# **NETWORK INTRUSION DETECTION SYSTEM USING MACHINE LEARNING**

**PROBLEM STATEMENT**: Traditional intrusion detection systems (IDS) use signature-based and anomaly-based detection methods to identify and alert on malicious network traffic. It struggles to effectively identify and classify diverse and evolving cyberattacks, especially when data is imbalanced with some attack types being severely underrepresented. The traditional approach uses a predefined list of attacks to check for the state of network traffic. Methods of machine learning are utilised to find a solution to the problem of recognising new attacks or (zero-day attacks), which is a challenge that is confronted by technologically advanced businesses in today's world.

### **PROPOSED SYSTEM:**

The goal of this project is to develop a **machine learning-based Network Intrusion Detection System (NIDS)** that can perform **multiclass classification** of different attack types while effectively addressing class imbalance through **Adasyn technique.** By leveraging the **NSL-KDD dataset** and applying supervised learning models like **Random Forest, K Nearest Neighbours, Decision Trees, Naïve bayes** and **Xtreme Gradient Boosting**,the system aims to achieve **improved reliability** in detecting both frequent and rare cyberattacks.

## **INTRODUCTION:**

Intrusion refers to any series of actions aimed at undermining the CIA (confidentiality, integrity, or availability) of resources. Integrity entails safeguarding data from unauthorized alterations, and ensuring its integrity is preserved. Confidentiality aims to prevent unauthorized access to sensitive information, making certain that only authorized individuals have access. Availability is focused on ensuring that information is consistently accessible to those with authorized access. An intrusion detection system (IDS) is a software or instrument designed to detect malicious actions or breaches of policy within a network.

In today's hyper-connected world, cyber threats have become increasingly sophisticated, making robust network security a critical necessity. Traditional intrusion detection systems often struggle to keep up with evolving attack patterns, resulting in vulnerabilities that can lead to data breaches, financial loss, or system compromise.

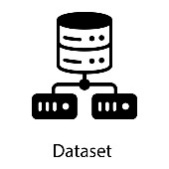
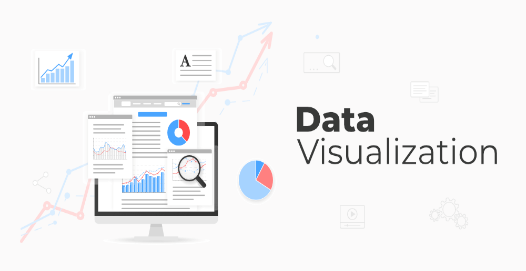
To address this, our project focuses on building an intelligent **Network Intrusion Detection System (NIDS)** powered by **machine learning** techniques. We utilize the **NSL-KDD dataset**, a refined benchmark dataset widely used for evaluating intrusion detection systems, offering a balanced mix of normal and attack traffic types.

We explore multiple supervised learning algorithms, like:

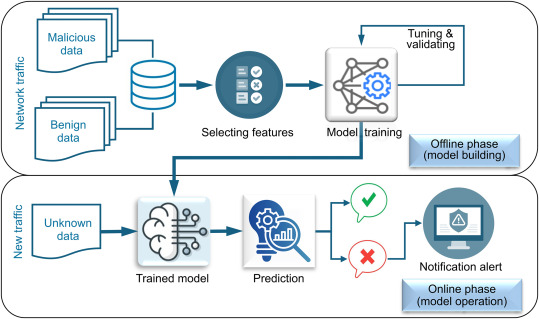
* **Random Forest (RF)**: For its high accuracy and interpretability,
* **XGBoost**: For its speed and superior handling of complex patterns.

Our final system demonstrates robust detection across multiple intrusion categories with improved accuracy—highlighting the power of machine learning in securing modern networks against cyber threats.

## **METHODOLOGY**

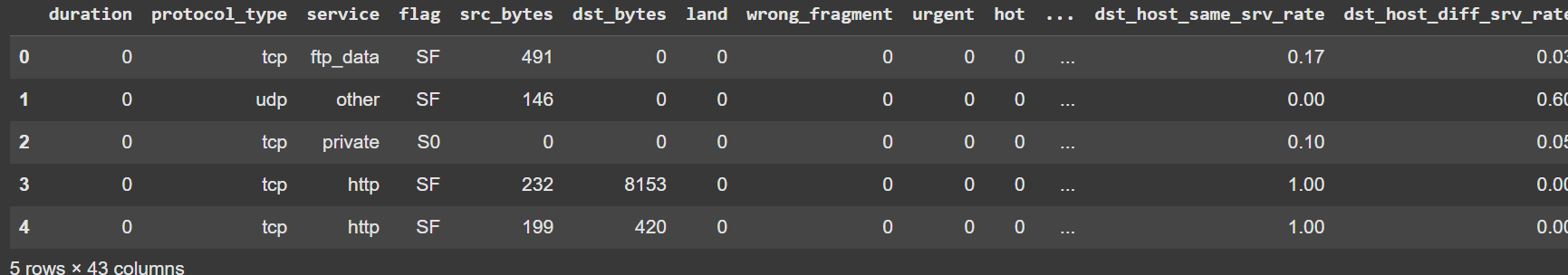
  

**NSL-KDD Dataset Performing EDA Visualising the data**



**Training and Testing**

**1.DATA COLLECTION:** For the data, we have used the NSL-KDD Dataset, which is accessible as a public data set for the network intrusion detection system. The fact that the KDD dataset contains a significant number of records that are identical to others is, to begin, one of its primary drawbacks. 78% of the records in the training set have been duplicated, and approximately 75% of the total number of records in the testing dataset has been duplicated; as a result, our findings translate to biassed learning methods. Even if there may potentially be a new version of the KDD Cup 99 dataset, this dataset is used in the NSL KDD dataset. There is no duplication of data in the new NSL KDD Test and Train dataset, which was created by combining only the most relevant information from the original KDD dataset.

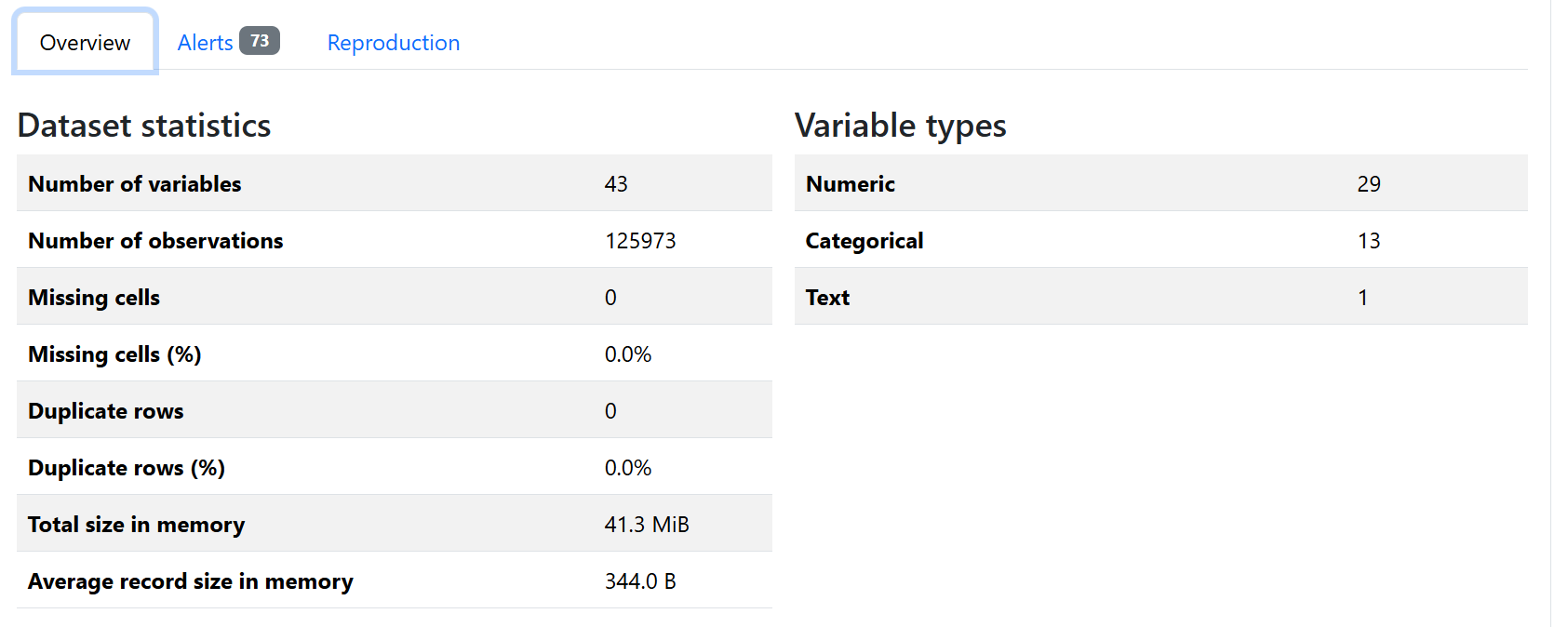
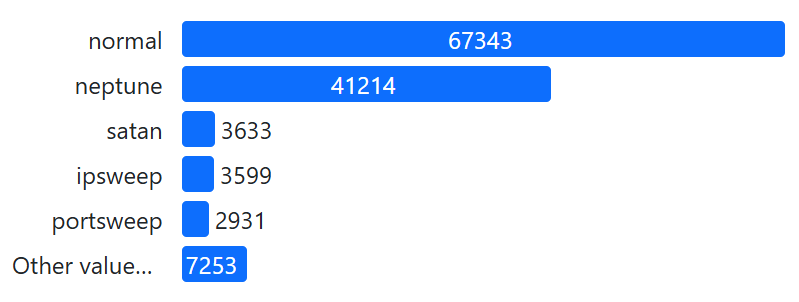


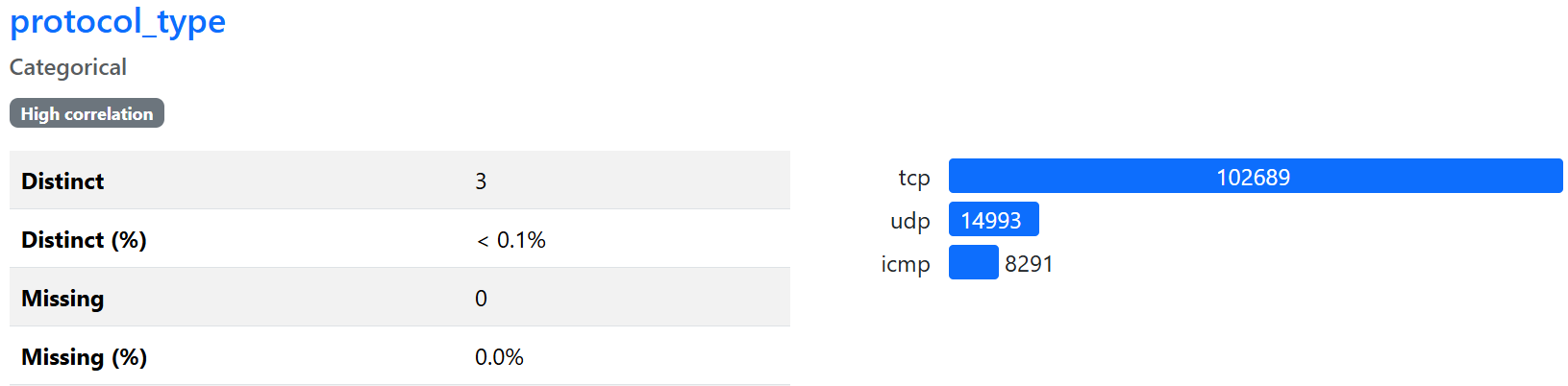
**2**. **EXPLORATORY DATA ANALYSIS:** EDA refers to understand the underlying relationships between the features or various dimensions of the dataset, understand its main features, find patterns and discover how different parts of the data are connected.

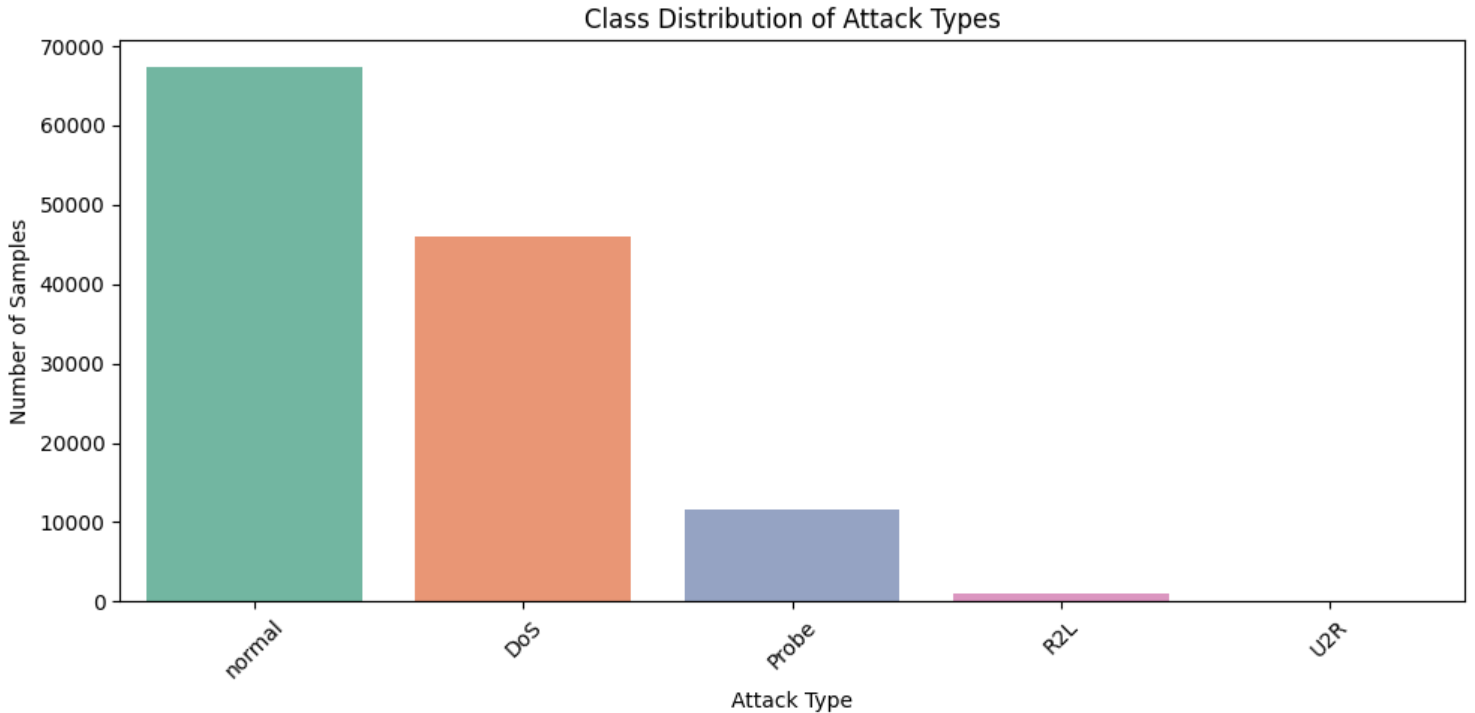
1. It helps to understand the dataset by showing how many features it has, what type of data each feature contains and how the data is distributed.
2. It helps to identify hidden patterns and relationships between different data points which help us in and model building.
3. Allows to identify errors or unusual data points (outliers) that could affect our results.
4. The insights gained from EDA help us to identify most important features for building models and guide us on how to prepare them for better performance.
5. By understanding the data it helps us in choosing best modelling techniques and adjusting them for better results.

We have used **ydata profiler and Pandas library** to gain various insights from the dataset and detect all the significant patterns. ydata-profiling is a leading package for data profiling, that automates and standardizes the generation of detailed reports, complete with statistics and visualizations. The significance of the package lies in how it streamlines the process of understanding and preparing data for analysis in a single line of code.

The results extracted from the same are shown below:





**Class Imbalance highlighted by data visualization**

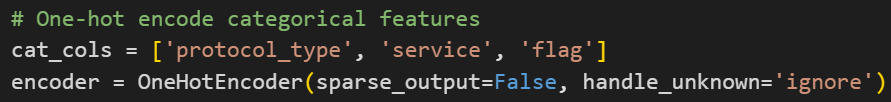


**Sample of alerts for training dataset (ydata profiling)**

**3. FEATURE ENGINEERING:** Feature engineering is the process of selecting, manipulating, and transforming raw data into features that can be used to train machine learning models. A feature is any measurable input variable that a predictive model uses to make decisions, such as age, income, or transaction type. The main objective of feature engineering is to improve model performance by ensuring that the data fed into the model is both relevant and in a format that the algorithm can effectively use.

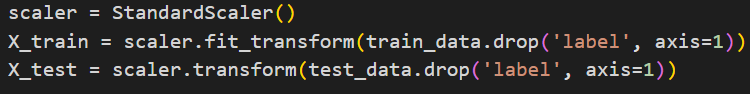
**1. Categorical Feature Encoding**

Categorical columns such as protocol\_type, service, and flag are converted into numerical format using **one-hot encoding**. This process creates binary columns for each category, enabling machine learning algorithms to interpret categorical data effectively.



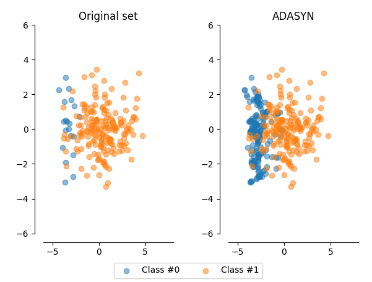
**2. Feature Scaling**

Numerical features are normalized using StandardScaler. This standardization ensures that all features have a mean of zero and a standard deviation of one, preventing features with larger scales from dominating the learning process and improving model convergence.



**3. Handling class Imbalance**

**ADASYN** is used to balance the dataset by generating synthetic samples for minority classes. This prevents the model from being biased towards the majority class and improves its ability to detect rare events. Imbalanced datasets are common in real-world applications, such as medical research, network intrusion detection, and fraud detection in credit card transactions. These datasets have a majority class with many samples and minority classes with few samples, causing machine learning algorithms to be biased towards the majority class.



**4. Train/Test Feature Alignment**

After encoding, the training and testing datasets are aligned to ensure they contain the same features in the same order. Missing columns in either set are filled with zeros, maintaining consistency and preventing errors during model evaluation.



**5. Label Mapping**

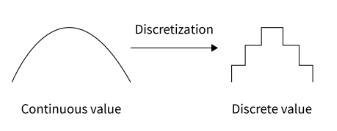
For Multiclass classification, all attack types are mapped to their respective type of attack label, while normal traffic is labelled as 'normal'. This simplifies the classification task and addresses issues related to label imbalance and class diversity.

**6.Feature Selection**

Feature Selection means selecting only the most important features from your dataset that contribute significantly to model performance. It removes irrelevant, redundant, or noisy features and also simplifies the model and often improves accuracy, especially when you have high-dimensional data like in NSL-KDD. **SelectKBest** is a **filter-based feature selection method** from sklearn. It ranks all features based on a scoring function that measures how strongly each feature is related to the target variable then it selects the top K best features according to their scores. For classification problems like ours, commonly used scoring functions include mutual\_info\_classif, which captures nonlinear dependencies between features and the target, and chi2, which works well for categorical data. Each feature receives a score reflecting its predictive power, and then the top K features with the highest scores are selected. In this project, SelectKBest was applied after discretization and scaling of the features. Out of the original 41 features, the 13 most informative features were selected based on their scores. This helped reduce dimensionality, eliminate less relevant information, and allowed the machine learning models to focus on the most significant indicators for detecting different types of network intrusions.

**7.Discretization**

Real-world data tend to be noisy. Noisy data is data with a large amount of additional meaningless information in it called noise. Data cleaning (or data cleansing) routines attempt to smooth out noise while identifying outliers in the data. There are three main techniques for smoothing out data: Binning, regression and Outlier detection. Here, we are concerned with the Binning method for data smoothing. In this method the data is first sorted and then the sorted values are distributed into a number of *buckets*or *bins*. As binning methods consult the neighbourhood of values, they perform local smoothing. Discretization refers to the process of converting or partitioning continuous attributes, features or variables to discretized or nominal attributes/features/variables/intervals.



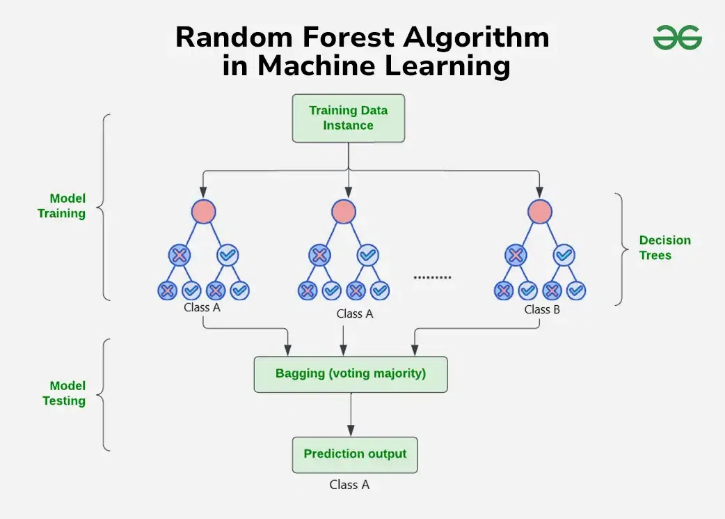
In this project, KBinsDiscretizer was applied after scaling the features, using 5 bins with a uniform strategy. This discretization helped simplify numeric features, reduce the influence of outliers, and made the data more suitable for certain machine learning algorithms that perform better with categorical or discretized data. Although tree-based models like Random Forest and XGBoost handle continuous variables well, discretization contributed to better performance for algorithms sensitive to feature distributions and scales, especially when combined with feature selection.

**4. MODEL TRAINING:** Training a machine learning model on the NSL-KDD dataset involves several systematic steps to ensure the model can effectively learn to distinguish between normal and attack network traffic. Various machine learning algorithms can be applied, such as Random Forest, Naïve Bayes, Decision Trees, KNN and XGBoost. The choice of model depends on the desired balance between accuracy, interpretability, and computational efficiency. We have used Random Forest, Decision Trees, K nearest neighbours, Naïve bayes and Xtreme Gradient Boosting in this project.

1. **Random Forest:** Random forest is a commonly-used machine learning algorithm, trademarked by Leo Breiman and Adele Cutler, that combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems. The model is made up of multiple Decision trees which are its base learners.
2. **Decision Trees:** We have mainly two types of decision tree based on the nature of the target variable: classification trees and regression trees.

* Classification trees: They are designed to predict categorical outcomes means they classify data into different classes. They can determine whether an email is "spam" or "not spam" based on various features of the email.
* Regression trees: These are used when the target variable is continuous It predict numerical values rather than categories. For example a regression tree can estimate the price of a house based on its size, location, and other features. Random forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, there’s no need to combine a decision tree with a bagging classifier because you can easily use the classifier-class of random forest. With random forest, you can also [deal with regression tasks](https://builtin.com/data-science/random-forest-python) by using the algorithm’s regressor. Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting [a node](https://builtin.com/software-engineering-perspectives/tree-traversal), it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Therefore, in a random forest classifier, only a random subset of the features is taken into consideration by the algorithm for splitting a node. You can even make trees more random by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does).



1. **XG Boost:** Traditional machine learning models like decision trees and random forests are easy to interpret but often struggle with accuracy on complex datasets. XGBoost short form for eXtreme Gradient Boosting is an advanced machine learning algorithm designed for efficiency, speed and high performance. It is an optimized implementation of [**Gradient Boosting**](https://www.geeksforgeeks.org/ml-gradient-boosting/) and is a type of [**ensemble learning**](https://www.geeksforgeeks.org/a-comprehensive-guide-to-ensemble-learning/) method that combines multiple weak models to form a stronger model.

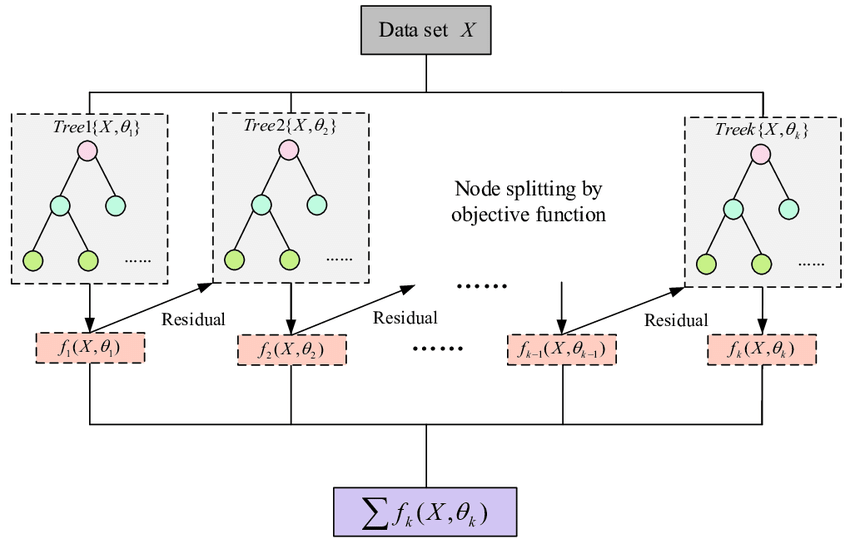
XGBoost uses [**decision trees**](https://www.geeksforgeeks.org/decision-tree/)as its base learners and combines them sequentially to improve the model’s performance. Each new tree is trained to correct the errors made by the previous tree and this process is called [boosting](https://www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/).

It has built-in parallel processing to train models on large datasets quickly. XGBoost also supports customizations allowing users to adjust model parameters to optimize performance based on the specific problem.

**How XGBoost Works?**

It builds decision trees sequentially with each tree attempting to correct the mistakes made by the previous one. The process can be broken down as follows:

1. **Start with a base learner**: The first model decision tree is trained on the data. In regression tasks this base model simply predicts the average of the target variable.
2. **Calculate the errors**: After training the first tree the errors between the predicted and actual values are calculated.
3. **Train the next tree**: The next tree is trained on the errors of the previous tree. This step attempts to correct the errors made by the first tree.
4. **Repeat the process**: This process continues with each new tree trying to correct the errors of the previous trees until a stopping criterion is met.
5. **Combine the predictions**: The final prediction is the sum of the predictions from all the trees.

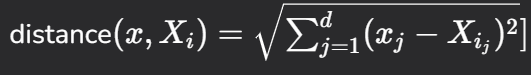


1. **K-Nearest Neighbours:** K-Nearest Neighbours (KNN) is a supervised machine learning algorithm generally used for classification but can also be used for regression tasks. It works by finding the "k" closest data points (neighbours) to a given input and makes a prediction based on the majority class (for classification) or the average value (for regression). Since KNN makes no assumptions about the underlying data distribution it makes it a non-parametric and instance-based learning method.

In the k-Nearest Neighbours algorithm k is just a number that tells the algorithm how many nearby points or neighbours to look at when it makes a decision.

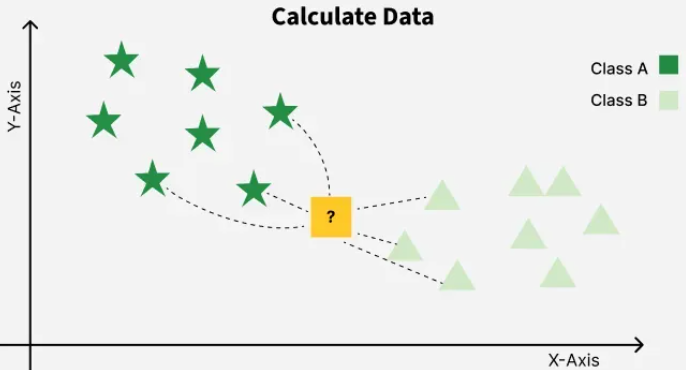
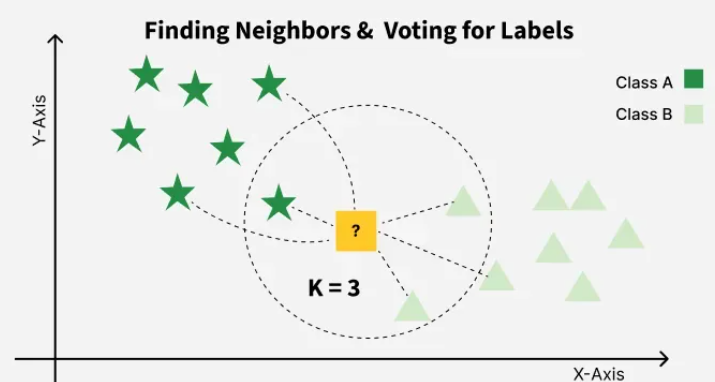
* The value of k in KNN decides how many neighbours the algorithm looks at when making a prediction.
* Choosing the right k is important for good results.
* If the data has lots of noise or outliers, using a larger k can make the predictions more stable.
* But if k is too large the model may become too simple and miss important patterns and this is called underfitting.
* So, k should be picked carefully based on the data. Some methods for selecting the optimal value of k are cross validation and elbow method.

KNN uses distance metrics to identify nearest neighbour, these neighbours are used for classification and regression task.  We use two types of distance metrics: Euclidean and Manhattan Distance.

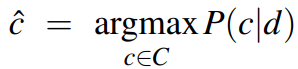
**1. Euclidean Distance:** Euclidean distance is defined as the straight-line distance between two points in a plane or space. You can think of it like the shortest path you would walk if you were to go directly from one point to another. 

**2. Manhattan Distance:** This is the total distance you would travel if you could only move along horizontal and vertical lines like a grid or city streets. It’s also called "taxicab distance" because a taxi can only drive along the grid-like streets of a city. 

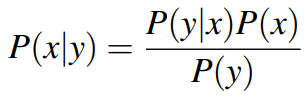
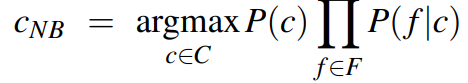
**3. Minkowski Distance:** Minkowski distance is like a family of distances, which includes both Euclidean and Manhattan distances as special cases. ****

**  **

1. **Naïve Bayes:** Naive Bayes is a probabilistic classifier, meaning that for a document d, out of all classes c ∈ C the classifier returns the class ˆc which has the maximum posterior ˆ probability given the document. we use the hat notation ˆ to mean “our argmax estimate of the correct class”, and we use argmax to mean an operation that selects the argument (in this case the class c) that maximizes a function (in this case the probability P(c|d).

****

This idea of Bayesian inference has been known since the work of Bayes (1763), Bayesian inference and was first applied to text classification by Mosteller and Wallace (1964). The intuition of Bayesian classification is to use Bayes’ rule to transform Eq. 4.1 into other probabilities that have some useful properties. Bayes’ rule is presented in Eq. 4.2; it gives us a way to break down any conditional probability P(x|y) into three other probabilities:

** **

**5.EVALUATION METRICS:** Evaluation metrics are used to measure the quality of the model. We have different evaluation metrics for a different set of machine learning algorithms. For evaluating classification models, we use classification metrics and for evaluating regression models, we use regression metrics. It can help you assess your model’s performance, monitor your ML system in production, and control your model to fit your business needs.

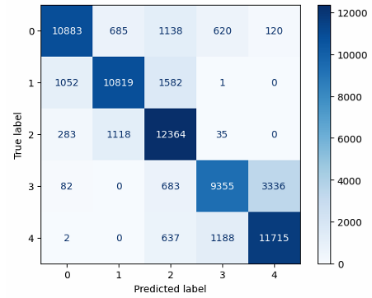
**1. Confusion Matrix:** The confusion matrix consists of four basic characteristics (numbers) that are used to define the measurement metrics of the classifier. These four numbers are:

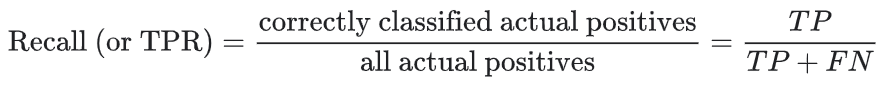
1.TP (True Positive): TP represents the number of patients who have been properly classified to have malignant nodes, meaning they have the disease.

2.TN (True Negative): TN represents the number of correctly classified patients who are healthy.

3.FP (False Positive): FP represents the number of misclassified patients with the disease but actually they are healthy. FP is also known as a *Type I error*.

4.FN (False Negative): FN represents the number of patients misclassified as healthy but actually they are suffering from the disease. FN is also known as a *Type II error*.

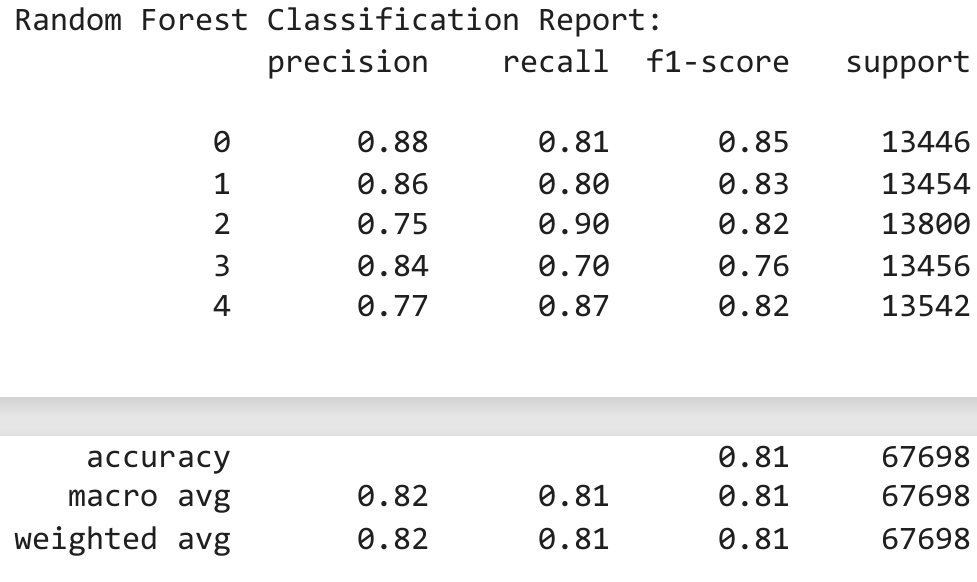
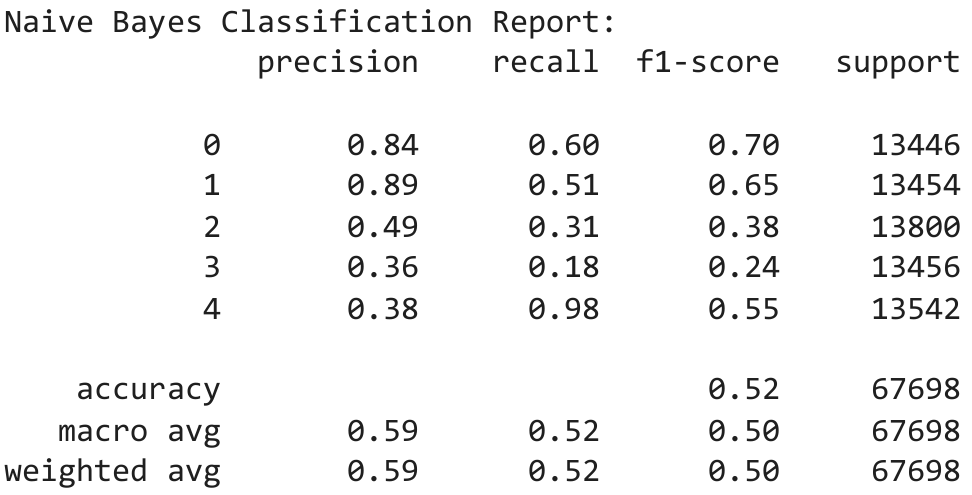


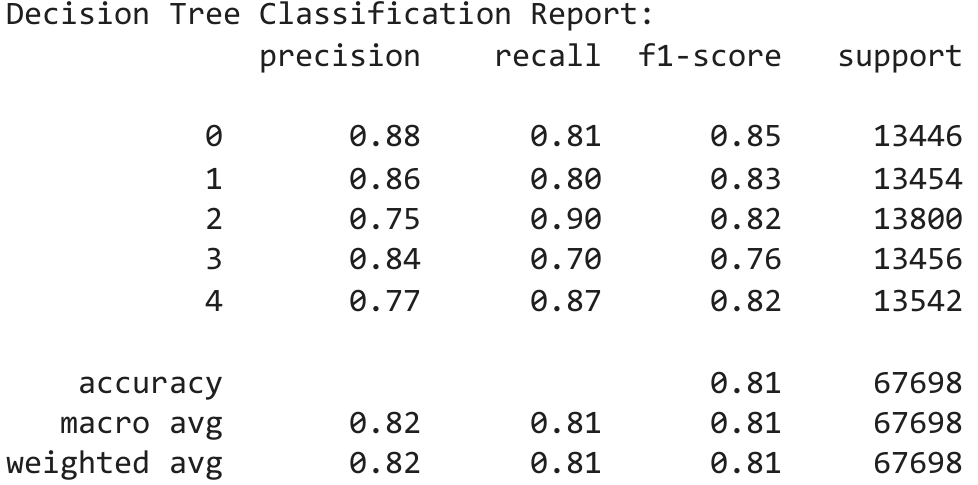
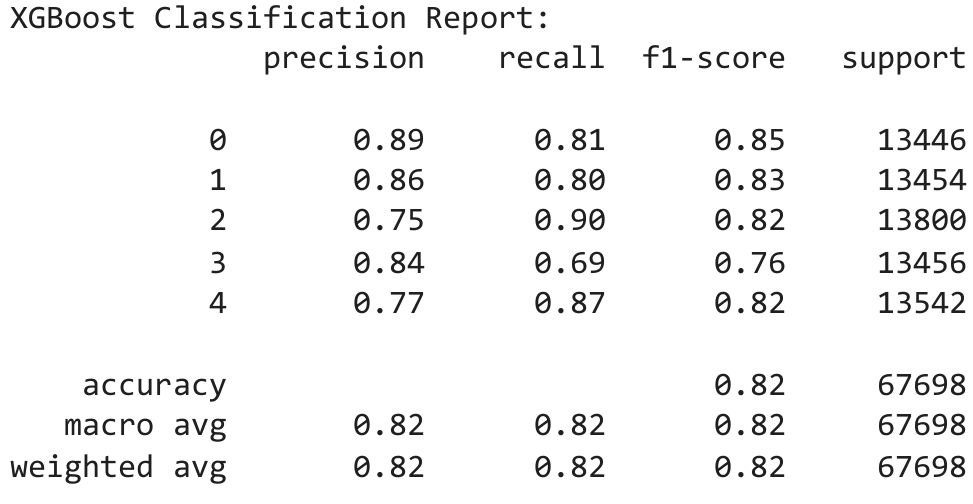
* 1. **Accuracy:** Here, we compare the actual and predicted class of each data point, and each match counts for one correct prediction. Accuracy is then given as the number of correct predictions divided by the total number of predictions. 
  2. **Recall:** The **true positive rate (TPR)**, or the proportion of all actual positives that were classified correctly as positives, is also known as [**recall**](https://developers.google.com/machine-learning/glossary#recall). 
  3. **Precision:** Precision is the proportion of all the model's positive classifications that are actually positive. 
  4. **F1 Score:** [It](https://www.geeksforgeeks.org/precision-recall-and-f1-score-using-r/)combines precision and recall into a single metric to balance their trade-off. It is helpful when both false positives and false negatives are important though it assumes precision and recall are equally important but in some situations one might matter more than the other.

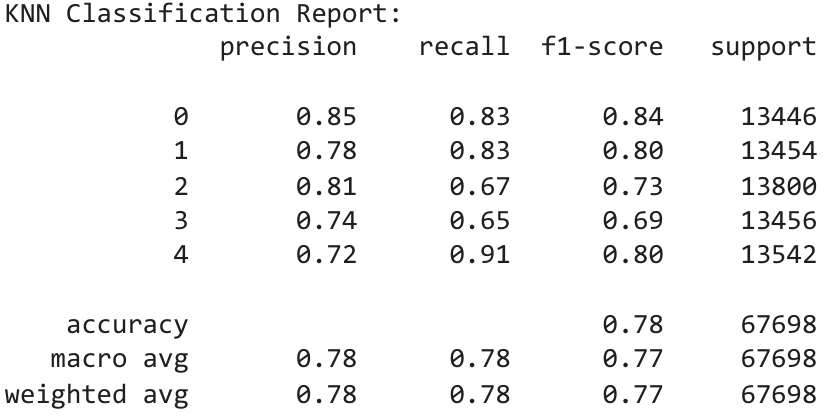


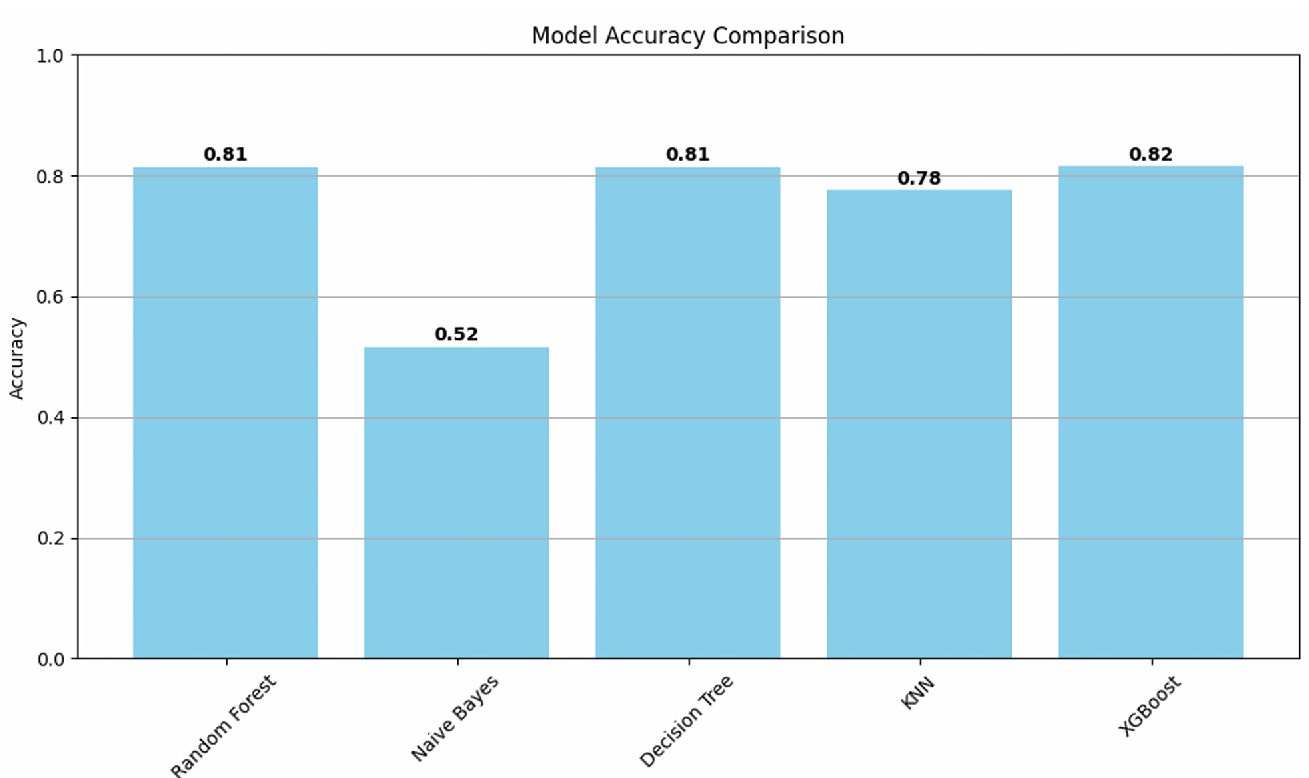
* 1. **Specificity:**It measures the ability of a model to correctly identify negative instances. Specificity is also known as the True Negative Rate. 

**Classification reports**

** **

** **

****

****

**RESULTS**

|  |  |
| --- | --- |
| **Model** | **Accuracy** |
| Random Forest | 81.46% |
| Decision Trees | 81.43% |
| K-NN | 77.58% |
| Naïve Bayes | 51.65% |
| XG Boost | 81.58% |