 4   V4      153758 non-null  float64
 5   V5      153758 non-null  float64
 29  Amount  153758 non-null  float64
 30  Class   153758 non-null  int64
dtypes: float64(29), int64(2)
memory usage: 36.4 MB
```

Samatrix.io

Data Type

- There are 30 rows. We have removed some of them so that the output can be displayed on the slide
- Our observations are as follows
  - NaN values do not present in the data set. Because of the Non-Null Count and number of rows in the dataset match.
  - There are 29 Input Variables and 1 Output Variable (Class)
  - The data type of all the input variables is float64 whereas the data type of out variable (Class) is int64

Null Values

```
fraud_data.isnull().sum()

Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V24       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

- The dataset does not contain any null values

Exploratory Data Analysis

Samatrix.io

Statistical Data Analysis

`DataFrame.describe()` is used to get the descriptive statistics.

The descriptive statistics summarize the count of values in each column of the data set.

We get the mean(), standard deviation, and interquartile ranges while excluding NaN values.

However, the `describe()` method deals only with numeric values, not with any categorical values.

The `describe()` method ignores the categorical values in a column and displays a summary for the other columns.

To display the categorical values, we need to pass the parameter `include="all"`

Data Analysis

```
fraud_data.describe()

Time          V1         V2   ...        V28     Amount     Class
count  284807.00  2.85e+05  2.85e+05  ...  2.85e+05  284807.00  2.85e+05
mean    94813.86  3.92e-15  5.69e-16  ... -1.21e-16      88.35  1.73e-03
std     47488.15  1.96e+00  1.65e+00  ...  3.30e-01     250.12  4.15e-02
min         0.00 -5.64e+01 -7.27e+01  ... -1.54e+01       0.00  0.00e+00
25%     54201.50 -9.20e-01 -5.99e-01  ... -5.30e-02       5.60  0.00e+00
50%     84692.00  1.81e-02  6.55e-02  ...  1.12e-02      22.00  0.00e+00
75%    139320.50  1.32e+00  8.04e-01  ...  7.83e-02      77.16  0.00e+00
max    172792.00  2.45e+00  2.21e+01  ...  3.38e+01   25691.16  1.00e+00

[8 rows x 31 columns]
```

We can see that the data for the variables from V1 to V28 is already scaled and cleaned. So there is no need for a data cleaning process in this case

Samatrix.io

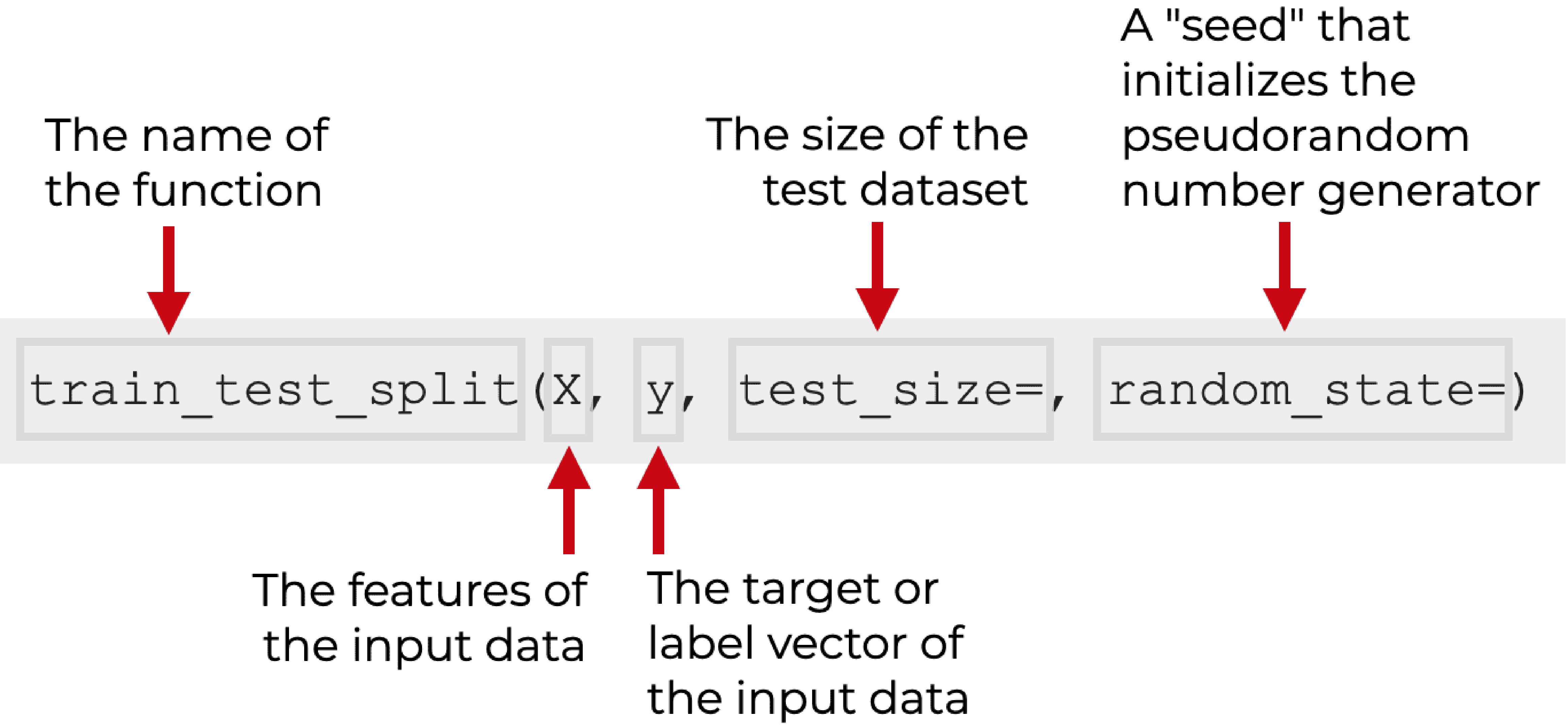
Response Variable Analysis

```
class_names = {0:'Not Fraud', 1:'Fraud'}
print(fraud_data.Class.value_counts().rename(index = class_names))

Not Fraud    153428
Fraud           330
Name: Class, dtype: int64
```

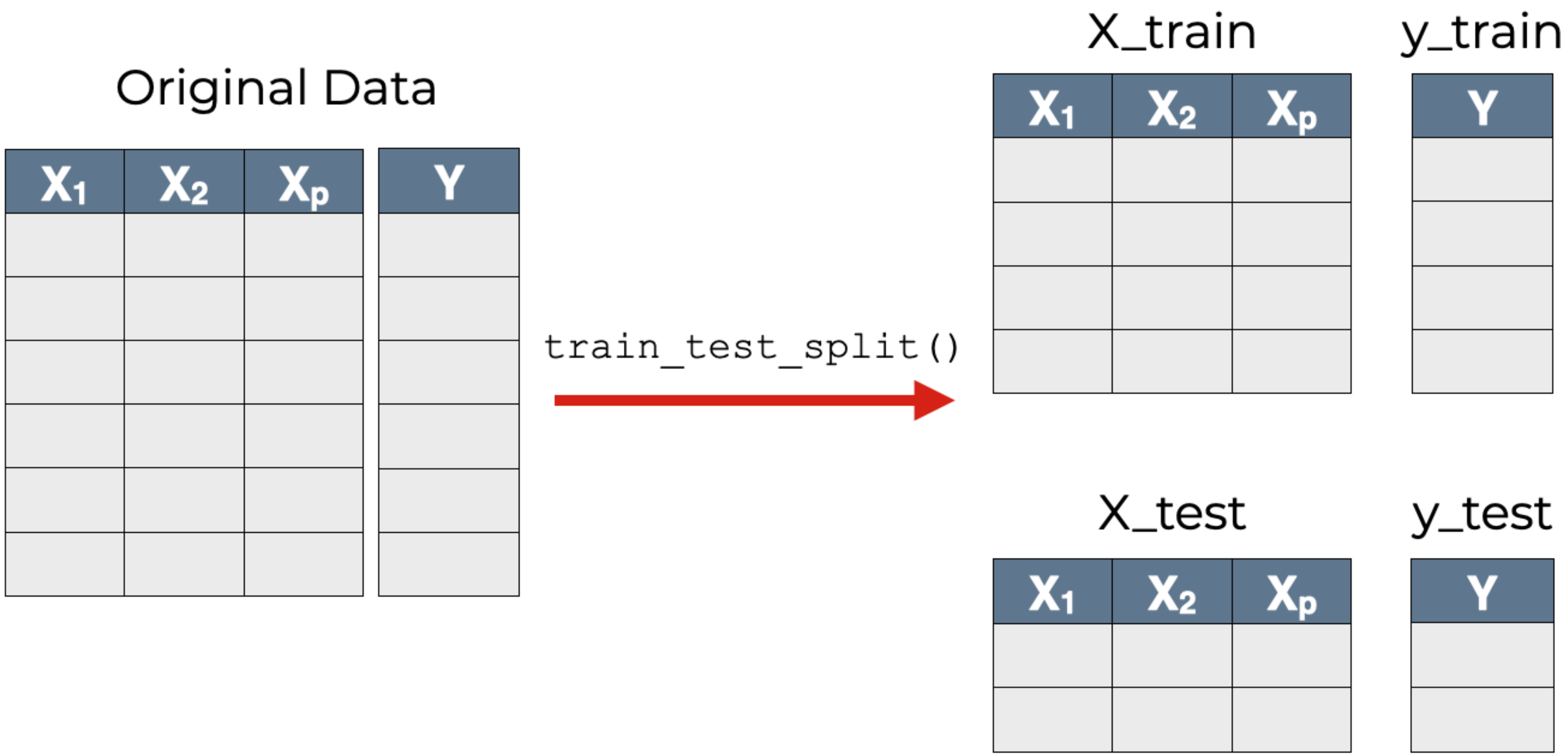
- You may notice the imbalance in the data labels.
- The majority of the transactions are nonfraud.
- Due to the imbalance, most models will not place the required emphasis on the fraud signals and the model will assume all the transactions to be nonfraud which would be an unacceptable result.
- We shall however learn to handle such issues in subsequent case studies

Data Visualization and Data Cleaning

- Data Visualization
  - We can draw the scatterplot matrix and heatmap. Since the variable description is not given in this case, we will not gain any useful insights from the plot. Hence we can skip the step
- Data Cleaning
  - This data is already in a cleaned format without any empty rows or columns. Data cleaning or categorization is not required in this case


A "seed" that initializes the pseudorandom number generator
The name of the function
The size of the test dataset
train_test_split(X, y, test_size=, random_state=)
The features of the input data
The target or label vector of the input data

TRAIN_TEST_SPLIT SPLITS DATA INTO TRAINING DATA AND TEST DATA



Train Test Split

- Before fitting the data into the machine learning model, we should split the data into training data and testing data.
- This is an important step because we would like to train the model by fitting the training data. But to test the data, we should use the data that is new to the model.
- Then only we would be able to calculate the performance of the model on the unseen data.
- We use **`sklearn.model_selection.train_test_split()`** method for Train Test Split.
- The first parameter of the train_test_split is **`test_size`** which specifies the ratio of data in the train dataset and test dataset.
- The value 1/3 will put one-third values in the test data set and two-thirds values in the training data set.

Samatrix.io

Train Test Split

- The second parameter is `random_state`.
- Before splitting the data into training and test datasets, the data is randomly shuffled.
- By giving a value for the random state we ensure, the data is shuffled in a similar way every time so that you get the consistent training and test dataset.
- The third parameter is **stratify**.
- Stratify parameter ensures that the proportion of values in the training and test data set will be the same as the proportion of values in the master dataset.
- For example, if variable y is a binary categorical variable with values 0 and 1. Suppose there are 25% of zeros and 75% of ones, `stratify=y` will make sure that your random split has 25% of 0's and 75% of 1's.
- Before splitting the data in Training and Test Dataset, we need to split the data in X and y variables.
- The variable "Class" is output variable "y" and rest of the variables are input variables "X"

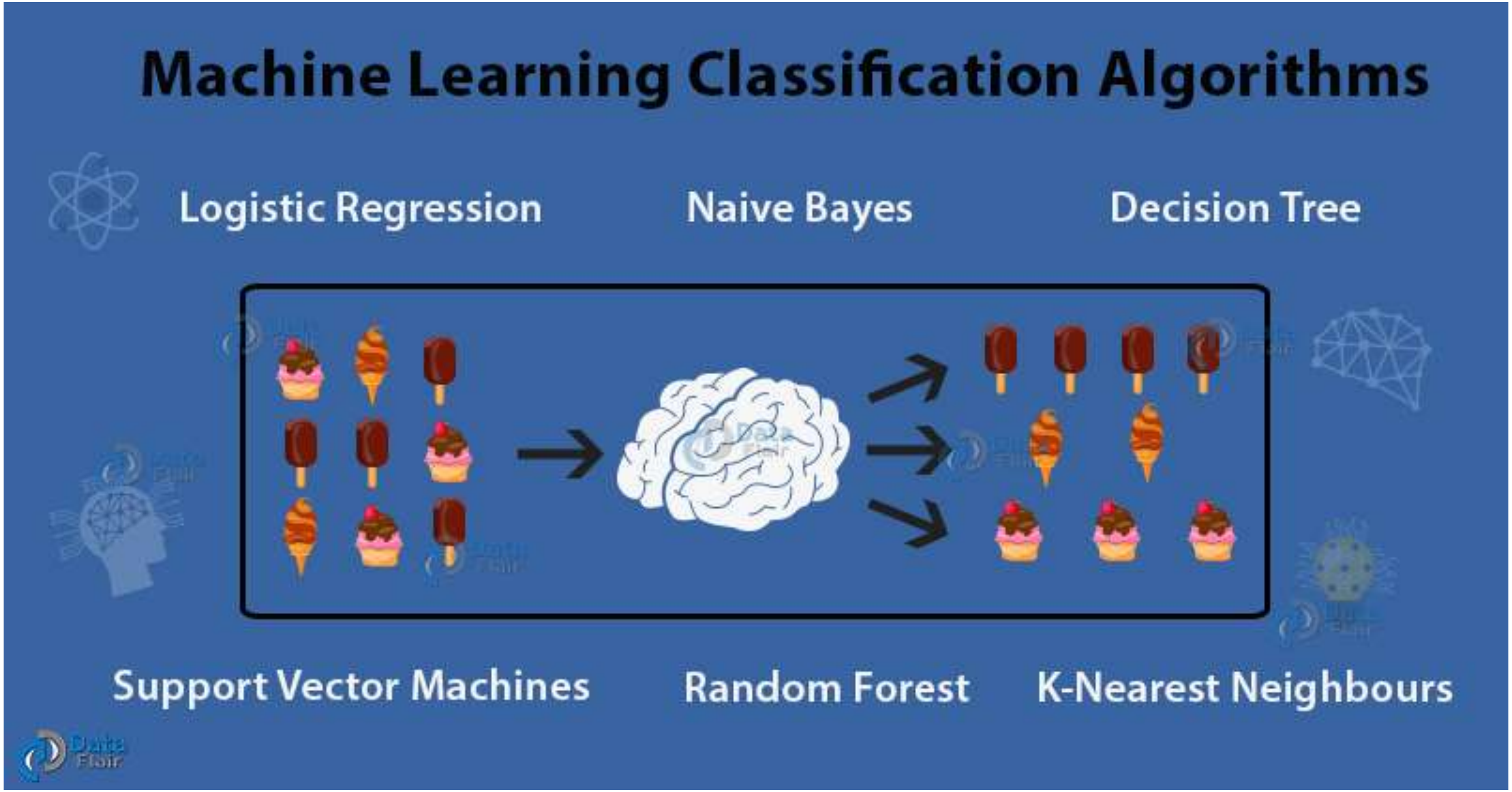
Train Test Split

```
#importing the library
from sklearn.model_selection import train_test_split

# output
y= fraud_data["Class"]

#input
X = fraud_data.loc[:, fraud_data.columns != 'Class']

X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=1/3,
random_state=42, stratify=y)
```

Data Modeling

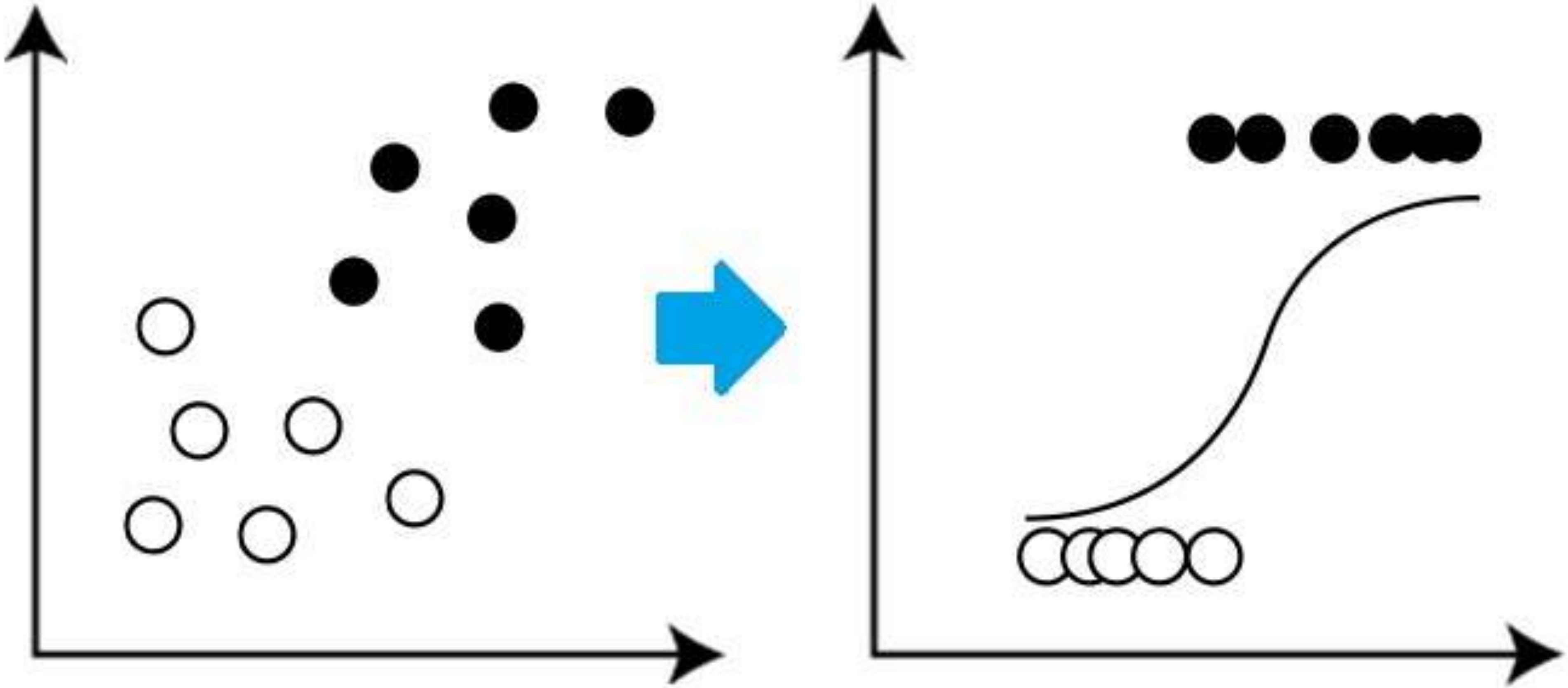
Data Modeling

- In this step, we will evaluate different machine learning models
- We will use Linear as well as Non-Linear Algorithms for this evaluation


Machine Learning Classification Algorithms
Logistic Regression
Naive Bayes
Decision Tree
Support Vector Machines
Random Forest
K-Nearest Neighbours

Samatrix.io

LOGISTIC REGRESSION



Samatrix.io

Logistic Regression

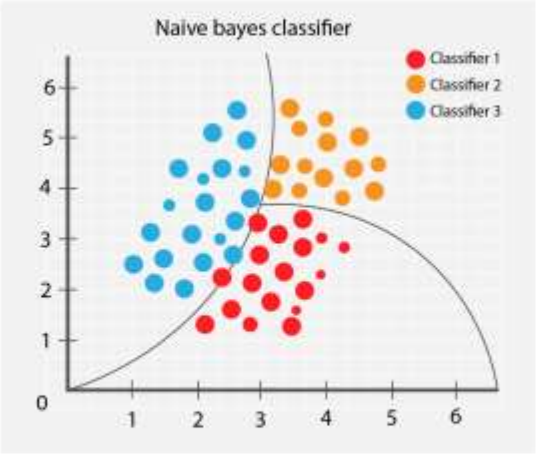
```
#Import Library for Accuracy Score
from sklearn.metrics import accuracy_score

#Import Library for Logistic Regression
from sklearn.linear_model import LogisticRegression
#Initialize the Logistic Regression Classifier
logisreg = LogisticRegression()
#Train the model using Training Dataset
logisreg.fit(X_train, y_train)
# Prediction using test data
y_pred = logisreg.predict(X_test)
# Calculate Model accuracy by comparing y_test and y_pred
acc_logisreg = round( accuracy_score(y_test, y_pred) * 100, 2 )
print( 'Accuracy of Logistic Regression model : ', acc_logisreg )

Accuracy of Logistic Regression model :  99.91
```

Samatrix.io

Naive Bayes

In machine learning, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$



using Bayesian probability terminology, the above equation can be written as

$$\text{Posterior} = \frac{\text{prior x likelihood}}{\text{evidence}}$$

Classification process

New data = (X) = $(X_1, X_2, \ldots, X_m)$

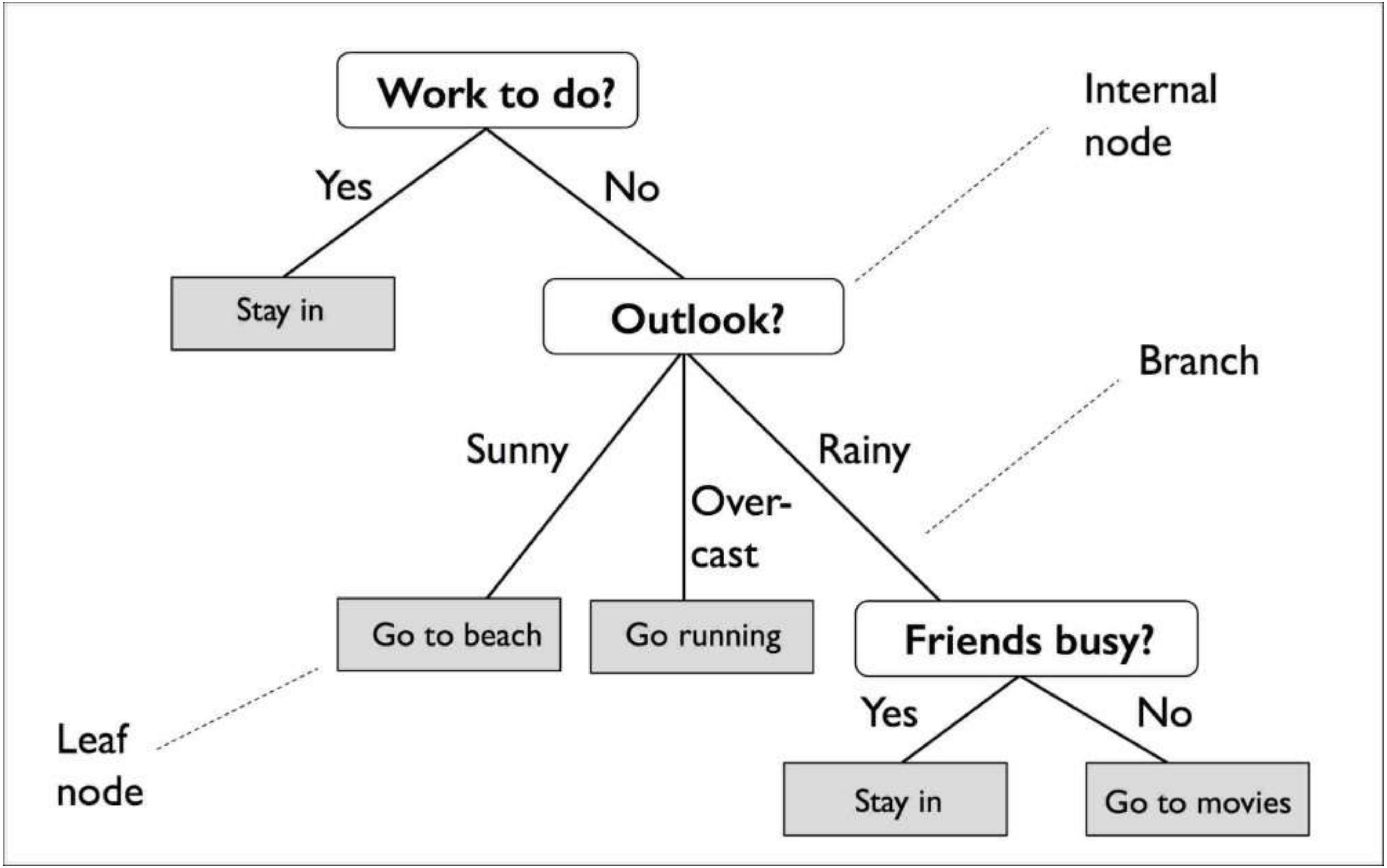
Class C is a member of $\{C_1, C_2, \ldots, C_k\}$

Gaussian Naïve Bayes

```
#Import Library for Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
#Initialize the Gaussian Naive Bayes Classifier
model = GaussianNB()
#Train the model using Training Dataset
model.fit(X_train, y_train)
# Prediction using test data
y_pred = model.predict(X_test)
# Calculate Model accuracy by comparing y_test and y_pred
acc_ganb = round( accuracy_score(y_test, y_pred) * 100, 2 )
print( 'Accuracy of Gaussian Naive Bayes : ', acc_ganb )
```

```
Accuracy of Gaussian Naive Bayes :  99.28
```

Samatrix.io

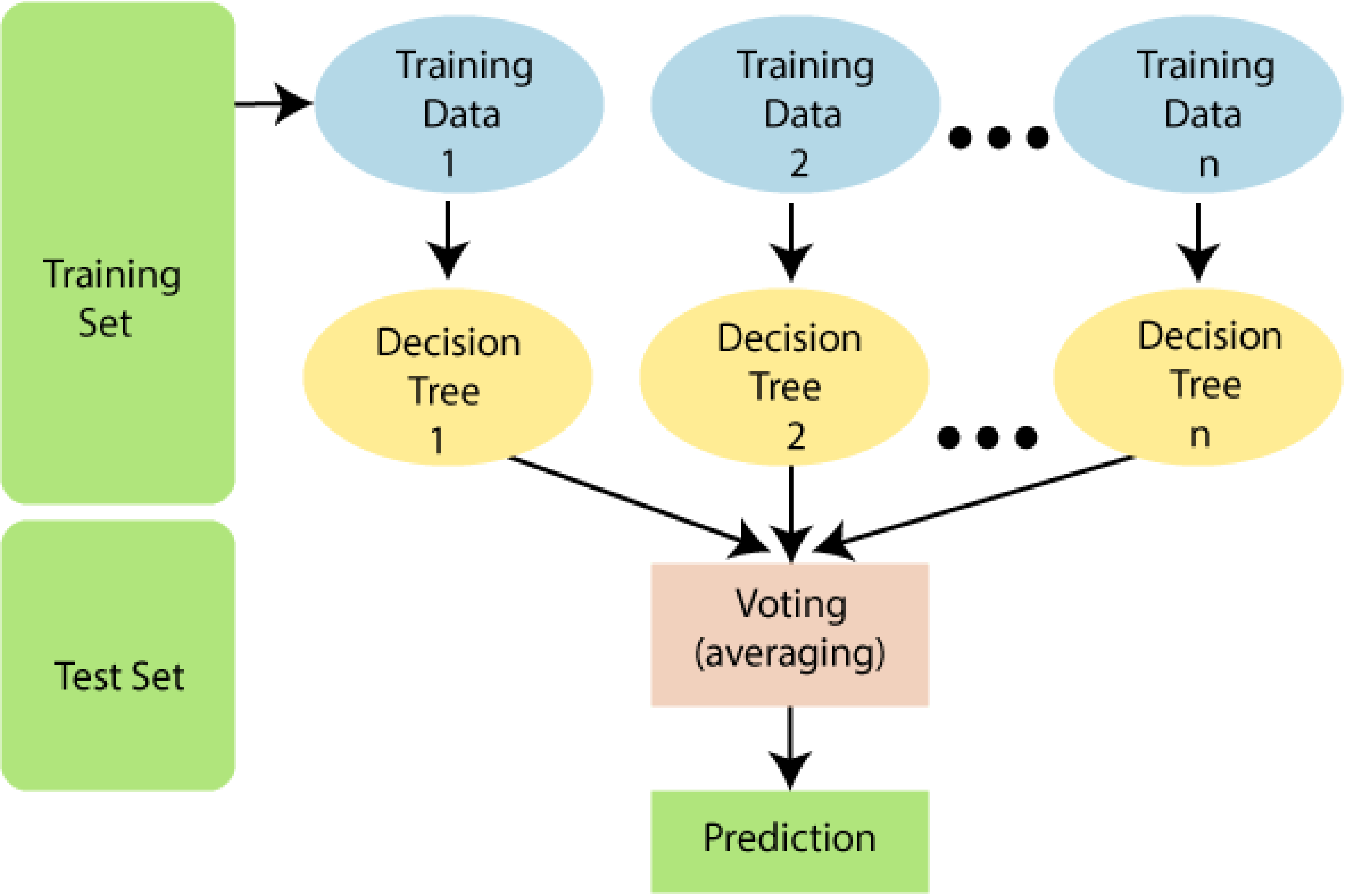

Work to do?
Yes
No
Stay in
Outlook?
Internal node
Branch
Sunny
Over-cast
Rainy
Go to beach
Go running
Friends busy?
Yes
No
Leaf node
Stay in
Go to movies

Samatrix.io

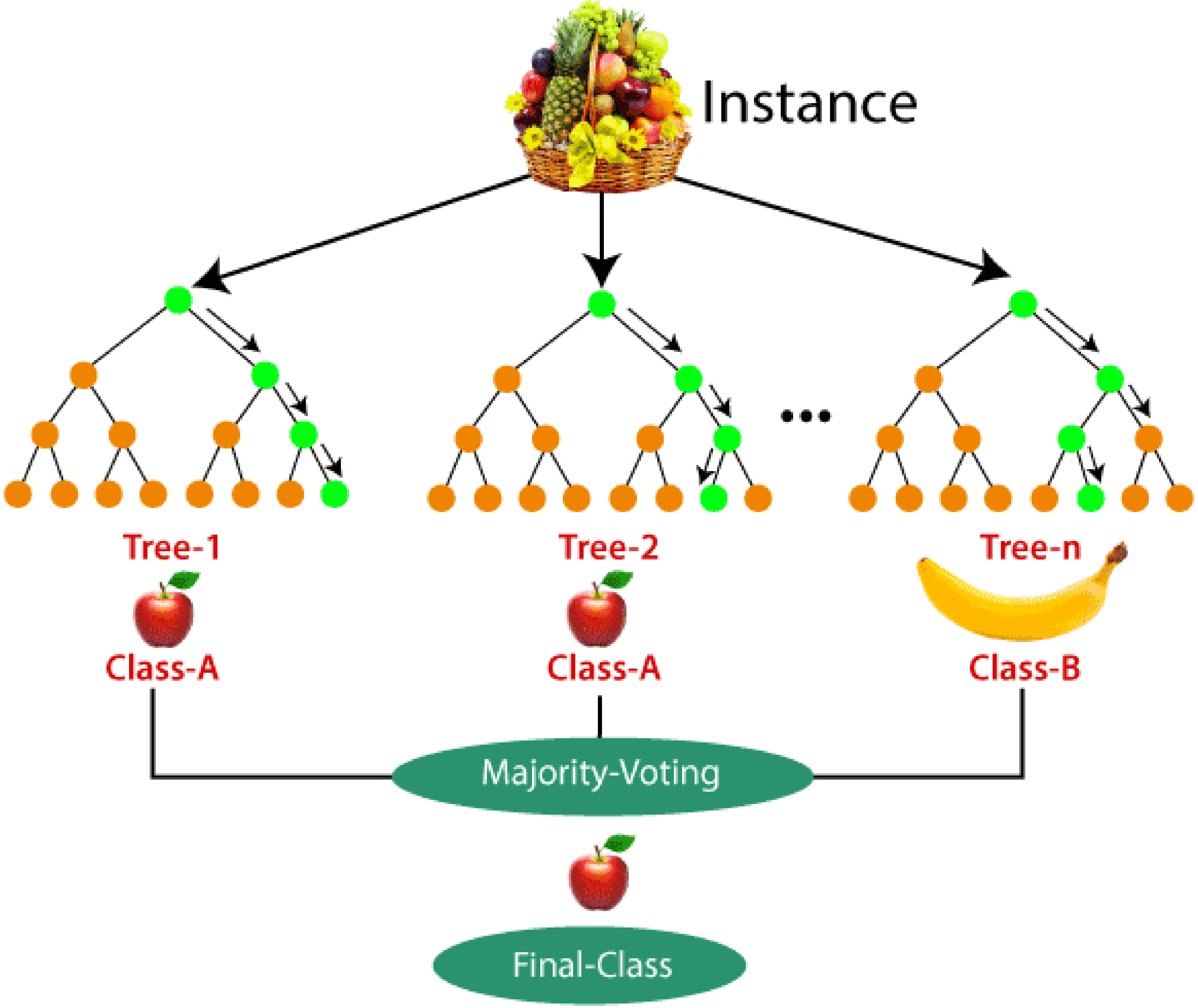
Decision Tree (CART)

```
#Import Library for Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
#Initialize the Decision Tree Classifier
model = DecisionTreeClassifier()
#Train the model using Training Dataset
model.fit(X_train, y_train)
# Prediction using test data
y_pred = model.predict(X_test)
# Calculate Model accuracy by comparing y_test and y_pred
acc_dtree = round( accuracy_score(y_test, y_pred) * 100, 2 )
print( 'Accuracy of  Decision Tree Classifier : ', acc_dtree )

Accuracy of  Decision Tree Classifier :  99.91
```

Samatrix.io


Training Set
Test Set
Training Data 1
Training Data 2
Training Data n
Decision Tree 1
Decision Tree 2
Decision Tree n
Voting (averaging)
Prediction

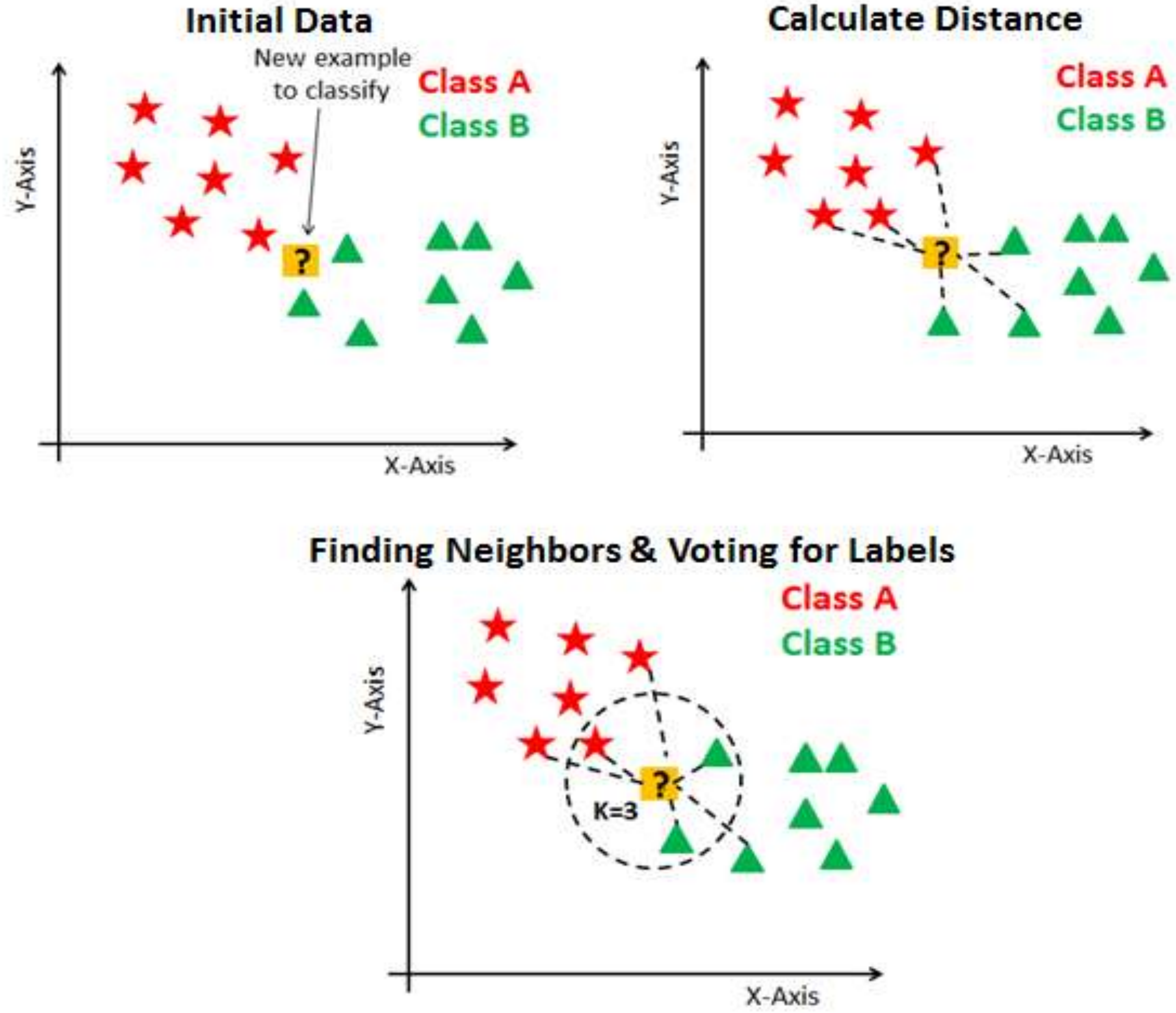
Samatrix.io


Instance
Tree-1
Tree-2
Tree-n
Class-A
Class-A
Class-B
Majority-Voting
Final-Class

Samatrix.io

Random Forest

```
#Import Library for Random Forest
from sklearn.ensemble import RandomForestClassifier
#Initialize the Random Forest
model = RandomForestClassifier()
#Train the model using Training Dataset
model.fit(X_train, y_train)
# Prediction using test data
y_pred = model.predict(X_test)
# Calculate Model accuracy by comparing y_test and y_pred
acc_rf = round( accuracy_score(y_test, y_pred) * 100, 2 )
print( 'Accuracy of  Random Forest : ', acc_rf )

Accuracy of  Random Forest :   99.96
```



Initial Data
New example to classify
Class A
Class B
Y-Axis
X-Axis
Calculate Distance
Class A
Class B
Y-Axis
X-Axis
Finding Neighbors & Voting for Labels
Class A
Class B
K=3
Y-Axis
X-Axis

Samatrix.io

K Nearest Neighbour Classifier

```
#Import Library for K Nearest Neighbour Model
from sklearn.neighbors import KNeighborsClassifier
#Initialize the K Nearest Neighbour Model with Default Value of K=5
model = KNeighborsClassifier()
#Train the model using Training Dataset
model.fit(X_train, y_train)
# Prediction using test data
y_pred = model.predict(X_test)
# Calculate Model accuracy by comparing y_test and y_pred
acc_knn = round( accuracy_score(y_test, y_pred) * 100, 2 )
print( 'Accuracy of KNN Classifier: ', acc_knn )

Accuracy of KNN Classifier:  99.83
```

Samatrix.io

Model Selection

Samatrix.io

Model Selection

- We have no idea which algorithms will do well on this problem.
- Let’s design our test now.
- We have used a number of models to fit Y against X.
- We will evaluate algorithms using the accuracy metric, which is one of the measures of the model performance.
- We can compare the accuracy of all the models and choose the one with the maximum accuracy

Model Selection

```
models = pd.DataFrame({
    'Model': ['Logistic Regression', 'Naive Bayes', 'Decision 
Tree', 'Random Forest', 'K - Nearest Neighbors'],
    'Score': [acc_logisreg, acc_ganb, acc_dtree, acc_rf, acc_knn]})

models.sort_values(by='Score', ascending=False)
```

Model Selection

	Model	Score
3	Random Forest	99.94
2	Decision Tree	99.91
0	Logistic Regression	99.86
4	K - Nearest Neighbors	99.80
1	Naive Bayes	98.54

Thanks

Samatrix Consulting Pvt Ltd