**Network instruction Detection System Using Machine Learning**

**Abstract:**

The objective of this study was to assess the performance of various resampling strategies aimed at mitigating the class imbalance problem in Network Intrusion Detection Systems (NIDS) using machine learning models and imbalanced benchmark datasets. Due to this class imbalance problem, detection of known or unknown attacks in NIDS often results in suboptimal performance. Resampling methods, statistically designed to generate synthetic samples from existing datasets, were employed to rebalance class labels and train the machine learning models. The Support Vector Machine (SVM), a robust supervised classifier, was utilized to classify data by identifying the optimal decision boundary that maximally separates different classes. In this context, efforts were made to enhance the effectiveness of these resampling techniques and consider the potential benefits of hybrid models. No resampling (NR), Synthetic Minority Over-sampling Technique (SMOTE), Random Under Sampling (RUS), Random Under Sampling and Random Over Sampling (RUS+ROS), and Random Under Sampling and SMOTE (RUS+SMOTE) were evaluated. The SVM classifier with Radial Basis Function (RBF) was employed, validated against the imbalanced benchmark dataset CICIDS-2017 (Canadian Institute for Cyber Security Intrusion Detection dataset-2017), to assess the effectiveness of these methods using performance metrics such as Accuracy, Precision, Recall, F1 score, and wall time. The proposed method achieved a remarkable accuracy of 99.63% in intrusion detection, demonstrating impressive results when compared to state-of-the-art methods for detecting network attacks on imbalanced datasets. The findings from this research provide valuable insights into the potential of various resampling methods in tackling class imbalance problems in NIDS.

**Introduction:**

Cloud technology, which provides on-demand network access to computing resources across various domains worldwide, has formed the backbone of the current digital era. Its primary objective is to offer customers an array of services under a pay-as-you-use model, requiring minimal supervision. Two of the most prevalent fields where cloud technology has been adopted are global communication and networking, where an enormous number of end-users and devices are connected to the cloud in cyberspace to access an array of amenities. The benefits of cloud computing, either in terms of service delivery or economic efficiency, are numerous. However, despite these advantages, the rapid advancement in communication technology has introduced numerous security and privacy challenges, such as maintaining confidentiality, integrity, and availability, leading to the emergence of a new class of cyber-attacks [1].

Traditional security measures, including firewalls and encryption of sensitive data, are widely deployed. However, they are increasingly considered outdated for organizations requiring robust security measures, such as government entities and military bases, largely due to the difficulties in human configuration and the extended timeframes required to develop advanced solutions for these attacks [2].

Among these emerging attacks, Distributed Denial of Service (DDoS) attacks hold significant importance due to their potential to critically impact cloud servers and consumers. DDoS attacks purposefully target websites, storage, cloud-hosted applications, and network setups, absorbing all available bandwidth and disrupting access for legitimate users and partners. This can tarnish the reputation of companies by reducing their productivity and affecting their profitability. These disruptive activities are frequently conducted through innovative network penetration methods by unauthorized users.

Although the cloud's elasticity principle ensures that service delivery remains uninterrupted, the service provider's bill is inevitably increased to maintain the Quality of Service (QoS) as per the Service Level Agreement (SLA). Therefore, DDoS attacks often lead to Economic Denial of Service (EDoS) attacks [3]. As such, it is crucial to mitigate these types of attacks in the cloud before the billing mechanism commences for the service provider. To combat these attacks, numerous approaches for intrusion prevention and detection systems have been developed and documented in the literature. However, to ensure the cloud's survival, there is a growing need for more robust mechanisms to defend against increasingly creative and sophisticated attacks. A significant portion of cloud attacks are network-based, and with the volume of data increasing every second, network traffic classification becomes an essential step in intrusion detection.

The sophistication and complexity of contemporary cyber-attacks have begun to challenge the efficacy of the existing statistical and threshold approaches [4]. Machine Learning (ML) techniques have emerged as automatic and relevant solutions, particularly suited to intrusion detection and prevention within the realm of cloud security [5-7]. The efficacy of intrusion detection heavily relies on the accurate representation of features in network traffic. However, standard datasets available for such tasks are typically of high dimensionality and unbalanced, where each class label is not equally represented. This imbalance can potentially impact the performance of the classifier [8, 9].

When ML models are trained on such imbalanced datasets, the resultant predictive outcomes often favor dominant classes, leading to poor classification rates for minority classes and potentially misdirecting the predictive analysis [10]. Thus, it is imperative to develop intelligent systems capable of overcoming these biases when confronted with such data imbalances. Consequently, learning from imbalanced data has become a significant area of research over the past two decades. The concept of class imbalance has been extensively studied in diverse application areas, including but not limited to medical science [11], sentiment analysis [12], bioinformatics, intrusion detection, text mining [13], credit scoring, and fraud detection.

Resampling methods offer a potential solution to this problem by adjusting the class ratio to create a balanced dataset. When integrated with classification techniques, these resampling methods have the potential to substantially improve the intrusion detection and classification rates. Therefore, the exploration and optimization of such approaches forms the basis of this study.

This study endeavors to examine the impact of various resampling techniques on intrusion detection, including but not limited to Synthetic Minority Over-sampling Technique (SMOTE), Random Under Sampling (RUS), Random Over Sampling (ROS), and a combination of RUS-ROS and RUS-SMOTE. The initial approach involves the under-sampling of the majority class, followed by the application of the Synthetic Minority Oversampling Technique (SMOTE) for oversampling [2]. Ultimately, a combination of over-sampling and under-sampling techniques is employed to seek further enhancement in intrusion detection performance.

To assess the effectiveness of these resampling techniques, the Support Vector Machine classifier, renowned for its robust generalization capabilities and pattern recognition skills, is utilized [14]. The investigation employs the “CICIDS-2017” dataset, a benchmark dataset extensively detailed in the study of Vamsi Krishna et al. [15].

Evaluation metrics are subsequently applied to compare and contrast the performance of the classifier for each resampling technique. Although the intrusion detection rate is somewhat lower when compared to the combination of unbalancing techniques, it remains significantly higher than when the imbalance issue is left unaddressed. Consequently, the results obtained indicate an increase in performance.

The CICIDS2017 dataset, which exhibits a high-class imbalance, is composed of five days' worth of typical traffic and attacker traffic information sourced from the Canadian Institute of Cybersecurity. It encompasses benign traffic as well as the most recent common attacks, distributed among 14 classes, thereby closely mirroring real-world data. Previous research recommends the development of multi-class detector models on Tuesday, Wednesday, and Thursday mornings [13]. Therefore, the present study has chosen to focus on the dataset from Wednesday.

Given the aforementioned complexities inherent in high-dimensional, imbalanced datasets, this study aims to identify an effective resampling approach to balance the dataset, thereby enhancing classification accuracy. The contributions of this research are multifaceted and are outlined as follows:

(1) A novel framework, integrating hybrid pre-processing, is proposed to analyze and classify various benign network activities and malicious actions, leveraging the application of resampling techniques in the cloud domain.

(2) The highly unbalanced CICIDS2017 dataset is rebalanced using an array of resampling techniques, rendering it more meaningful and informative for model training.

(3) A suitable resampling technique for the Support Vector Machine (SVM) with a Radial Basis Function (RBF) classifier is suggested.

(4) The CICIDS2017 dataset is selected for performance metric evaluation and computational time analysis.

(5) The efficacy of hybrid pre-processing techniques is examined, with a focus on addressing the issue of disproportionate class distribution in the dataset.

(6) A review of the area of class imbalance is conducted, and to bolster the detection of rare attacks, Synthetic Minority Oversampling (SMOTE), Random Under Sampling, and Random Over Sampling techniques are employed.

(7) Efforts are made to balance the dataset, mitigating the negative effects of an imbalanced dataset on minority intrusion detection rates and other performance metrics, including recall rates.

When class distributions within a dataset are unequal, F1-Score and Recall metrics offer an optimal evaluation of the classification model. Higher metrics correlate with an improved ML model, yielding consistent grades. Performance measures underscore the importance of balanced datasets in optimizing intrusion detection systems, highlighting the performance degradation caused by imbalanced datasets.

The remainder of the paper is structured as follows: Section 2 encompasses related studies conducted by several researchers in the field. Section 3 presents the proposed method. Section 4 is dedicated to results and discussion, and finally, Section 5 draws conclusions from the experimental findings.

**LITERATURE REVIEW**

The issue of class imbalance necessitates the utilization of resampling strategies. The confluence of resampling techniques with supervised classifiers becomes critical in the realm of network intrusion detection. Comprehensive surveys and analyses of diverse studies indicate that imbalanced learning contributes significantly to the performance of intrusion detection methodologies. This section provides a succinct overview of select previous studies. Recent developments in the field of resampling methods are also discussed. A novel hybrid framework, "ImmuneNet", is described in the study of Kumaar et al. [16] that aims at fortifying the security of patient records in healthcare systems. This 1230 framework is a synthesis of deep learning and feature engineering techniques. It employs an array of oversampling methods and hyper-parameter optimization strategies to enhance accuracy and performance. The "ImmuneNet" framework was subjected to rigorous testing against various benchmark datasets, including CIC-IDS-2017, CI-IDS-2018, and Bell DNS 2021. Additionally, four different machine learning algorithms were evaluated. The authors concluded that "ImmuneNet" demonstrated superior performance, achieving an accuracy of 99.19% and an ROC-AUC of 99.2%. In a study by Ustebay et al. [17], a fusion of recursive feature elimination via Random Forest and Deep Multilayer Perceptron (DMLP) is proposed for Intrusion Detection Systems (IDS). This research aims to tackle the challenges posed by big data, identifying the impact of diverse attributes from the dataset and determining the most informative features that meaningfully represent the data. In the Random Forest (RF) methodology, various tree structures are employed to discard non-essential features from a total of 80. Based on the graphical representations, 10 key features are selected. The truncated data is then subjected to binary classification using DMLP, resulting in an accuracy of 89%. In another research effort [18], an ensemble learning approach is proposed, constituting a resilient and efficacious intrusion detection framework using eXtreme Gradient Boosting (XGBoost) alongside an embedded feature selection methodology. Ensemble-based Gradient Boost Trees are utilized as a filter method, evaluating each feature based on its significance. The embedded feature selection approach unfolds in three phases, examining a subset of features by employing filter methods, wrapper methods, or hybrid methods. Subsequently, the imbalance issue is addressed by amalgamating similar types of attacks into a single category, resulting in categories of Normal, WebAttack, Infiltration, BruteForce, Dos, Botnet, PortScan, and DDoS types. This methodology is tested using the CICIDS 2017 dataset and is evaluated for both binary and multi-class classification problems. The XGBoost classifier is finally applied as an evaluator, achieving an accuracy of 99.86% and 99.90% for binary and multi-class classification, respectively. In the face of burgeoning data volumes and expansive Internet connectivity, the significance of intrusion detection in safeguarding our infrastructure and national security cannot be overstated. A model employing machine learning techniques for network intrusion detection is proposed by Tauscher et al. [19]. This model is bifurcated into two stages: the initial stage is dedicated to distinguishing normal from suspicious behaviors, while the subsequent stage classifies specific attacks. To address the predicament of class imbalance, the Synthetic Minority Oversampling Technique (SMOTE) is employed by the authors. The model's performance is evaluated with eight classifiers, among which an unsupervised autoencoder-based model rendered the most effective performance. All experiments were conducted using the NSLKDD dataset. However, a significant limitation of this study is the exclusive reliance on a single technique to balance the dataset. In the study of Fu et al. [2], a network intrusion detection system is introduced that addresses challenges such as data imbalance and low detection rate accuracy in cloud computing. The ADASYN oversampling algorithm is utilized to increase the number of samples in minority class labels, thereby addressing data imbalance. Furthermore, the model's generalization capability is enhanced, and the network structure is refined by integrating the channel attention mechanism with bidirectional LSTM networks. Comparison with other models in the literature suggests that the proposed DLNID yields superior classification results. The model is evaluated using the publicly accessible benchmark dataset NSL-KDD, outperforming the other methods compared, with an accuracy of 90.73% and an F1-score of 89.65%. To fortify defenses against emerging threats, an advanced anomaly intrusion detection system is proposed by Elmasri et al. [20]. The proposed model leverages both KNN and LOF algorithms, supplemented by an enhanced version utilizing PCA. In terms of detecting novel attacks, both models demonstrate considerable proficiency, achieving detection rates of 88.3% and 90.54%, respectively. Moreover, a significant reduction in time complexity is observed across all models. In another comprehensive study by Razan Abdulhammed et al. [21], two feature dimensionality reduction methods, Autoencoder (AE) and Principal Component Analysis (PCA), were employed. The authors tested various classifiers, including Random Forest (RF), Bayesian Network, Linear Discriminant Analysis (LDA), and Quadratic Discriminant Analysis (QDA), using a subset of the CICIDS 2017 dataset. AE was used to reduce the dimensionality of the CICIDS 2017 dataset from 81 to 59, while PCA was used to further reduce it to 10. In both instances, Random Forest exhibited superior accuracy compared to the other classifiers. A framework for Network Intrusion Detection Systems (NIDS) designed to combat a wide range of contemporary and emerging threats is proposed by Ahmed et al. [22]. Five machine learning algorithms, namely Random Forest (RF), Decision Tree (DT), Logistic Regression (LR), K-Nearest Neighbor (KNN), and Artificial Neural Networks (ANN), were evaluated for their effectiveness in classifying attacks. The UNSW-NB15 dataset was used to assess the performance of the proposed framework. Feature selection and SMOTE resampling methods were employed in conjunction with these algorithms to balance the classification labels. Accuracies were reported for classification models both with and without feature selection, following the handling of class imbalance. In both scenarios, the Random Forest (RF) classifier demonstrated superior performance, achieving 95.1% accuracy when used in conjunction with SMOTE and PCA. However, time complexity was not considered in this study. In the context of IoT networks, where security and privacy are paramount, a multi-stage classification system is proposed by Qaddoura et al. [23]. This system comprises three stages: dataset size reduction using k-means clustering, dataset balancing using the SMOTE oversampling technique, and the subsequent classification of the balanced data. A comparative analysis of various classifiers in the final stage revealed SVMSMOTE to be the most effective. The imbalanced dataset CIDDS-001 was addressed using various techniques in the study of Abdulhammed et al. [24]. The Synthetic Minority Reconstruction Technique (SMRT) in combination with a Variational Autoencoder (VAE) was applied to the data for classification. A multitude of sampling methods were used to address class imbalance, including the down-sampling of majority classes and up-sampling of minority classes. Following this, the dataset underwent analysis using various machine/deep learning algorithms, such as random forest and deep neural networks, to evaluate progress in attack detection. However, computational overhead was not considered in this study

**Existing system:-**

In existing system we are using SMOTE ,

It is vital to precisely estimate INSTRUSION DETECTION .

There are a variety of traditional methods for measuring it, but the findings aren't always exact, and it necessitates a lot of mathematical calculations.

**Disadvantages:-**

* Less amount of accuracy score
* Small level dataset
* Applicable on small level prediction work.

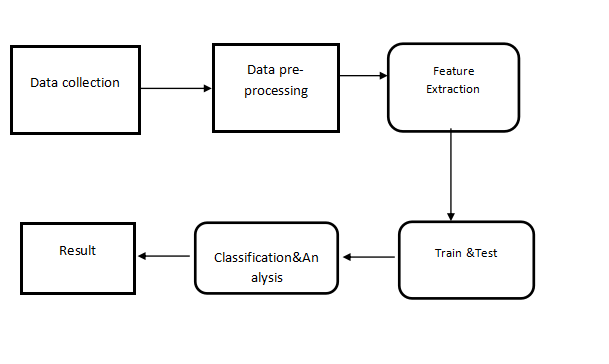
**Proposed system:-**

Decision Tree, Random Forest, and Support Vector Machine and Kmeans clustering are some of the Machine Learning algorithms used.

**Advantages:-**

* Increasing the accuracy score
* Large amount of feature we are taking for the training and testing.

**System Architecture:**



**Software and Hardware Requirements:**

**Hardware:**

* OS – Windows 7, 8 and 10 (32 and 64 bit)
* RAM – 4GB

**Software:**

* Python
* Anaconda

**Feasibility study**

Feasibility study in the sense it's a practical approach of implementing the proposed model of system . Here for a machine learning projects .we generally collect the input from online websites and filter the input data and visualize them in graphical format and then the data is divided for training and testing . That training is testing data is given to the algorithms to predict the data .

1. First, we take dataset.

2. Filter dataset according to requirements and create a new dataset which has attribute according to analysis to be done

3. Perform Pre-Processing on the dataset

4. Split the data into training and testing

5. Train the model with training data then analyze testing dataset over classification algorithm

6. Finally you will get results as accuracy metrics.

**Modules**

1. **DATA COLLECTION**
2. **DATA PRE-PROCESSING**
3. **FEATURE EXTRATION**
4. **EVALUATION MODE**

**DATA COLLECTION**

Data collection is a process in which information is gathered from many sources which is later used to develop the machine learning models. The data should be stored in a way that makes sense for problem. In this step the data set is converted into the understandable format which can be fed into machine learning models.

Data used in this paper is a set of cervical cancer data with 15 features . This step is concerned with selecting the subset of all available data that you will be working with. ML problems start with data preferably, lots of data (examples or observations) for which you already know the target answer. Data for which you already know the target answer is called *labelled data*.

**DATA PRE-PROCESSING**

Organize your selected data by formatting, cleaning and sampling from it.

Three common data pre-processing steps are:

**Formatting**: The data you have selected may not be in a format that is suitable for you to work with. The data may be in a relational database and you would like it in a flat file, or the data may be in a proprietary file format and you would like it in a relational database or a text file.

**Cleaning:** Cleaning data is the removal or fixing of missing data. There may be data instances that are incomplete and do not carry the data you believe you need to address the problem. These instances may need to be removed. Additionally, there may be sensitive information in some of the attributes and these attributes may need to be anonymized or removed from the data entirely.

**Sampling:** There may be far more selected data available than you need to work with. More data can result in much longer running times for algorithms and larger computational and memory requirements. You can take a smaller representative sample of the selected data that may be much faster for exploring and prototyping solutions before considering the whole dataset.

**FEATURE EXTRATION**

Next thing is to do Feature extraction is an attribute reduction process. Unlike feature selection, which ranks the existing attributes according to their predictive significance, feature extraction actually transforms the attributes. The transformed attributes, or features, are linear combinations of the original attributes.  Finally, our models are trained using Classifier algorithm. We use classify module on Natural Language Toolkit library on Python. We use the labelled dataset gathered. The rest of our labelled data will be used to evaluate the models. Some machine learning algorithms were used to classify pre-processed data. The chosen classifiers were Random forest. These algorithms are very popular in text classification tasks.

**EVALUATION MODEL**

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. Evaluating model performance with the data used for training is not acceptable in data science because it can easily generate overoptimistic and over fitted models. There are two methods of evaluating models in data science, Hold-Out and Cross-Validation. To avoid over fitting, both methods use a test set (not seen by the model) to evaluate model performance.

Performance of each classification model is estimated base on its averaged. The result will be in the visualized form. Representation of classified data in the form of graphs.

**Accuracy** is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

**DATA FLOW DIAGRAM**

**LEVEL 0**

**LEVEL 1**

**LEVEL 2**

**UML Diagrams**

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as:

* actors
* business processes
* (logical) components
* activities
* programming language statements
* database schemas, and
* Reusable software components.

UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

**Sequence Diagram:**

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

**Activity Diagrams-:**

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency.In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

**Usecase diagram:**

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

UML was created by Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

OMG is continuously putting effort to make a truly industry standard.

UML stands for Unified Modeling Language.

UML is a pictorial language used to make software blue prints

**Class diagram**

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling.[1] The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

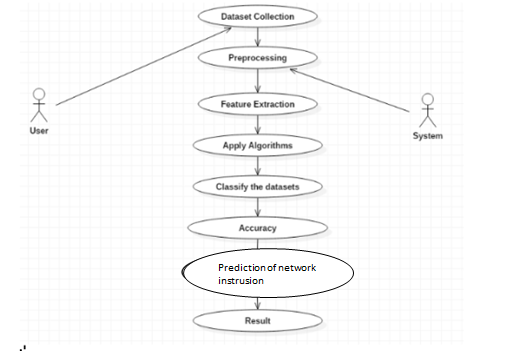
In the diagram, classes are represented with boxes that contain three compartments:

The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.

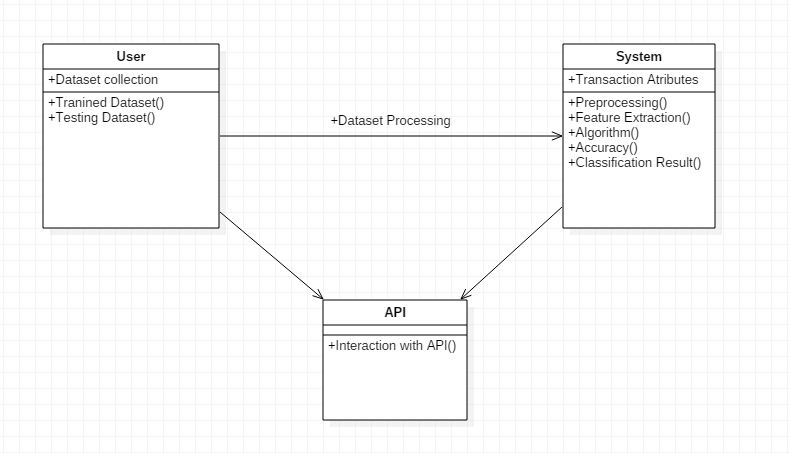
The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.

The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.

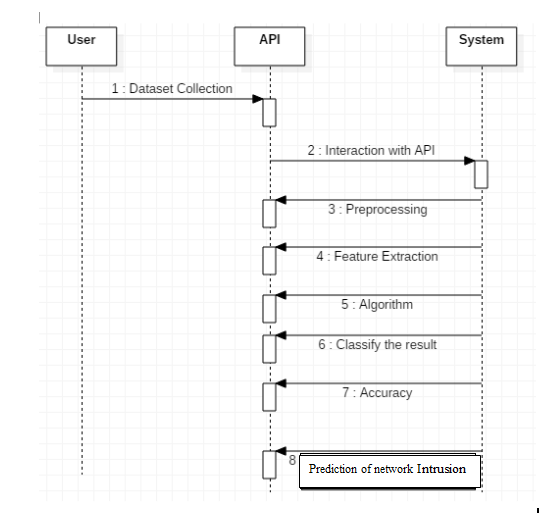
**Class diagram**



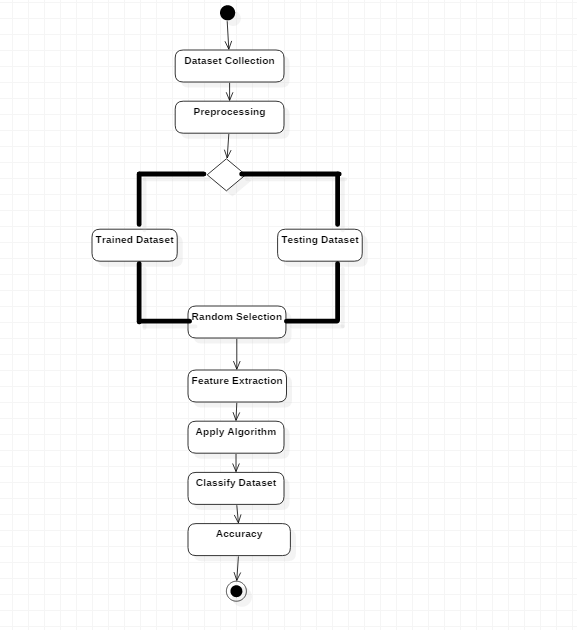
**CLASS DIAGRAM**

****

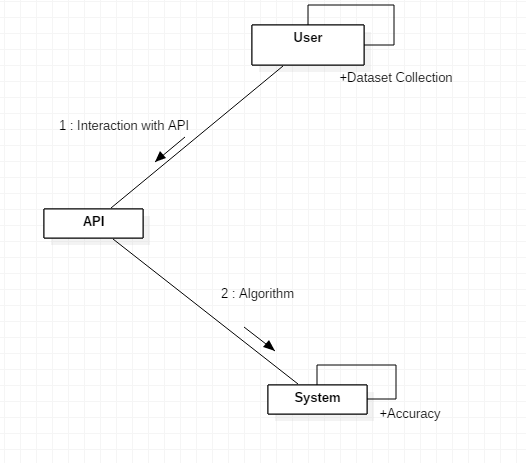
SEQUENCE DIAGRAM



**ACTIVITY DIAGRAM**



COLLABRATION DIAGRAM



**Algoritnms:**

**1.Decision tree**

**2.Random forest**

**3.Support vector machine**

**4.Kmeans Clustering**

**Decision tree:**

A decision tree classifier is a type of supervised machine learning algorithm used primarily for classification tasks. It operates by recursively partitioning the input space into regions or subsets, with each partition corresponding to a decision rule based on feature values. Here are some key points about decision tree classifiers:

Tree Structure: At the root of the tree, you have the entire dataset. As you move down the tree, the dataset gets partitioned into smaller subsets based on feature values, leading to a hierarchical structure resembling a tree.

Nodes: Each decision point or partition in the tree is represented by a node. The topmost node is called the root node. Nodes that do not split further are called leaf nodes and represent the final decision or prediction.

Splitting Criteria: The decision to split data at each node is based on specific criteria that aim to maximize the homogeneity of the target variable within each partition. Common criteria include:

Gini impurity

Information gain (using entropy or information gain ratio)

Classification error

Pruning: Decision trees can grow to be very complex and can overfit the training data. Pruning techniques are applied to simplify the tree by removing branches that provide little power in predicting target values, thus improving the model's generalization to unseen data.

**Advantages:**

Easy to understand and interpret.

Can handle both numerical and categorical data.

Non-parametric, meaning they make no assumptions about the underlying distributions of the data.

**Disadvantages:**

Can create over-complex trees that do not generalize well to new data (overfitting).

Can be sensitive to small changes in the training data, leading to different tree structures.

May not be the most accurate method for classification compared to other algorithms like random forests or gradient boosting machines in some scenarios.

Applications: Decision trees are widely used in various domains such as:

Healthcare for disease prediction.

Finance for credit scoring.

Marketing for customer segmentation.

Any domain where a clear decision-making process based on features is required.

Extensions: To address some of the limitations of basic decision trees, ensemble methods like Random Forests and Gradient Boosted Trees have been developed. These methods use multiple decision trees in conjunction, leading to improved performance and robustness.

When using decision trees or any machine learning algorithm, it's essential to preprocess the data properly, handle missing values, and tune hyperparameters to optimize performance and avoid common pitfalls like overfitting.

**Random forest:**

Random Forest is an ensemble learning method used for both classification and regression tasks. It constructs multiple decision trees during training and outputs the mode of the classes (in classification) or the mean prediction (in regression) of the individual trees.

Here's an overview of Random Forest:

Ensemble Learning: Random Forest is an example of ensemble learning, where multiple models (in this case, decision trees) are trained to solve the same problem and their predictions are aggregated to produce a final prediction. This aggregation typically reduces overfitting and improves the model's generalization capability.

Bootstrap Sampling: For each tree in the Random Forest, a bootstrap sample (sample with replacement) is drawn from the original dataset. This means that each tree is trained on a slightly different dataset, introducing diversity among the trees.

Feature Randomness: At each split in a decision tree, instead of considering all features, a random subset of features is considered. This further increases the diversity among the trees and ensures that the trees are not highly correlated.

Voting: For classification tasks, the mode (most frequent class) of the classes predicted by individual trees is taken as the final prediction. For regression tasks, the mean of the predictions from individual trees is taken as the final prediction.

**Advantages:**

Reduces overfitting compared to individual decision trees.

Provides a robust and accurate prediction.

Can handle a large number of features and variables without feature selection.

Provides feature importance scores, which can help in understanding the importance of different features.

**Disadvantages:**

Random Forest models can be computationally expensive and require more resources compared to single decision tree models.

They can be more challenging to interpret compared to individual decision trees.

Applications: Random Forest is widely used in various domains, including:

Finance for credit risk modeling.

Healthcare for disease prediction and diagnosis.

E-commerce for recommendation systems.

Remote sensing for land cover classification.

Any task where a robust and accurate prediction is required.

Tuning: Random Forest has several hyperparameters that can be tuned to optimize performance, such as the number of trees, depth of trees, and the number of features to consider at each split. Techniques like cross-validation can be used to find the optimal set of hyperparameters.

In summary, Random Forest is a powerful and versatile machine learning algorithm that combines the strength of multiple decision trees to produce robust and accurate predictions. Proper data preprocessing, feature engineering, and hyperparameter tuning are essential for obtaining the best performance from a Random Forest model.

**Support vector machine:**

Support Vector Machines (SVM) is a supervised machine learning algorithm primarily used for classification tasks, although it can be applied to regression as well. SVM aims to find the optimal hyperplane that best separates the data into different classes by maximizing the margin between classes.

Here's an overview of Support Vector Machines:

Hyperplane: In a binary classification scenario, a hyperplane in an

n-dimensional space is a flat affine subspace of dimension

−

1

n−1 that divides the space into two half-spaces. The goal of SVM is to find the hyperplane that separates the classes with the largest margin.

Support Vectors: These are the data points that lie closest to the hyperplane and have a non-zero (or nonzero and positive) coefficient in the representation of the hyperplane. They play a crucial role in defining the decision boundary.

Margin: The margin is the distance between the closest points (support vectors) of the two classes to the hyperplane. SVM aims to maximize this margin, as it provides a measure of robustness to the model.

Kernel Trick: SVM can efficiently handle non-linear classification tasks by using the kernel trick. Kernels transform the input data into a higher-dimensional space, making it possible to find a linear separating hyperplane in that space. Common kernels include:

Linear Kernel

Polynomial Kernel

Radial Basis Function (RBF) or Gaussian Kernel

Sigmoid Kernel

Regularization Parameter (C): SVM introduces a regularization parameter

C that controls the trade-off between maximizing the margin and minimizing classification error on the training data. A smaller value of C allows for a wider margin but may lead to misclassification of some training examples, while a larger value of

C emphasizes correct classification of training examples but may result in a narrower margin.

**Advantages:**

Effective in high-dimensional spaces and with a clear margin of separation.

Versatile due to the use of different kernel functions for handling non-linear data.

Memory efficient as it uses only a subset of training points (support vectors) in the decision function.

**Disadvantages:**

Not suitable for large datasets due to high training time complexity, especially with non-linear kernels.

Can be sensitive to the choice of the regularization parameter

C and the kernel parameters.

May not perform well when the number of features is much greater than the number of samples.

Applications: SVM has been successfully applied to various domains, including:

Text classification and sentiment analysis.

Image recognition and object detection.

Bioinformatics for protein classification and gene expression analysis.

Financial forecasting and risk management.

When using SVM, it's essential to preprocess the data properly, normalize or standardize features, and tune hyperparameters to achieve the best performance. SVMs are powerful models but require careful parameter selection and understanding of the underlying data distribution for optimal results.

**Kmeans Clustering:**

K-Means is a popular clustering algorithm that partitions a dataset into K distinct, non-overlapping subsets (clusters). Each data point belongs to the cluster with the nearest mean. It's an iterative algorithm that aims to minimize the sum of squared distances between data points and their cluster's centroid.

**Advantages:**

**Simple and Intuitive:**

K-Means is easy to understand and implement, making it a straightforward choice for many applications.

The algorithm is based on the principle of minimizing intra-cluster distances and maximizing inter-cluster distances.

**Computational Efficiency:**

K-Means is computationally efficient and can handle large datasets with a moderate number of features.

It converges relatively quickly, making it suitable for scenarios where real-time clustering is required.

**Scalability:**

It scales well with the number of data points.

**Versatility:**

Applicable to various types of data, assuming the data satisfies the assumptions of the algorithm (e.g., roughly spherical clusters).

**Well-Suited for Equal-Sized Clusters:**

Works well when the clusters are roughly equal in size.

**Disadvantages:**

**Dependence on Initial Centroids:**

The final clusters can be sensitive to the initial choice of centroids. Different initializations can lead to different results.

K-Means++, a smart initialization technique, is often used to mitigate this issue.

**Sensitive to Outliers:**

K-Means is sensitive to outliers, as a single outlier can significantly affect the position of the cluster centroids.

**Assumption of Circular/Spherical Clusters:**

K-Means assumes that clusters are roughly spherical, isotropic, and equally sized. It may not perform well on elongated or irregularly shaped clusters.

**Needs the Number of Clusters (K) to be Specified:**

The number of clusters (K) needs to be specified in advance, which can be a drawback when the true number of clusters is unknown.

**Sensitive to Scaling:**

The algorithm is sensitive to the scale of the features. It is recommended to scale features before applying K-Means.

**Not Suitable for Non-Convex Clusters:**

K-Means may struggle with clusters of non-convex shapes or clusters with complex structures.

**Domain Overview:**

**MACHINE LEARNING**

Machine Learning is a system that can learn from example through self-improvement and without being explicitly coded by programmer. The breakthrough comes with the idea that a machine can singularly learn from the data (i.e., example) to produce accurate results.

Machine learning combines data with statistical tools to predict an output. This output is then used by corporate to makes actionable insights. Machine learning is closely related to data mining and Bayesian predictive modeling. The machine receives data as input, use an algorithm to formulate answers.

A typical machine learning tasks are to provide a recommendation. For those who have a Netflix account, all recommendations of movies or series are based on the user's historical data. Tech companies are using unsupervised learning to improve the user experience with personalizing recommendation.

Machine learning is also used for a variety of task like fraud detection, predictive maintenance, portfolio optimization, automatize task and so on.

**Machine Learning vs. Traditional Programming**

Traditional programming differs significantly from machine learning. In traditional programming, a programmer code all the rules in consultation with an expert in the industry for which software is being developed. Each rule is based on a logical foundation; the machine will execute an output following the logical statement. When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain.

**COMPUTER**

**DATA RULES**

**OUTPUT**

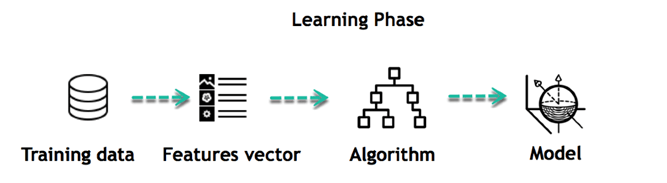
**Machine Learning**

## How does Machine learning work?

Machine learning is the brain where all the learning takes place. The way the machine learns is similar to the human being. Humans learn from experience. The more we know, the more easily we can predict. By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation. Machines are trained the same. To make an accurate prediction, the machine sees an example. When we give the machine a similar example, it can figure out the outcome. However, like a human, if its feed a previously unseen example, the machine has difficulties to predict.

The core objective of machine learning is the **learning** and **inference**. First of all, the machine learns through the discovery of patterns. This discovery is made thanks to the **data**. One crucial part of the data scientist is to choose carefully which data to provide to the machine. The list of attributes used to solve a problem is called a **feature vector.** You can think of a feature vector as a subset of data that is used to tackle a problem.

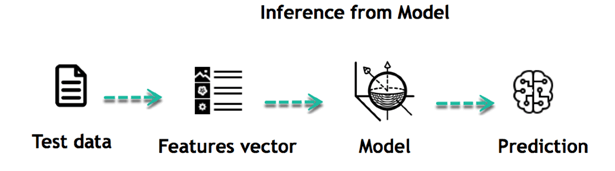
The machine uses some fancy algorithms to simplify the reality and transform this discovery into a **model**. Therefore, the learning stage is used to describe the data and summarize it into a model.



For instance, the machine is trying to understand the relationship between the wage of an individual and the likelihood to go to a fancy restaurant. It turns out the machine finds a positive relationship between wage and going to a high-end restaurant: This is the model

#### Inferring

When the model is built, it is possible to test how powerful it is on never-seen-before data. The new data are transformed into a features vector, go through the model and give a prediction. This is all the beautiful part of machine learning. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data.

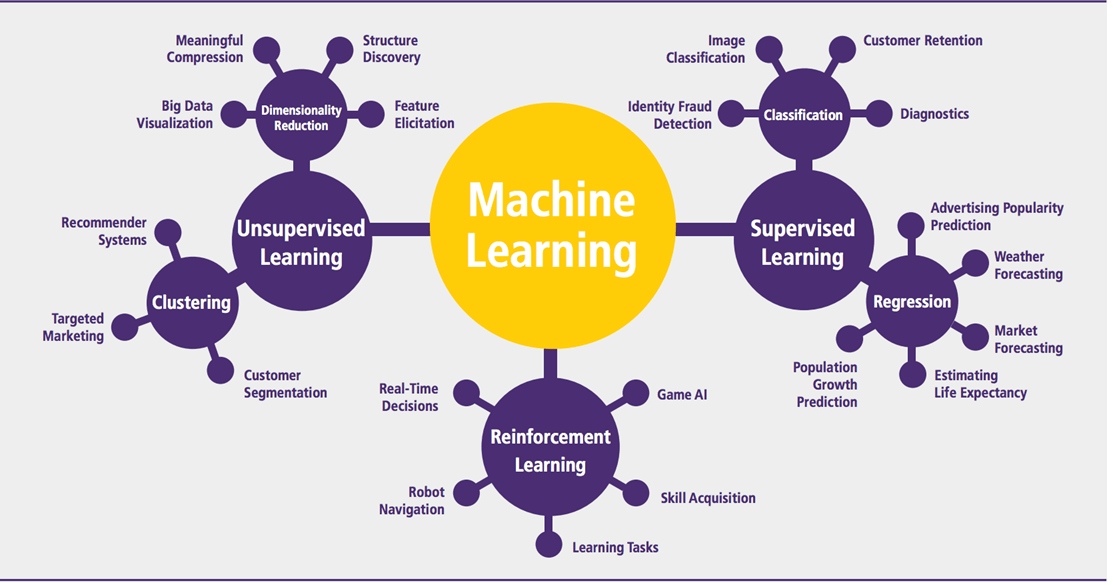


The life of Machine Learning programs is straightforward and can be summarized in the following points:

1. Define a question
2. Collect data
3. Visualize data
4. Train algorithm
5. Test the Algorithm
6. Collect feedback
7. Refine the algorithm
8. Loop 4-7 until the results are satisfying
9. Use the model to make a prediction

Once the algorithm gets good at drawing the right conclusions, it applies that knowledge to new sets of data.

## Machine learning Algorithms and where they are used?



Machine learning can be grouped into two broad learning tasks: Supervised and Unsupervised. There are many other algorithms

#### Supervised learning

An algorithm uses training data and feedback from humans to learn the relationship of given inputs to a given output. For instance, a practitioner can use marketing expense and weather forecast as input data to predict the sales of cans.

You can use supervised learning when the output data is known. The algorithm will predict new data.

There are two categories of supervised learning:

* Classification task
* Regression task

#### Classification

Imagine you want to predict the gender of a customer for a commercial. You will start gathering data on the height, weight, job, salary, purchasing basket, etc. from your customer database. You know the gender of each of your customer, it can only be male or female. The objective of the classifier will be to assign a probability of being a male or a female (i.e., the label) based on the information (i.e., features you have collected). When the model learned how to recognize male or female, you can use new data to make a prediction. For instance, you just got new information from an unknown customer, and you want to know if it is a male or female. If the classifier predicts male = 70%, it means the algorithm is sure at 70% that this customer is a male, and 30% it is a female.

The label can be of two or more classes. The above example has only two classes, but if a classifier needs to predict object, it has dozens of classes (e.g., glass, table, shoes, etc. each object represents a class)

#### Regression

When the output is a continuous value, the task is a regression. For instance, a financial analyst may need to forecast the value of a stock based on a range of feature like equity, previous stock performances, macroeconomics index. The system will be trained to estimate the price of the stocks with the lowest possible error.

|  |  |  |
| --- | --- | --- |
| **Algorithm Name** | **Description** | **Type** |
| **Linear regression** | Finds a way to correlate each feature to the output to help predict future values. | Regression |
| **Logistic regression** | Extension of linear regression that's used for classification tasks. The output variable 3is binary (e.g., only black or white) rather than continuous (e.g., an infinite list of potential colors) | Classification |
| **Decision tree** | Highly interpretable classification or regression model that splits data-feature values into branches at decision nodes (e.g., if a feature is a color, each possible color becomes a new branch) until a final decision output is made | Regression Classification |
| **Naive Bayes** | The Bayesian method is a classification method that makes use of the Bayesian theorem. The theorem updates the prior knowledge of an event with the independent probability of each feature that can affect the event. | Regression Classification |
| **Support vector machine** | Support Vector Machine, or SVM, is typically used for the classification task. SVM algorithm finds a hyperplane that optimally divided the classes. It is best used with a non-linear solver. | Regression (not very common) Classification |
| **Random forest** | The algorithm is built upon a decision tree to improve the accuracy drastically. Random forest generates many times simple decision trees and uses the 'majority vote' method to decide on which label to return. For the classification task, the final prediction will be the one with the most vote; while for the regression task, the average prediction of all the trees is the final prediction. | Regression Classification |
| **AdaBoost** | Classification or regression technique that uses a multitude of models to come up with a decision but weighs them based on their accuracy in predicting the outcome | Regression Classification |
| **Gradient-boosting trees** | Gradient-boosting trees is a state-of-the-art classification/regression technique. It is focusing on the error committed by the previous trees and tries to correct it. | Regression Classification |

#### Unsupervised learning

In unsupervised learning, an algorithm explores input data without being given an explicit output variable (e.g., explores customer demographic data to identify patterns)

You can use it when you do not know how to classify the data, and you want the algorithm to find patterns and classify the data for you

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **Description** | **Type** |
| **K-means clustering** | Puts data into some groups (k) that each contains data with similar characteristics (as determined by the model, not in advance by humans) | Clustering |
| **Gaussian mixture model** | A generalization of k-means clustering that provides more flexibility in the size and shape of groups (clusters | Clustering |
| **Hierarchical clustering** | Splits clusters along a hierarchical tree to form a classification system.  Can be used for Cluster loyalty-card customer | Clustering |
| **Recommender system** | Help to define the relevant data for making a recommendation. | Clustering |
| **PCA/T-SNE** | Mostly used to decrease the dimensionality of the data. The algorithms reduce the number of features to 3 or 4 vectors with the highest variances. | Dimension Reduction |

**Application of Machine learning**

**Augmentation**:

* Machine learning, which assists humans with their day-to-day tasks, personally or commercially without having complete control of the output. Such machine learning is used in different ways such as Virtual Assistant, Data analysis, software solutions. The primary user is to reduce errors due to human bias.

**Automation**:

* Machine learning, which works entirely autonomously in any field without the need for any human intervention. For example, robots performing the essential process steps in manufacturing plants.

**Finance Industry**

* Machine learning is growing in popularity in the finance industry. Banks are mainly using ML to find patterns inside the data but also to prevent fraud.

**Government organization**

* The government makes use of ML to manage public safety and utilities. Take the example of China with the massive face recognition. The government uses Artificial intelligence to prevent jaywalker.

**Healthcare industry**

* Healthcare was one of the first industry to use machine learning with image detection.

**Marketing**

* Broad use of AI is done in marketing thanks to abundant access to data. Before the age of mass data, researchers develop advanced mathematical tools like Bayesian analysis to estimate the value of a customer. With the boom of data, marketing department relies on AI to optimize the customer relationship and marketing campaign.

**Example of application of Machine Learning in Supply Chain**

Machine learning gives terrific results for visual pattern recognition, opening up many potential applications in physical inspection and maintenance across the entire supply chain network.

Unsupervised learning can quickly search for comparable patterns in the diverse dataset. In turn, the machine can perform quality inspection throughout the logistics hub, shipment with damage and wear.

For instance, IBM's Watson platform can determine shipping container damage. Watson combines visual and systems-based data to track, report and make recommendations in real-time.

In past year stock manager relies extensively on the primary method to evaluate and forecast the inventory. When combining big data and machine learning, better forecasting techniques have been implemented (an improvement of 20 to 30 % over traditional forecasting tools). In term of sales, it means an increase of 2 to 3 % due to the potential reduction in inventory costs.

**Example of Machine Learning Google Car**

For example, everybody knows the Google car. The car is full of lasers on the roof which are telling it where it is regarding the surrounding area. It has radar in the front, which is informing the car of the speed and motion of all the cars around it. It uses all of that data to figure out not only how to drive the car but also to figure out and predict what potential drivers around the car are going to do. What's impressive is that the car is processing almost a gigabyte a second of data.

Deep Learning

Deep learning is a computer software that mimics the network of neurons in a brain. It is a subset of machine learning and is called deep learning because it makes use of deep neural networks. The machine uses different layers to learn from the data. The depth of the model is represented by the number of layers in the model. Deep learning is the new state of the art in term of AI. In deep learning, the learning phase is done through a neural network.

**Reinforcement Learning**

Reinforcement learningis a subfield of machine learning in which systems are trained by receiving virtual "rewards" or "punishments," essentially learning by trial and error. Google's DeepMind has used reinforcement learning to beat a human champion in the Go games. Reinforcement learning is also used in video games to improve the gaming experience by providing smarter bot.

One of the most famous algorithms are:

* Q-learning
* Deep Q network
* State-Action-Reward-State-Action (SARSA)
* Deep Deterministic Policy Gradient (DDPG)

**Applications/ Examples of deep learning applications**

**AI in Finance:**The financial technology sector has already started using AI to save time, reduce costs, and add value. Deep learning is changing the lending industry by using more robust credit scoring. Credit decision-makers can use AI for robust credit lending applications to achieve faster, more accurate risk assessment, using machine intelligence to factor in the character and capacity of applicants.

Underwrite is a Fintech company providing an AI solution for credit makers company. underwrite.ai uses AI to detect which applicant is more likely to pay back a loan. Their approach radically outperforms traditional methods.

**AI in HR:**Under Armour, a sportswear company revolutionizes hiring and modernizes the candidate experience with the help of AI. In fact, Under Armour Reduces hiring time for its retail stores by 35%. Under Armour faced a growing popularity interest back in 2012. They had, on average, 30000 resumes a month. Reading all of those applications and begin to start the screening and interview process was taking too long. The lengthy process to get people hired and on-boarded impacted Under Armour's ability to have their retail stores fully staffed, ramped and ready to operate.

At that time, Under Armour had all of the 'must have' HR technology in place such as transactional solutions for sourcing, applying, tracking and onboarding but those tools weren't useful enough. Under armour choose **HireVue**, an AI provider for HR solution, for both on-demand and live interviews. The results were bluffing; they managed to decrease by 35% the time to fill. In return, the hired higher quality staffs.

**AI in Marketing:**AI is a valuable tool for customer service management and personalization challenges. Improved speech recognition in call-center management and call routing as a result of the application of AI techniques allows a more seamless experience for customers.

For example, deep-learning analysis of audio allows systems to assess a customer's emotional tone. If the customer is responding poorly to the AI chatbot, the system can be rerouted the conversation to real, human operators that take over the issue.

Apart from the three examples above, AI is widely used in other sectors/industries.

**Artificial Intelligence**

**Deep Learning**

**Machine Learning**

ML

## Difference between Machine Learning and Deep Learning

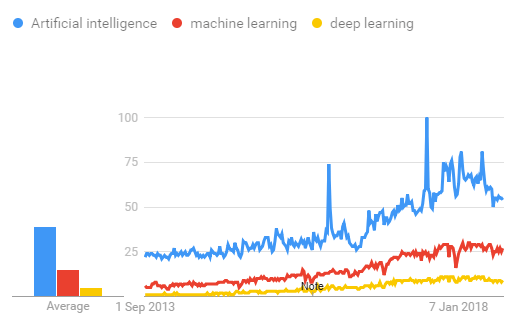
|  |  |  |
| --- | --- | --- |
|  | **Machine Learning** | **Deep Learning** |
| **Data Dependencies** | Excellent performances on a small/medium dataset | Excellent performance on a big dataset |
| **Hardware dependencies** | Work on a low-end machine. | Requires powerful machine, preferably with GPU: DL performs a significant amount of matrix multiplication |
| **Feature engineering** | Need to understand the features that represent the data | No need to understand the best feature that represents the data |
| **Execution time** | From few minutes to hours | Up to weeks. Neural Network needs to compute a significant number of weights |
| **Interpretability** | Some algorithms are easy to interpret (logistic, decision tree), some are almost impossible (SVM, XGBoost) | Difficult to impossible |

## When to use ML or DL?

In the table below, we summarize the difference between machine learning and deep learning.

|  |  |  |
| --- | --- | --- |
|  | **Machine learning** | **Deep learning** |
| **Training dataset** | Small | Large |
| **Choose features** | Yes | No |
| **Number of algorithms** | Many | Few |
| **Training time** | Short | Long |

With machine learning, you need fewer data to train the algorithm than deep learning. Deep learning requires an extensive and diverse set of data to identify the underlying structure. Besides, machine learning provides a faster-trained model. Most advanced deep learning architecture can take days to a week to train. The advantage of deep learning over machine learning is it is highly accurate. You do not need to understand what features are the best representation of the data; the neural network learned how to select critical features. In machine learning, you need to choose for yourself what features to include in the model.



## TensorFlow

The most famous deep learning library in the world is Google's TensorFlow. Google product uses machine learning in all of its products to improve the search engine, translation, image captioning or recommendations.

To give a concrete example, Google users can experience a faster and more refined the search with AI. If the user types a keyword a the search bar, Google provides a recommendation about what could be the next word.

Google wants to use machine learning to take advantage of their massive datasets to give users the best experience. Three different groups use machine learning:

* Researchers
* Data scientists
* Programmers.

They can all use the same toolset to collaborate with each other and improve their efficiency.

Google does not just have any data; they have the world's most massive computer, so TensorFlow was built to scale. TensorFlow is a library developed by the Google Brain Team to accelerate machine learning and deep neural network research.

It was built to run on multiple CPUs or GPUs and even mobile operating systems, and it has several wrappers in several languages like Python, C++ or Java.

In this tutorial, you will learn

**TensorFlow Architecture**

Tensorflow architecture works in three parts:

* Preprocessing the data
* Build the model
* Train and estimate the model

It is called Tensorflow because it takes input as a multi-dimensional array, also known as **tensors**. You can construct a sort of **flowchart** of operations (called a Graph) that you want to perform on that input. The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output.

**Code:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")

df=pd.read\_excel("Book1(1).xlsx")

df

df.shape

df.head()

df.tail()

df.info()

df.describe()

df.isnull().sum()

df.dropna(inplace=True)

df.isnull().sum()

df.isna().sum()

df.columns

df.duplicated().sum()

df.drop\_duplicates(inplace=True)

df.duplicated().sum()

df['Protocol'].value\_counts()

from sklearn import preprocessing

le=preprocessing.LabelEncoder()

df['Protocol'] = le.fit\_transform(df['Protocol'].values.reshape(-1,1).ravel())

df.head()

df['Protocol'].value\_counts()

df

df.shape

print(df.columns)

import seaborn as sns

import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))

sns.heatmap(df.isnull(), annot=True, cmap='coolwarm')

plt.title('Heatmap with Null Values Filled')

plt.show()

sns.distplot(df)

plt.show()

sns.countplot(x=df['Protocol'])

plt.title('No of protocal types',fontsize=15)

plt.show()

sns.countplot(x=df[' Destination Port'])

plt.show()

sns.boxplot(data=df,x=df[" Flow Duration"],color='red')

plt.title("Boxplot of Flow Duration");

sns.boxplot(x=df[" Down/Up Ratio"],data=df,color='k')

plt.title("Boxplot of Down/Up Ratio");

sns.histplot(x=df[" Average Packet Size"],data=df,color='k')

plt.title("histplot of Average Packet Size");

X = df.iloc[:,0:20]

y = df.loc[:,'Protocol']

X

y

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test=train\_test\_split(X,y,test\_size=0.5)

X\_train.head(50)

X\_train.tail(50)

X\_train.columns

y\_train.head(50)

y\_train.tail(50)

from sklearn.svm import SVC

from sklearn.metrics import mean\_squared\_error,r2\_score,accuracy\_score

from sklearn.metrics import classification\_report,confusion\_matrix

support=SVC()

support.fit(X\_train,y\_train)

y\_pred=support.predict(X\_test)

mse=mean\_squared\_error(y\_pred,y\_test)

r2=accuracy\_score(y\_pred,y\_test)

print(mse)

print(r2)

print(classification\_report(y\_pred,y\_test))

print(confusion\_matrix(y\_pred,y\_test))

from sklearn.ensemble import RandomForestClassifier

random = RandomForestClassifier(random\_state=150)

random.fit(X\_train, y\_train)

pred=random.predict(X\_test)

score=accuracy\_score(y\_test,pred)

print(score)

mse=mean\_squared\_error(pred,y\_test)

print(mse)

print(classification\_report(pred,y\_test))

print(confusion\_matrix(pred,y\_test))

from sklearn.tree import DecisionTreeClassifier

tree=DecisionTreeClassifier()

tree.fit(X\_train,y\_train)

y\_p=tree.predict(X\_test)

score1=accuracy\_score(y\_test,y\_p)

score1

mse=mean\_squared\_error(y\_p,y\_test)

print(mse)

print(classification\_report(y\_p,y\_test))

print(confusion\_matrix(y\_p,y\_test))

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

kmeans = KMeans(n\_clusters=3)

labels = kmeans.fit\_predict(X\_train)

silhouette\_avg = silhouette\_score(X\_train, labels)

print("Silhouette Score:", silhouette\_avg)

import matplotlib.pyplot as plt; plt.rcdefaults()

objects = ('Random Forest','DecisionTree','Support Vector',"KMeans")

y\_pos = np.arange(len(objects))

performance = [score,score1,r2,silhouette\_avg]

plt.bar(y\_pos, performance, align='center', alpha=0.5)

plt.xticks(y\_pos, objects)

plt.ylabel('Accuracy')

plt.title('RF vs DT vs SVM vs Kmeans')

plt.show()

import pickle

filename='network.sav'

pickle.dump(kmeans,open('network.sav','wb'))

loaded\_model=pickle.load(open('network.sav','rb'))

**CONCLUSION**

This experimental analysis aims to examine, various resampling methods, to mitigate the class imbalance problem in NIDS using SVM-RBF classifier. The SVM-RBF classifier receives input data from different sampling methods adopted after generating the synthetic data to balance the class distribution. From the results obtained This process yields better accuracy with high detection rate of minority classes. Future research will focus on studying the comparative analysis of various kernels of SVM ,DECISION TREE CLASSIFIER,RANDOM FOREST classifier and KMEANS clustering . In addition, the effect of feature selection to select the subset of original dataset before and after resampling methods can be tested by the proposed approach.

**References:**

[1] Ogwara, N.O., Petrova, K., Yang, M.L. (2022). Towards the development of a cloud computing intrusion detection framework using an ensemble hybrid feature selection approach. Journal of Computer Networks and Communications, 2022: 1-16. https://doi.org/10.1155/2022/5988567  
  
[2] Fu, Y., Du, Y., Cao, Z., Li, Q., Xiang, W. (2022). A Deep Learning Model for Network Intrusion Detection with Imbalanced Data. Electronics, 11(6): 898. https://doi.org/10.3390/electronics11060898  
  
[3] Narisetty, N., Kancherla, G. R., Bobba, B., Swathi, K. (2021). Hybrid Intrusion detection method based on constraints optimized SAE and grid search based SVM-RBF on cloud. International Journal of Computer Networks and Applications, 8(6): 776-787. https://doi.org/10.22247/ijcna/2021/210725  
  
[4] Khan, M.A. (2021). HCRNNIDS: hybrid convolutional recurrent neural network-based network intrusion detection system. Processes, 9(5): 834. https://doi.org/10.3390/pr9050834  
  
[5] Tulasi Bhavani, T., Rao, M.K., Reddy, A.M. (2020). Network intrusion detection system using random forest and decision tree machine learning techniques. Advances in Intelligent Systems and Computing, Rajasthan, India, 637-643. https://doi.org/10.1007/978-981-15-0029-9\_50  
  
[6] Krishna Anne, V.P., Rajasekhara Rao, K. (2017). Standards and analysis of intrusion detection-based system: A comparative study. Ponte, 73(2): 87-97. https://doi.org/10.21506/j.ponte.2017.2.7  
  
[7] Jadhav, A.D., Pellakuri, V. (2019). Performance analysis of machine learning techniques for intrusion detection system. In 2019 5th International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, India, pp. 1-9. https://doi.org/10.1109/ICCUBEA47591.2019.9128917   
  
[8] Zhang, Y.Z. (2018). Deep generative model for multi-class imbalanced learning. Open Access Master's Theses. https://doi.org/10.23860/thesis-zhang-yazhou-2018  
  
[9] Chawla, N.V. (2009). Data mining for imbalanced datasets: An overview. In Data Mining and Knowledge Discovery Handbook, New York, NY, pp. 875-886. https://doi.org/10.1007/978-0-387-09823-4\_45  
  
[10] Bagui, S., Li, K. (2021). Resampling imbalanced data for network intrusion detection datasets. Journal of Big Data, 8(1): 1-41. https://doi.org/10.1186/s40537-020-00390-x  
  
[11] Krawczyk, B., Galar, M., Jeleń, Ł., Herrera, F. (2016). Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy. Applied Soft Computing, 38: 714-726. https://doi.org/10.1016/j.asoc.2015.08.060  
  
[12] Xu, R., Chen, T., Xia, Y., Lu, Q., Liu, B., Wang, X. (2015). Word embedding composition for data imbalances in sentiment and emotion classification. Cognitive Computation, 7(2): 226-240. https://doi.org/10.1007/s12559-015-9319-y  
  
[13] Munkhdalai, T., Namsrai, O.E., Ryu, K.H. (2015). Self-training in significance space of support vectors for imbalanced biomedical event data. BMC Bioinformatics, 16(7): 1-8. https://doi.org/10.1186/1471-2105-16-S7-S6  
  
[14] Narisetty, N., Kancherla, G.R., Bobba, B., Swathi, K. (2021). Performance analysis of different activation and loss functions of stacked autoencoder for dimension reduction for NIDS on cloud environment. International Journal of Engineering Trends and Technology, 69(4): 169-176. https://doi.org/10.14445/22315381/IJETT-V69I4P224  
  
[15] Vamsi Krishna, K., Swathi, K., Rama Koteswara Rao, P., Basaveswara Rao, B. (2022). A detailed analysis of the CIDDS-001 and CICIDS-2017 datasets. In Pervasive Computing and Social Networking, Salem, India, pp. 619-638. https://doi.org/10.1007/978-981-16-5640-8\_47  
  
[16] Kumaar, M.A., Samiayya, D., Vincent, P.D.R., Srinivasan, K., Chang, C.Y., Ganesh, H. (2021). A Hybrid framework for intrusion detection in healthcare systems using deep learning. Frontiers in Public Health. https://doi.org/10.3389/fpubh.2021.824898  
  
[17] Ustebay, S., Turgut, Z., Aydin, M.A. (2018). Intrusion detection system with recursive feature elimination by using random forest and deep learning classifier. In 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT), Ankara, Turkey, pp. 71-76. https://doi.org/10.1109/IBIGDELFT.2018.8625318  
  
[18] Mokbal, F., Dan, W., Osman, M., Ping, Y., Alsamhi, S. (2022). An efficient intrusion detection framework based on embedding feature selection and ensemble learning technique. The International Arab Journal of Information Technology, 19(2): 237-248. https://doi.org/10.34028/iajit/19/2/11  
  
[19] Tauscher, Z., Jiang, Y., Zhang, K., Wang, J., Song, H. (2021). Learning to detect: A data-driven approach for network intrusion detection. In 2021 IEEE International Performance, Computing, and Communications Conference (IPCCC), Austin, TX, USA, pp. 1-6. https://doi.org/10.1109/IPCCC51483.2021.9679415  
  
[20] Elmasri, T., Samir, N., Mashaly, M., Atef, Y. (2020). Evaluation of CICIDS2017 with qualitative comparison of Machine Learning algorithm. In 2020 IEEE Cloud Summit, Harrisburg, PA, USA, pp. 46-51. https://doi.org/10.1109/IEEECloudSummit48914.2020.00013  
  
[21] Abdulhammed, R., Musafer, H., Alessa, A., Faezipour, M., Abuzneid, A. (2019). Features dimensionality reduction approaches for machine learning based network intrusion detection. Electronics, 8(3): 322. https://doi.org/10.3390/electronics8030322  
  
[22] Ahmed, H.A., Hameed, A., Bawany, N.Z. (2022). Network intrusion detection using oversampling technique and machine learning algorithms. PeerJ Computer Science, 8: e820. https://doi.org/10.7717/peerj-cs.820  
  
[23] Qaddoura, R., Al-Zoubi, A. M., Almomani, I., Faris, H. (2021). A multi-stage classification approach for iot intrusion detection based on clustering with oversampling. Applied Sciences, 11(7): 302. https://doi.org/10.3390/app11073022  
  
[24] Abdulhammed, R., Faezipour, M., Abuzneid, A., AbuMallouh, A. (2018). Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic. IEEE Sensors Letters, 3(1): 1-4. https://doi.org/10.1109/LSENS.2018.2879990  
  
[25] Choraś, M., Pawlicki, M. (2021). Intrusion detection approach based on optimised artificial neural network. Neurocomputing, 452: 705-715. https://doi.org/10.1016/j.neucom.2020.07.138  
  
[26] Yang, H., Xu, J., Xiao, Y., Hu, L. (2023). SPE-ACGAN: A resampling approach for class imbalance problem in network intrusion detection systems. Electronics, 12(15): 3323. https://doi.org/10.3390/electronics12153323  
  
[27] Narisetty, N., Kancherla, G. R., Bobba, B., Swathi, K. (2021). Investigative study of the effect of various activation functions with stacked autoencoder for dimension reduction of NIDS using SVM. International Journal of Advanced Computer Science and Applications, 12(5): 152-161. http://doi.org/10.14569/IJACSA.2021.0120519  
  
[28] Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. International Conference on Information Systems Security and Privacy, 1: 108-116. https://doi.org/10.5220/0006639801080116  
  
[29] Das, B., Krishnan, N.C., Cook, D.J. (2014). Handling imbalanced and overlapping classes in smart environments prompting dataset. Data mining for service, 199-219. https://doi.org/10.1007/978-3-642-45252-9\_12  
  
[30] Khandekar, V.S., Shrinath, P. (2022). Ensemble Model for Multiclass Imbalanced Data Using Cluster Computing of Spark. https://doi.org/10.21203/rs.3.rs-1981706/v1  
  
[31] Govindarajan, M. (2022). Effective intrusion detection system using classifier ensembles. Ingénierie des Systèmes d’Information, 27(1): 151-156. https://doi.org/10.18280/isi.270118