electronics

Review

Advancements and Challenges in Machine Learning: A Comprehensive Review of Models, Libraries, Applications, and Algorithms

Shahid Tufail, Hugo Riggs, Mohd Tariq * and Arif I. Sarwat *

Department of Electrical and Computer Engineering, Florida International University, Miami, FL 33174, USA; stufa001@fiu.edu (S.T.); hrigg002@fiu.edu (H.R.)
* Correspondence: tmohd@fiu.edu (M.T.); asarwat@fiu.edu (A.I.S.)

Abstract: In the current world of the Internet of Things, cyberspace, mobile devices, businesses, social media platforms, healthcare systems, etc., there is a lot of data online today. Machine learning (ML) is something we need to understand to do smart analyses of these data and make smart, automated applications that use them. There are many different kinds of machine learning algorithms. The most well-known ones are supervised, unsupervised, semi-supervised, and reinforcement learning. This article goes over all the different kinds of machine-learning problems and the machine-learning algorithms that are used to solve them. The main thing this study adds is a better understanding of the theory behind many machine learning methods and how they can be used in the real world, such as in energy, healthcare, finance, autonomous driving, e-commerce, and many more fields. This article is meant to be a go-to resource for academic researchers, data scientists, and machine learning engineers when it comes to making decisions about a wide range of data and methods to start extracting information from the data and figuring out what kind of machine learning algorithm will work best for their problem and what results they can expect. Additionally, this article presents the major challenges in building machine learning models and explores the research gaps in this area. In this article, we also provided a brief overview of data protection laws and their provisions in different countries.

Keywords: machine learning; artificial intelligence; neural network; auto encoder; support vector machine; regression; classification; object detection; supervised training; random forest; decision tree

Citation: Tufail, S.; Riggs, H.; Tariq, M.; Sarwat, A.I. Advancements and Challenges in Machine Learning: A Comprehensive Review of Models, Libraries, Applications, and Algorithms. *Electronics* **2023**, *12*, 1789. https://doi.org/10.3390/electronics12081789

Academic Editors: Juan M. Corchado, Byung-Gyu Kim, Carlos A. Iglesias, In Lee, Fuji Ren and Rashid Mehmood

Received: 16 February 2023
Revised: 29 March 2023
Accepted: 5 April 2023
Published: 10 April 2023

1. Introduction

In the modern world, the price of data storage is decreasing, there is a rapid increase in the speed of data processing, and with the increase in the integration of the Internet of Things, a massive amount of data is generated and stored in data repositories around the globe. In addition, the cost of data storage is decreasing, which means that it will become more affordable in the near future. A new subfield of research known as machine learning has been given the opportunity to flourish as a result of the availability of this massive dataset.

If we take a glance around, we can see examples of machine learning in almost every field. For instance, if we are watching a movie or web series, there is a recommender system that will suggest movies similar to the one we are watching. Likewise, if we buy online, it will suggest products that are either similar to the product we bought or products that were sold most frequently alongside the product we bought. This is done with association rule mining. The internet search engine is another application of machine learning that we use on a daily basis; for example, we used it to forecast tourism based on internet search [1]. When we enter a query into the search engine, the list of links returned to us is arranged from most relevant to least relevant to our search. This is accomplished through a ranking system that will be discussed further in this article. The techniques of machine learning are also helpful in the field of cybersecurity [2]. Algorithms such as isolation forests, support vector machines, neural networks, and others have been shown to be very useful in predicting any intrusion into the network. Therefore, we can say that machine learning is in every sector.

The data are the most important component of machine learning. These data can originate from a wide range of areas, including social networks, logs, blogs, and various sensors such as temperature sensors, current sensors, and humidity sensors, among others. The properties of these data can vary quite a bit. For instance, the data from social networks can come in the form of logs, speech data, image data, video data (a sequence of images), and text data. The data from sensors comes in vectors, and their data type can be float, integer, or string. The data can also be saved in the form of lists or some other format; the specifics of how the data are stored are determined by the conventions that are followed by the data storage repository. In a nutshell, we can say that a complete machine-learning framework consists of three phases: data scraping, model building, tuning and testing, and model deployment. Before model building, tasks such as preprocessing, algorithm selection, statistical analysis, and correlation analysis are carried out.

Figure 1 gives a concise overview of the field in which machine learning is now being applied. Machine learning is becoming an increasingly important tool in the banking industry for identifying potentially fraudulent activities. A few of the criteria that play a role in determining whether or not someone is engaging in fraudulent activity include the following: the time of the transaction, the amount involved in the transaction, the geographical location, and information on previous transactions. In a similar vein, the probability that the borrower will fail on their loan may be forecasted if the model is fed with the borrower's previous credit history as well as his behavior with his previous lenders. The energy industry is another industry that makes extensive use of machine learning. Machine learning is extremely important in many aspects of the energy industry, including but not limited to economic dispatch, customer grouping, optimal load scheduling, and the detection of energy theft. In addition, the implementation of the smart grid, which incorporates a significant amount of renewable resources, has resulted in their network being flooded with thousands of sensors. These sensors send every piece of information to the utility server via a communication network, which has resulted in a new challenge for cybersecurity in the smart grid. The smart grid network is susceptible to a wide variety of cyberattacks, the most prevalent of which include false data injection, denial of service attacks, and botnets. The algorithms for machine learning have a very high degree of accuracy when predicting these types of attacks [2–4]. In the field of healthcare, machine learning is utilized to make predictions regarding the early stages of cancer, the prediction of outbreaks, the performance of drugs, etc. In a manner analogous to this, there are other industries like entertainment, e-commerce, autonomous driving, social networks, email service providers, cloud service providers, and others that use machine learning for a variety of activities.

The following are the primary contributions of this paper: (1) an in-depth analysis of the current machine learning technique that is being applied to solve a wide range of classification, regression, and clustering issues. (2) The machine learning engineer can understand the reasoning behind all machine learning algorithms by reading this paper. This will allow them to determine which machine learning algorithms they can apply to their problem based on the type of data they have. (3) In this study, we also studied the current issues in machine learning, which, if not addressed, can deteriorate the performance of any machine learning model. (4) Finally, we provide a method for dealing with the difficulties encountered while developing a machine-learning model.

The rest of the paper is organized as follows. In Section 2, we discussed the types of machine learning, followed by Section 3, which presented the need for data analysis before ML model building and various tools for data analysis and visualization. In Section 4, we presented various machine learning problems. In Section 5, we described the machine learning algorithms for regression, classification, and clustering along with their mathematical models. Section 6 is about the challenges and solutions faced during the training of machine learning models; this is followed by Section 8, which gives an overview of the machine learning tools available for building the model. Section 9 provides current research in machine learning and use cases. Lastly, Section 10 concludes our work.

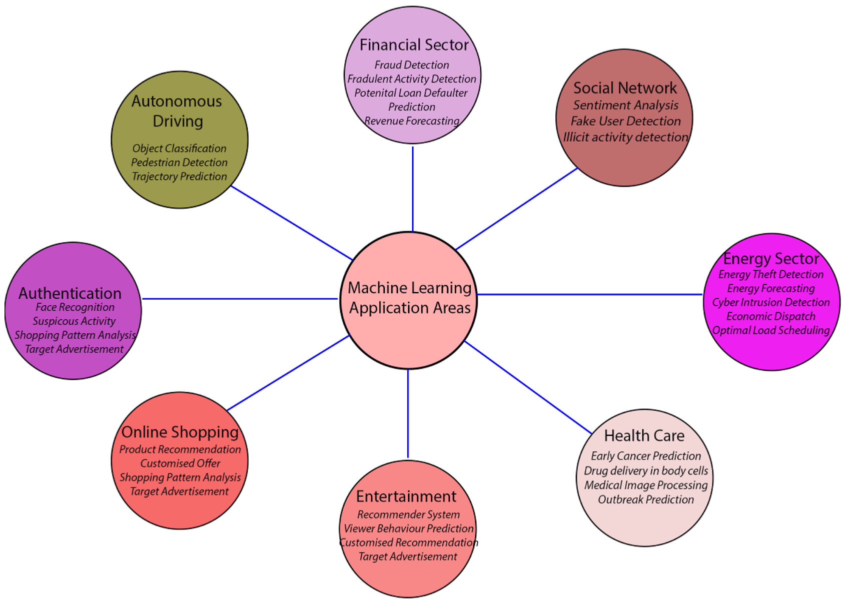

Figure 1. Machine Learning Applications In Different Domains.

2. Types of Machine Learning

Primarily, there are four types of machine learning: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. In supervised learning, the data contains features known as the independent variable and a target variable, or dependent variable. Whereas in unsupervised learning, there is no target variable. Semi-supervised learning is when some of the training data include target labels. In reinforcement learning, data with fewer labels is fed into the model, such as in unsupervised learning. However, we provide punishments and rewards based on the output of the algorithm.

2.1. Supervised Learning

During the training phase of a supervised learning system, the training data will be comprised of inputs coupled with target variables or labels. During the training phase, the algorithm will look for patterns in the data that are associated with the intended outputs and exploit them. When new inputs are introduced after training, a supervised learning algorithm will identify which label the new inputs should be categorized under based on the existing training data. The objective of a supervised learning model is to predict the proper label in case of a classification problem or the estimated value of the output in case of a regression problem for fresh input data given to the model. The purpose of the supervised learning paradigm is to infer a function from a set of data such that

$$y_n{}' = f(x_n, \Theta^*) \tag{1}$$

where y'_n denotes output at n^{th} instance and x_n,θ^* denotes input and set of parameters.

2.2. Unsupervised Learning

Unsupervised learning is used when labels are not present in the data. The data is provided to the machine to find the relationships among the data. Some of the most common unsupervised learning algorithms include clustering algorithms such as K-Nearest Neighbour, K-Means clustering, anomaly detection, such as isolation forest, and association rule learning, such as the apriori algorithm. Dimensionality reduction is another field in which unsupervised learning may be used. The process of limiting the features or combining different features in such a way that they don't lose their characteristics contained within a dataset is referred to as "dimensionality reduction." When it comes to machine learning tasks such as regression or classification, there are frequently an excessive number of variables to work with. This phenomenon, which is referred to as the "curse of dimensionality," describes how the complexity of modeling anything increases in direct proportion to the number of characteristics that it has. Therefore, the dimensionality reduction assists in making the model more efficient and quick to train without losing any critical information or with just a slight deterioration in the model [5]. Some standard dimensionality reduction techniques include Principal Component Analysis (PCA), Single Value Decomposition (SVD), and Linear Discriminant Analysis (LDA).

Autoencoders are a form of unsupervised learning that uses neural networks to perform the task of representation learning. In autoencoders, there are two parts. One part is an encoder, and the other part is a decoder. The primary use of the encoder is to generate more datasets of similar types. There is a design for neural networks that allows for the induction of a bottleneck in the network, which compels a condensed knowledge representation of the initial input. This compression and subsequent reconstruction would be an extremely tough task to perform if each of the input features could be considered independent of the others. However, if the data contain some structure, such as correlations between the input features, this structure may be learned. As a result, it can be utilized when attempting to force the input through the bottleneck in the network [6]. There are various types of autoencoders, such as variational autoencoders, denoising autoencoders, and Long Short-Term Memory (LSTM) autoencoders, that are discussed in this article [6].

2.3. Semi-Supervised Learning(SSL)

When most labels in the dataset are absent and only a select few are present, it is necessary to apply semi-supervised learning since the dataset only includes a partial quantity of the label. For instance, in voice recognition, there may only be a few examples of each voice; in this case, the SSL method will utilize the examples that are available to label the rest of the data. Another example of this would be classifying texts, such as determining the genre of a movie based on its synopsis. Because there are specific labels that are missing, the outcomes of semi-supervised learning are not as accurate as the results of supervised learning [7].

2.4. Reinforcement Learning

The learner in reinforcement learning is an agent that makes decisions, performs actions in an environment, and is rewarded (or penalized) for those actions as it tries to solve a problem. This kind of learning is known as "action-based learning". Following a series of iterations based on the trial-and-error learning strategy, it should eventually discover the optimal policy, which is the chain of behaviors that results in the greatest overall reward [8].

A comprehensive survey has been done by the authors of ref. [9] to show the applications and challenges of reinforcement learning in robotics. A framework is proposed in ref. [10] for dynamic pricing demand response using reinforcement learning.

3. Exploratory Data Analysis In Machine Learning

Exploratory Data Analysis (EDA) is a crucial tool for machine learning engineers to gain insights from their data. It helps them make informed decisions about their data and build models that accurately reflect it. The raw data may not immediately show patterns, correlations, or abnormalities, but with EDA, data scientists can use various visualizations to uncover hidden insights and better understand the structure of the data. This includes studying the data distribution, identifying outliers and missing values, and exploring relationships between features. A comprehensive explanation of the importance of data analysis and procedure is presented in [11].

The use of EDA in machine learning has gained popularity in recent years, as it helps machine learning scientists understand the data before building a model. By having a clear understanding of the data, they can improve their models and avoid making inaccurate assumptions. EDA can also identify any problems with the data that could affect the accuracy of the model.

EDA's ability to recognize correlations between variables is precious in machine learning. By analyzing these correlations, ML engineers can build more accurate models and determine which features are most important. EDA can also detect outliers and missing values, which can improve the model's accuracy.

For ML engineers, EDA provides a visual representation of the data, making it easier to identify trends, patterns, and abnormalities. It can also help determine the most critical aspects of a machine-learning model by investigating the data's distribution patterns. This saves time and resources by focusing on the most critical aspects while reducing the amount of data used in the model.

Moreover, EDA can be used to evaluate the performance of machine learning models. By visualizing the outcomes, data engineers can identify areas where the model is performing poorly and make improvements to increase accuracy and address potential flaws with the data. Some of the tools and libraries for EDA include:

- Python libraries: Pandas, Numpy, Matplotlib, Seaborn, Plotly
- R libraries: ggplot2, dplyr, tidyr, caret
- Tableau
- Excel
- SAS
- IBM SPSS
- Weka
- Matlab
- Statistica
- Minitab

Table 1 presents a list of the machine learning tools and programming languages. These tools are also used for data visualization.

Table 1. Some of the machine learning tools and supporting programming languages available.

Tool	**Company**	**Open Source**	**Programming Skill Required**	**Reference**
Matlab	MathWorks	No	Moderate	[12]
Microsoft Azure Auto ML	Microsoft, Inc.	No	Low	[13]
Python	Python Software Foundation	Yes	High	[14]
R	Microsoft	Yes	High	[15]
Amazon AWS	Amazon, Inc.	No	Moderate	[16]
IBM SPSS	IBM, Inc.	No	Low	[17]
Weka tool	University of Waikato	Yes	Moderate	[18]
DataRobot	DataRobot, Inc.	No	Low	[19]
Google Cloud AutoML	Google LLC	No	Moderate	[20]
Amazon SageMaker	Amazon, Inc.	No	Low	[21]
KNIME	Privately Held	Yes	Low	[22]
Alteryx	Alteryx, Inc.	No	Low	[23]

4. Machine Learning Problems

Many problems fall under the scope of machine learning; these include regression, clustering, image segmentation and classification, association rule learning, and ranking. These are developed to create intelligent systems that can solve advanced problems that, pre-ML, would require a human to solve or would be impossible without computers. The ability to classify unknown input data is very useful; one common example is natural language programming (NLP). The scope of NLP is large, and in this subject, ML classification can be used to extract the bias of an article or convert speech to text. Therefore, classification is a potent ML technology that many people interact with daily in various ways.

4.1. Classification

Classification is a key technology in ML that attempts to identify some input data into a set of categories correctly. The classification model is trained for binary or multi-class problems. While the binary approach is the simplest and most highly useful, multi-class is also very powerful and enables larger sets of applications derived from classification technology.

Within ML, classification is done by models driven by data. In one such model, the support vector machine (SVM), a hyperplane separates data points and is a very commonly used and powerful classification tool. Neural networks are also commonly used for classification, and they have greater applicability when it comes to image-based classification as compared to SVM. Neural networks can have a slower computing time due to the extensive network and many weights and biases to correct over training epochs. In some real-time-dependent classification problems, the SVM may outperform the neural network if online training is needed to update the model. Classification has its own robust set of metrics; the well-known accuracy metric applies to classification. F1 score Precision, recall sensitivity, and specificity are more advanced computed metrics of the classifier's performance. These metrics are computed considering the actual class and predicted class, true positives, true negatives, false positives, and false negatives. To visualize the performance of a classifier, a confusion matrix can be used.

The act of classifying or identifying an object is very useful in many contexts and is a major application of machine learning. Classification can be applied to all types of data and widely, e.g., to identify events in a time series sequence, to determine if a piece of writing has a bias, or to label the objects in an image. Real-time classification of visual data is a major problem in machine learning and is very relevant in autonomous vehicle systems. A self-driving car equipped with omnidirectional cameras should use machine learning to classify the objects nearby with the utmost accuracy, as the implementation of this technology will have life-and-death consequences. There are different types of classification in machine learning; binary classification is common and simple, in which an object is determined to belong to one of two categories. For example, an email is either delivered to an inbox or labeled as spam and moved to the spam folder. Multi-class classifiers are able to classify an object into three or more classes. A multi-class classifier might be used for identifying the genre of music that a song belongs to.

In an example of classification, the spoken digit to numerical digit classifier is created by the authors, and some results are shown. The spoken digit classifier is widely used by spoken language classifiers, such as those installed on smartphones. While those use more advanced cloud-based artificial intelligence systems, this example is based on a local instance of a three-layer recurrent neural network. The network is trained in a supervised manner on the multi-class problem of 0–9 spoken digit classification. The classifier takes as input the sound waveform and outputs the predicted number. Some of the input data is shown in Figures 2 and 3, where the input waveform data is shown for the spoken digits nine and zero, respectively. The RNN network is trained, and after 80 epochs, the classifier has some success ($\geq$80% accuracy), as shown in the confusion matrix in Figure 4.

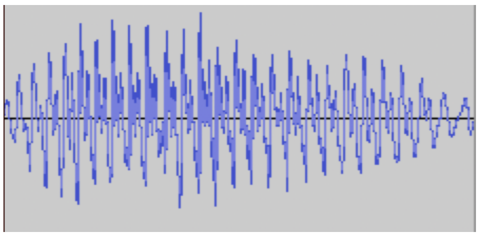
Figure 2. Waveform of spoken digit nine.

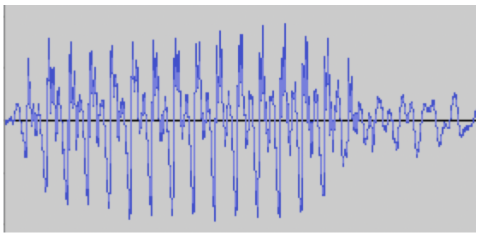
Figure 3. Waveform of spoken digit zero.

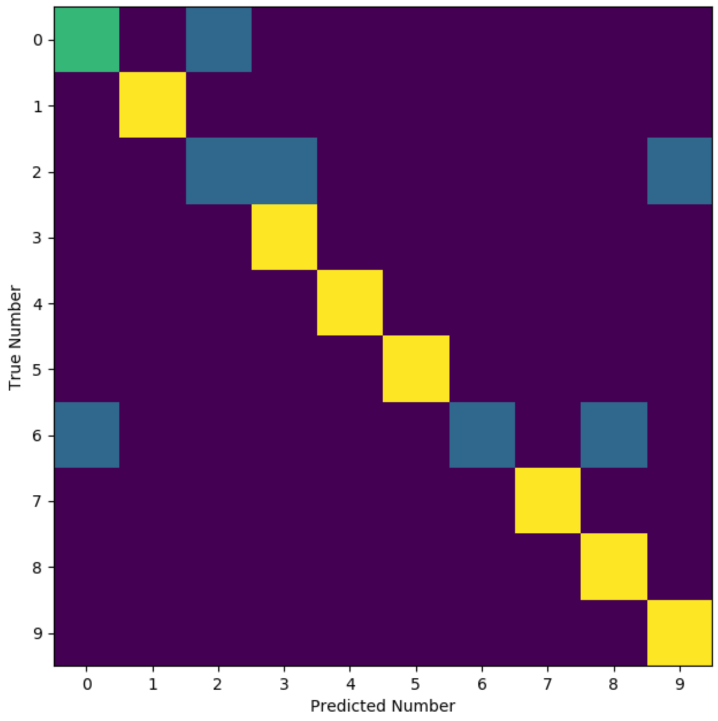

Figure 4. Confusion matrix of spoken digit classifier.

4.2. Regression

The objective of doing a regression analysis is to get an understanding of the relationship that exists between a number of independent variables and a dependent variable, sometimes referred to as the target feature. Regression methods will be used throughout the training process of models intended to anticipate or predict future trends and events. Using labeled training data, these models figure out how the input data relates to the output data. It is, therefore, possible to estimate future patterns, predict outcomes based on data that has not yet been seen, or utilize it to comprehend gaps in data that have already been collected.

The predicted target feature in the regression problem is a constant value. For example, if we know the number of rooms, the size of the house, the neighborhood, the zip code, and the number of people who live there, we can figure out how much the house costs since the labels or outputs are already known, so regression problems are classified as supervised learning. Linear regression is the most basic type of regression technique. The root mean square error (RMSE) value is a common way to determine how accurate a model is. There are also algorithms like decision trees, random forests, support vector regression, and neural networks that are not linear and can be used to solve regression problems [24–27].

4.3. Clustering

Clustering falls under the category of unsupervised learning, as the dataset we provide to the cluster does not contain any labels. Cluster analysis categorizes data items only based on the information inside the data itself that characterizes the objects and their relationships to one another. The key requirement is that the items in one cluster should be comparable or have some other relationship, while the items in other clusters should be distinct from one another or have no such relationship. The clustering is considered to be more effective or distinct when there is a greater degree of resemblance or homogeneity within a group as well as a greater degree of diversity across clusters. A comprehensive survey on clustering algorithms was presented by the authors of ref. [28] in 2005. They discussed various weaknesses and strengths in the clustering algorithms, which include squared error-based, hierarchical clustering, neural networks-based, density-based clustering, and some other clustering algorithms, including fuzzy c-means. Furthermore, it presented the key factors that should be considered prior to selecting the clustering algorithm for a particular set of problems. In this article, K-Means, K-Nearest Neighbor, and DBSCAN algorithms are demonstrated in the next section using the dataset from the sci-kit learn library.

4.4. Image Classification and Segmentation

Image classification can be used to label an image as belonging to a set of categories; more detailed information can be extracted through image segmentation, which identifies specific objects in the image. Within image classification, deep convolutional neural networks are state-of-the-art for high-performance multi-class detection within an image. These network neurons use local receptive fields that input a limited region of the image data. These neurons are typically situated in layers to perform down-convolutions and up-convolutions first, reducing and expanding the image data. This model architecture allows for the abstraction of shapes and the learning of shapes and boundaries. To realize the real-time processing requirements for image classification in self-driving vehicles and other real-time applications, the use of heterogeneous and parallel programming for computing technology can be leveraged. This essentially takes the graphics processing unit, which operates on many more processing cores simultaneously as compared to the standard central processing units. The graphics processing unit is then used to compute the classifications of objects in the image [29].

4.5. Object Detection

Object detection can be a sensor-initiated process in which such sensors capture data about the object. Then that data is run through the machine learning process to identify the object. Based on the amount of data on the object, the ability to correctly detect and categorize the object can be changed. A partial data capture would reduce the ability to detect that object successfully. Our minds are constantly detecting objects and even identifying objects based on only a partial view of the object. For example, by seeing the front of a car, we can know the whole car is also there, or by feeling a keyboard, we may know where the ‘f’ and ‘j’ keys are and, from there, how to type. Furthermore, those skilled at cooking by smell and taste can detect ingredients. Similarly, given partial data from sensors, a machine learning process that is sufficiently complex and well-trained can successfully detect an object. This has huge implications in technology, from fully self-driving vehicles to robotics and to time-series event analysis and classifications, such as real-time network analysis for identifying bad actors on the network.

4.6. Association Rule Learning

This technique of association rule learning draws on patterns in the dataset to extract a set of rules that are generally true about the dataset. This is a highly useful technique for data mining datasets and using the rules to draw inferences about outcomes from new data. These inferences can improve decision-making in planning for events ahead of time. Classically, the example of grocery items is provided. Such associations have been extracted, such that if milk is purchased, bread will likely be purchased by the same customer. Modeling inventory in this way can help stores keep their items stocked appropriately for the demand they expect from consumers. An association rule might be very strong but not 100% true in the dataset, yet such a rule could still prove useful to know. For this reason, association rules can be extracted based on an *interestingness* metric that can allow for strong rules to be elected or rejected based on a threshold.

4.7. Ranking

The ranking creates a hierarchy of importance for a set of information retrieved. ML ranking applications can be prepared through semi-supervised, supervised, or reinforced learning. The ranking is distinguished by type, either pointwise, pairwise, or listwise. Key to these is the application of a metric to score retrieved items; commonly, the mean average precision (MAP) is used.

$$MAP = \sum \frac{\sum_{q=1}^{Q} AveragePrecision(q)}{Q} \tag{2}$$

There are numerous models for ranking, including ANN, SVM, extreme gradient boost, and other custom or hybrid models. In the point-wise approach, ranking is handled as a regression problem. Pairwise pairs of retrieved documents are compared in a binary classification problem. Whereas listwise, the loss is computed on a list of documents' predicted ranks. In pairwise retrieval, binary cross entropy (BCE) is calculated for the retrieved document pairs utilizing y_{ij} is a binary variable of document preference y_i or y_j and $s_{ij} = \sigma(s_i - sj)$ is a logistic function:

$$BCE = -\sum_{i,j=1}^{n} y_{ij} log(s_{ij}) + (1 - y_{ij}) log(1 - s_{ij}) \tag{3}$$

In the listwise ranking, discounted cumulative gain (DCG) or variants of this form are used; it grades the relevance of documents set to evaluate the gain of a document based on its position; the DCG is defined as:

$$DCG_p = \sum_{i=1}^{p} \frac{relevance_i}{log_2(i+1)} \tag{4}$$

The ranking is highly implemented in our modern world, and ranking systems are trained for online systems that take user clicks as input in determining the precision of the ranking of retrieved items. Just some of the applications for ML-based ranking include:

- Web search results ranking
- Writing sentiment analysis
- Travel agency booking availability based on search criteria

4.8. Optimization Problems

Machine learning to automate solutions to optimization problems will search through the solution space for an optimal solution. Evolutionary algorithms are used to do this. The evolutionary algorithm (EA) includes genetic mutation and particle swarm algorithms. The genetic algorithm (GA) will model every solution as an individual in a population. There is a fitness function for evaluating an individual from the population; depending on the fitness score, individuals from a population will be chosen for their genetics when creating the next population. This heuristic leverages the theory of natural evolution into a computer program for automatically finding solutions to optimization problems. Particle swarm optimization (PSO) is an evolutionary algorithm like the GA. However, PSO defines particles instead of individuals and incorporates velocity into these particle objects. One hard requirement of EAs is that a fitness function can be used to evaluate a potential solution. Some other optimization algorithms are Vortices Search Algorithm, Simulated Annealing, Pattern Search, and Artificial Bee Colony, which are discussed later in this article.

5. Machine Learning Model Designs

In this section, the ML model designs are presented with their mathematical implementations. Many models have variants of their formulation, and all variants are not covered in this paper, but the essence or base models are shown along with some of the variant designs.

5.1. Regression and Classification Problems

Both regression and classification problems are predictive models that involve the learning of a mapping function from the inputs to an output approximation. However, classification problems predict a discrete class label, whereas regression problems predict a continuous quantity.

5.1.1. Linear Regression

Linear regression is a type of predictive analysis that is simple and often used. In this kind of regression analysis, the number of independent variables is kept to a minimum to focus on the existence of a linear correlation between the independent variables and the dependent variable, also known as the target variable.

$$y(x) = mx + c \tag{5}$$

where y is the target variable, m is the slope of the line, and c is the y-intercept.

The most significant benefit of using linear regression is that it takes a very short time to train and is a very simple model to implement. It has been observed in the literature that in some cases, linear regression performs the same as another statistical model, such as a decision tree or a random forest, or with just slight inaccuracy. However, the training time of linear regression is significantly lower compared to other models. Therefore, the linear regression machine learning model is often considered a baseline model to compare the performance of other machine learning models. The least squared error loss function used in linear regression is represented as shown in Equation (6).

$$LSE = \frac{1}{2}(f_\theta(x^{(i)} - y^{(i)})^2 \tag{6}$$

where $f_\theta(x^{(i)})$ shows the predicted value for a given point ad $y^{(i)}$ represents the actual value. The cost function is the sum of all the LSE for all the data points in the dataset and can be represented as in Equation (7), where n represents a number of data points.

$$J = \frac{1}{2n}\sum_{i=1}^{n}(f_\theta(x^{(i)} - y^{(i)})^2 \tag{7}$$

A more complex linear regression variant is polynomial regression. It can be represented as shown in Equation (8). However, it is observed that with a higher degree of polynomial equation, the model tends to overfit. To combat the overfitting problem, lasso and ridge regression techniques are used.

$$f(x) = c_0 + c_1x + c_2x^2 + c_3x^3.....c_nx^n \tag{8}$$

5.1.2. Logistic Regression

Like linear regression, logistic regression is the simplest machine-learning model for classification problems. Logistic regression is an example of supervised machine learning and works when the labels are available during the training process. The logistic regression model calculates the weighted sum of the input characteristics (plus a bias term) and then produces the logistic of this result rather than the actual result itself [30]. The input variable is the crucial element in this process. Predicting a binary result is one of the most common applications of logistic regression. When there is a possibility of more than two outcomes coming from a condition, multinomial logistic regression is a useful tool to employ to predict the output class [31]. The logistic function is represented as in Equation (9):

$$f(x) = \frac{1}{1+e^{-x}} \tag{9}$$

The output can be defined as $y_{pred} = \begin{cases} 0 & \hat{p} < 0.5 \\ 1 & \hat{p} > 0.5 \end{cases}$.

5.1.3. Decision Tree

To assess the model performance of the classification tree, the gini index and entropy are two measures that are preferred.

$$GINI = \sum_{m=1}^{k} p_{ik}(1 - p_{ik}) \tag{10}$$

$$Entropy = -\sum_{m=1}^{k} p_{ik} \log p_{ik} \tag{11}$$

Decision trees are also used for a regression problem. A process called binary recursive partitioning is used to build a regression tree. This is an iterative process that divides the data into partitions or branches and divides each partition into smaller groups as the method goes up each branch. First, all the records in the training set are put into the same partition. Using every possible binary split on every column, the algorithm allocates data to the first two partitions or branches. Once all of the data has been allocated, the process repeats. This method selects partitioning that results in the least overall increase in total squared residuals from the mean in both subgroups. For the reason that the tree is developed from the training set, a completely grown-up tree will typically have issues with over-fitting after it has reached its full potential. Due to this, the tree has to have its branches pruned using the validation set to minimize the effects of overfitting.

$$RSS = \sum_{j=1}^{k}(y_j - f(x_j))^2 \tag{12}$$

5.1.4. Random Forest

Random forest is a complex version of the decision tree. Like a decision tree, it also falls under supervised machine learning. The main idea of random forest is to build many decision trees using multiple data samples, using the majority vote of each group for categorization and the average if regression is performed. The mean importance feature is calculated from all the trees in the random forest and is represented as shown in Equation (13).

$$F_i = \frac{\sum_{j=1}^{n} f_{ij}}{n} \tag{13}$$

where F_i denotes the mean feature importance for all the trees, f_{ij} denotes the feature importance of i in j^{th} tree and n shows the number of trees in the forest.

In the random forest, there is no rule to find the optimal number of trees. However, ref. [32] suggests starting with ten times the number of features, increasing the size of the tree in each iteration, and comparing the accuracy.

5.1.5. Support Vector

Support Vector Machines, or SVMs for short, are a model used in machine learning characterized by their robustness and adaptability. It can conduct linear or nonlinear classificational and regression and even identify outliers with decent accuracy.

The core idea of SVM is to put each feature vector in high-dimensional space and draw an imaginary high-dimensional line [33], known as a hyperplane. The expression of hyperplane is given in Equation (14)

$$w.x - b = 0 \tag{14}$$

$$wx = \sum_{i=1}^{n} w_i x_i \tag{15}$$

where w represents the real valued vector, x is the input feature vector, b is a real number, and n is the number of dimensions of the feature vector. For example, when we consider binary classification with labels -1 and 1, the predicted label can be represented as

$$y = \pm(wx + b) \tag{16}$$

Therefore, the primary objective is to find the optimal value of w and b. Thus, Equation (16) can be translated as

$$f(x) = \pm(w^* x - b^*) \tag{17}$$

where $(w^* x - b^*) >= 1$ when the output is 1 and $(w^* x - b^*) <= -1$ when the output is -1. The difference between these parallel hyperplanes is known as the margin. The main idea is to maximize the margin. The distance between two parallel hyperplanes can be represented as $\frac{2}{|w|}$. Therefore, to maximize the distance between the hyperplanes, w has to be minimized. The vectors that support making the hyperplane are known as support vectors.

However, there are very rare instances where data is linearly separable, so for that SVM uses the kernel to classify non-linearly separable data. There are mainly three types of kernels.

1. Linear Kernel
2. Polynomial Kernel
3. Radial Basis Functions Kernels

The linear kernel works well for linearly separable data, as shown in Figure 5, whereas the polynomial kernel represents kernels with more than one degree and is represented as $f(x) = (w_i x_i)^k$, where k denotes the degree of the polynomial. Radial Basis Function (RBF) kernels are one of the most commonly used kernels in SVM problems. They don't require any prior information about the data and perform well for linearly separable data as well as non-linearly separable data. RBFs are represented as $k(x, y) = exp(-\gamma(|x - y|)^2)$, where $|x - y|$ is the euclidean distance between the vectors and γ is a hyper-parameter equivalent to $\frac{1}{2\sigma^2}$, where σ is a free parameter. γ is a parameter of the RBF kernel that controls the decision region. When gamma is small, the 'curve' of the decision boundary has a very low slope, which results in the decision zone having a relatively wide breadth. There are big regions of decision boundaries surrounding data points when the gamma is big because the 'curve' of the border is likewise high, as shown in Figures 6 and 7.

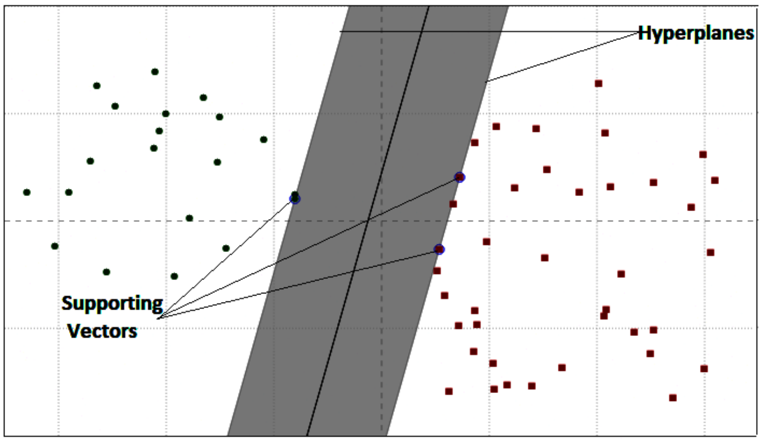

Figure 5. The depiction of the SVM with the linear kernel, supporting vectors, and hyperplanes.

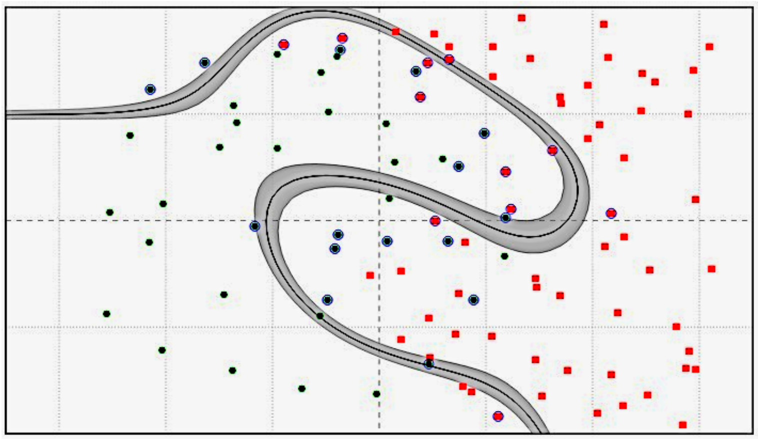

Figure 6. SVM with RBF(γ = 0.1).

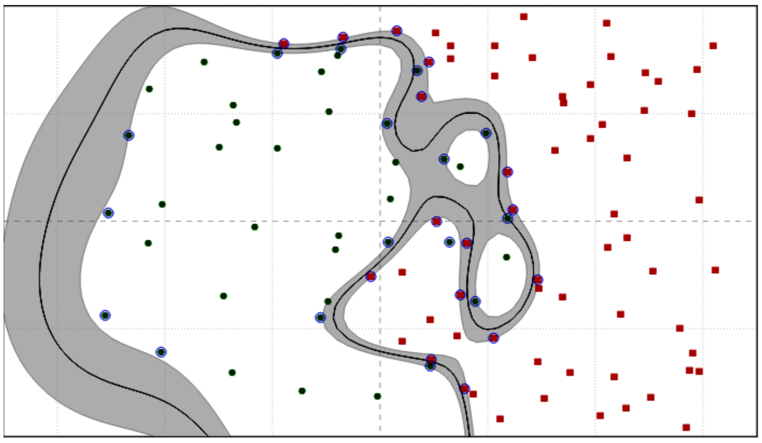

Figure 7. SVM with RBF (γ = 1.0).

5.1.6. K-Nearest Neighbour

The k-nearest neighbors (KNN) algorithm has gained much popularity because it is a basic and easy-to-implement algorithm. It comes under the category of supervised machine learning as the target variable is present in the dataset. It is used to tackle both classification and regression problems. The *k* in the KNN algorithm is the number of neighbors to analyze for predicting the class or estimating the output value. In the classification problem, KNN assigns the category of the majority of its neighbors, whereas for the regression problem, it takes the mean of the target variable of its neighbors. Therefore, to begin, KNN starts to find its neighbor by calculating the distance between the vector for which prediction has to be performed and all the other vectors for which labels are present.

The distance can be calculated with any of the following distance matrices, Euclidean distance, Manhattan distance, Minkowski distance, or Hamming distance. The K-number of neighbors is selected based on their closeness to the vector for prediction. It is recommended to take K as an odd integer value to avoid any ties among the classes. The challenge with KNN is to find an optimal number of neighbors. Therefore, to find an optimum number of neighbors, elbow methods are used. It uses the sum of squares (SSR) to search for the best value of K as shown in Figure 8.

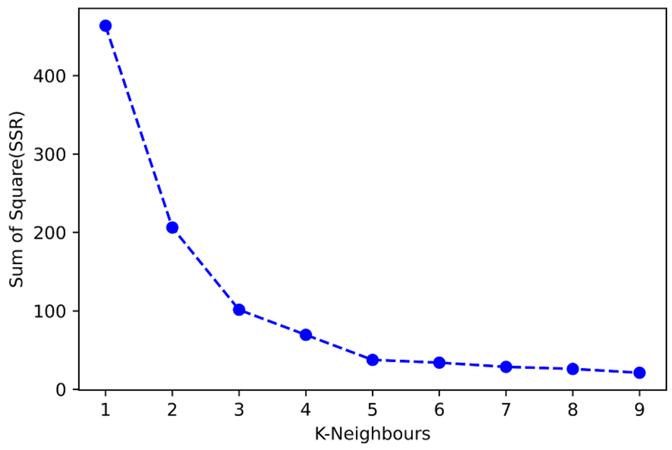

Figure 8. Elbow method.

5.1.7. Neural Networks

Neural networks can be trained to classify based on supervised or unsupervised learning, and specialized neural network models are the leading approach in the visual classification of objects. Every neural network contains at least three layers: (1) Input Layer, (2) Hidden Layer, and (3) Output Layer. These networks' generalizability makes them well-suited for all types of classification problems, from binary to multi-class, with all types of data. Some of the best computer vision models include u-networks, so called based on the 'U' shaped structure of their design. Another type of network is a recurrent neural network. One popular form of this network uses so-called long short-term memory cells in its hidden layers. These types of cells are able to capture or learn relations between events that are occurring over more extended time separations. This is applied to time-series classification. The neural network can also work for numerous regression problems with higher accuracy than linear regression, decision trees, random forests, etc. In this article, we have explained six main types of neural networks.

Feed Forward: This is the simplest form of all neural work. The most basic feed-forward neural network contains one input layer, one hidden layer, and one output layer. The processing of data only occurs in one way. Although the data could travel via a number of the hidden layers. The output of the neuron is represented by Equation (18)

$$z = \sum_{i=1}^{n} (a_i * w_i) + b \tag{18}$$

where z is the output of the neuron, n is the number of neurons in the previous layer, a in the input vector, w is the weight vector, and b is the bias.

Figure 9 illustrates the workings of a feed-forward neural network.

$$h_1 = a_1 * w_{11} + a_2 * w_{21} + a_3 * w_{31} \tag{19}$$

$$h_2 = a_1 * w_{12} + a_2 * w_{22} + a - 3 * w_{32} \tag{20}$$

$$t = h_1 * w_1' + h_2 * w_2' \tag{21}$$

The output of the hidden neuron is the sum of the dot product of the corresponding weight and the input. Similarly the the output,t, if the summation of the dot product of the hidden neuron output and its corresponding weight.

1. Multi Layer Perceptron(MLP)/ Deep Neural Network: Multi layer perceptrons are similar to feed-forward neural network, but in this network there are more than one hidden layer.
   In the backpropogation algorithm, the output from the output layer is compared with the real value, and then weights are adjusted according to the residual. This step is repeated until there is no improvement in the output. Figure 10 shows a typical example of a deep neural network. In the real world, the data is rarely linear, so to handle non-linear data, non-linear activation functions are required. Usually two types of activation functions are applied in the MLP model: one for the hidden layer, which is generally the same for all the layers, but there is no compulsion, and one for the output layer. The most commonly used activation functions are (1) linear activation function, (2) sigmoid, and (3) relu. The output of these activation functions decides whether the neuron will fire or not. For example, in one of the most common activation functions denoted by Equation (11) the neuron is activated or fired only when the output of the neuron is greater than or equal to zero. Figure 11 shows the graphical representation of the Relu activation function. And Figure 12 representation of Sigmoid activation function.

$$f(x) = \begin{cases} x & x >= 0 \\ 0 & x < 0 \end{cases} \tag{22}$$

2. Recurrent Neural Networks (RNN): Recurrent Neural Networks are derived from the feed-forward neural network. These networks have additional connections in their layers that enhance an ANN's ability to learn from sequential data. This can be any form of sequential data, such as numerical time-series data or video and audio. As seen in Figure 13, a hidden layer of RNN cells passes the output to the next layer and to the adjacent cell in series. This enables the network to learn from patterns that appear in sequence. There are a few different formulations for simple RNN networks, such as the following: The W, I, and b are the weights and recurrent connection matrices.

$$h_t = \sigma_h(W_h i_t + X_h h_{t-1} + b_h)$$
$$O_t = \sigma_O(W_o h_t + b_o)$$

   The hidden layer vector is calculated, including the recurrent connections as h_t. The output vector is O_t, and i_t is the input vector.
   The activation functions are σ_h and σ_O [34].
3. Long Short-Term Memory (LSTM): Long short-term memory is an artificial neural network cell type that maintains memory and has input-output and forgets gates as shown in Figure 14. The LSTM is a further developed form of the RNN that incorporates the memory cell. The memory is able to carry forward over various temporal intervals and learn relations between dependent events that are not directly temporally sequential. The LSTM is also an enhanced ANN cell for overcoming the vanishing gradient problem in deep ANNs. The vanishing gradient problem refers to the exponentially smaller error signal that propagates back through the ANN during the supervised learning process as small error values (0,1) are multiplied together. Through its cell formulation, the forget gate does not have an exponential decay factor and thus counters the vanishing gradient problem. ANN using LSTM has shown

record-breaking performance in language processing, such as speech recognition, in chatbots for responding to input text, and machine translation.

The formulation of an LSTM cell is as follows: let W represent weight matrices and U represent the cell's connection matrices. The components of the cell are the forget gate, input gate, output gate to the next recurrent cell in the layer, and memory cell input gate, represented respectively in the equations below.

$$f_t = \sigma_f(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_i(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_o(W_o x_t + U_o h_{t-1} + b_o)$$
$$v_t = \sigma_v(W_v x_t + U_v h_{t-1} + b_v)$$

The cell state is maintained as $c_t = f_t \circ c_{t-1} + i_t \circ v_t$. The output vector that goes to the next layer in the network $h_t = o_t \circ \sigma_h(c_t)$. The variable state of the memory cell is an element of real numbers, as with the network input $c_t \in \mathbb{R}$ and $x_t \in \mathbb{R}$. All vectors except the hidden state, output, and cell input activation vectors range from (0,1). The hidden state h_t is in the range of $(-1, 1)$ [35–37].

4. Autoencoders: Autoencoders are an unsupervised learning technique in which neural networks are utilized for representation learning. There is a neural network architecture that imposes a bottleneck in the network and forces a compressed knowledge representation of the original input as shown in Figure 15. If the input features were independent of one another, this compression and subsequent reconstruction would be very difficult. However, if a structure exists in the data, such as correlations between input features, this structure can be learned and applied when moving the input through the network's bottleneck. An auto-encoder comprises two parts that are encoders and decoders [38].

   The objective of the encoder is to compress the data in such a way that only minimal information is lost, whereas the objective of the decoder is to utilize the compressed information and regenerate the original data. Data compression, image denoising, dimensionality reduction, feature extraction, new data generation, image colorization, and anomaly detection are a few applications of autoencoders.

   The basic autoencoder consists of five parts: input layer, encoder hidden layer, compressed data, also known as a latent vector, decode hidden layer, and output layer. In autoencoders, the number of input features is always equal to the number of output features, and the output layer will always have fewer neurons than the input layer; otherwise, there is the possibility of data replication and there won't be any compression. Mathematically, autoencoders can be expressed as shown in Equation (23)

$$\boldsymbol{x} \rightarrow \underbrace{f(\boldsymbol{x}) \rightarrow \boldsymbol{z}}_{\text{encoding}} \rightarrow \underbrace{g(\boldsymbol{z}) \rightarrow \bar{\boldsymbol{x}}}_{\text{decoding}}. \tag{23}$$

   x is the input vector, $f(x)$ is the encoder function, z is the compressed data, $g(z)$ is the decoder function and $\bar{x}$ is the output. Deep autoencoders are similar to basic autoencoders, but they have multiple hidden layers for encoders and decoders, where the number of hidden layers for both the encoder and the decoder should be equal and should have the same number of neurons. It's similar to a mirror image, where the latent variable vector acts as a mirror. Some common types of autoencoders include denoising the autoencoder, the variational autoencoder, and the convolutional autoencoder [6].

5. Generative Adversarial Network (GAN): Like autoencoders, the primary application of GAN is to generate a new dataset from a given set of data. However, the architecture of GAN is very different from autoencoders. In GAN, there are two parts: the generator and discriminator. These are two different neural networks that run in parallel. The generator starts with random noise and generates a sample. Based on the feedback from the discriminator, the generator trains to generate more realistic

data, and the discriminator learns from the real sample of data and the feedback from the generator. With the improvement in the performance of the generator, the performance of the discriminator deteriorates, and for a perfect generator, the accuracy of the discriminator is 50%. The basic architecture is shown in Figure 16.
In GAN, two loss functions work in parallel, one for the generator and the other for the discriminator. The generator's objective is to minimize the loss, whereas the objective of the discriminator is to maximize the loss. The mathematical formulation for the loss function is shown in Equation (24).

$$min^{gen} max^{dis} E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))] \tag{24}$$

where $D(x)$ is the discriminators' probability that the real data is real, E_x is the expected value of the real data, z is the noise, $G(z)$ is the generator output with z, $D(G(z))$ is discriminator's probability estimate that the fake data is real, E_z is the expected output of the random samples from the generator.

6. Convolutional networks (CNN): CNNs have great applications in computer vision. These networks receive an image as input and perform a sequence of convolution and max pool operations to progressively smaller scales, descending the initial part of the 'U' followed by up-convolution and convolution in sequence to ascend the later part of the 'U'. This model design has proven highly accurate at classifying objects in an image. The convolution network captures spatial and temporal dependencies in an image and performs better at classifying image data when compared to a flattened feed-forward neural network. In a convolutional network, a kernel or matrix passes over the 2-D data array and performs an up or down convolution on the data; alternatively, in a color image, the convolution kernel can be 3-dimensional. These networks are trained in supervised learning and classify images using the softmax classification technique. The ability to automatically do feature extraction and determination is the strength of these models over any competing model. The top image classification AI is based on CNNs, including LeNet, AlexNet, VGGNet, GoogleLeNet, ResNet, and ZFNet.
7. Transformer Neural Network: Transformer networks are more effective at handling long data sequences because they handle the incoming data in parallel, while standard neural networks process the data sequentially. These types of neural networks are heavily used in Natural Language (NLP) for language translation and text summarization. Transformative networks are unique in that they can prioritize different subsets of incoming data while making decisions. This is called the self-attention show in Equation (25). An attention function is a mapping between a query and a collection of key-value pairs to an output, where the query, the key-value pairs, and the result are all vectors. The output is calculated as a weighted sum of the values, where a compatibility function between the query and the corresponding key determines the weight assigned to each value.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{25}$$

This equation computes the attention weights for a given input sequence by calculating the dot product of the query matrix Q and the key matrix K , scaling the result by the square root of the dimension of the key matrix d_k , and applying the softmax function. The resulting attention weights are then applied to the value matrix V to weigh it and compute the output.
The Transformer network consists of an encoder and a decoder. The encoder computes a sequence of hidden representations from the input sequence, utilizing several self-attention layers and feed-forward layers. After receiving the encoder's output, the decoder generates the output sequence by attending to the encoder's hidden representations and employing additional self-attention and feed-forward layers.

Some of this network's significant advantages are the ability to handle long sequences of data efficiently, its parallel processing capabilities, and its effectiveness in natural language processing tasks. However, it also comes with some disadvantages, including the high computational resources s required for training and the need for large amounts of data for effective training.

8. Graph Neural Network: Graph Neural Networks (GNNs) are a class of deep learning models designed to handle data structured as graphs. GNNs are particularly useful for tasks involving relational or structured data, such as social networks, citation networks, and molecular structures. The core principles of Graph Neural Networks (GNNs) and their operation in the context of graph-structured data.

   (a) Information diffusion mechanism: GNNs work by iteratively propagating information between nodes in the graph, simulating a process similar to information diffusion or spreading across the graph.
   (b) Graph processing: In a GNN, each node in the graph is associated with a processing unit that maintains a state and is connected to other units according to the graph's connectivity.
   (c) State updates and information exchange: The processing units (or nodes) update their states and exchange information with their neighbors in the graph. This process is iteratively performed until the network reaches a stable equilibrium, meaning the node states no longer change significantly between iterations.
   (d) Output computation: Once the network reaches a stable equilibrium, the output for each node is computed locally based on its final state. This can be used for various tasks, such as node classification, link prediction, or graph classification.
   (e) Unique stable equilibrium: The GNN's information diffusion mechanism is designed so that a unique stable equilibrium always exists. This ensures that the iterative process of state updates and information exchange eventually converges to a stable solution, making the GNN's output consistent and reliable.

   One simple example of a Graph Neural Network (GNN) is the Graph Convolutional Network (GCN), introduced in [39]. The GCN operates on graph-structured data and updates each node's representation based on its neighbors' representations. The update rule for the hidden representation of a node in a GCN can be expressed as:

$$\mathbf{H}^{(l+1)} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) \tag{26}$$

   Here:
   $\mathbf{H}^{(l)}$ denotes the hidden representation of nodes at the l-th layer, where $\mathbf{H}^{(0)}$ is the input node features. $\tilde{\mathbf{A}}$ is the adjacency matrix of the graph with added self-connections, computed as $\mathbf{A} + \mathbf{I}$, where $\mathbf{A}$ is the original adjacency matrix and $\mathbf{I}$ is the identity matrix. $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}$, with $\tilde{\mathbf{D}}ii = \sum_j \tilde{\mathbf{A}}ij$. $\mathbf{W}^{(l)}$ is the trainable weight matrix at the l-th layer. $\sigma(\cdot)$ is an activation function, such as ReLU. In a GCN, the node representations are updated iteratively for a fixed number of layers or iterations. After the final layer, the node representations can be used for various tasks, such as node classification, link prediction, or graph classification.

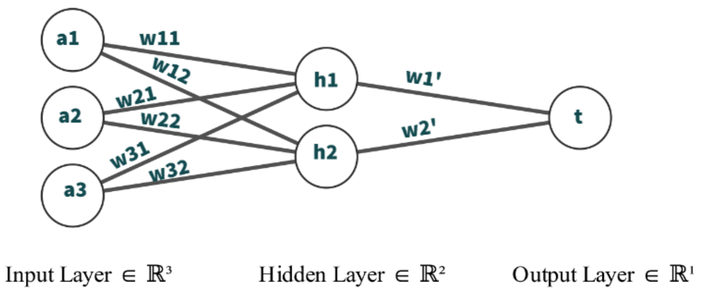
a1
a2
a3
w11
w12
w21
w22
w31
w32
h1
h2
w1'
w2'
t
Input Layer ∈ $\mathbb{R}^3$
Hidden Layer ∈ $\mathbb{R}^2$
Output Layer ∈ $\mathbb{R}^1$

Figure 9. Basic Neural Network Structure.

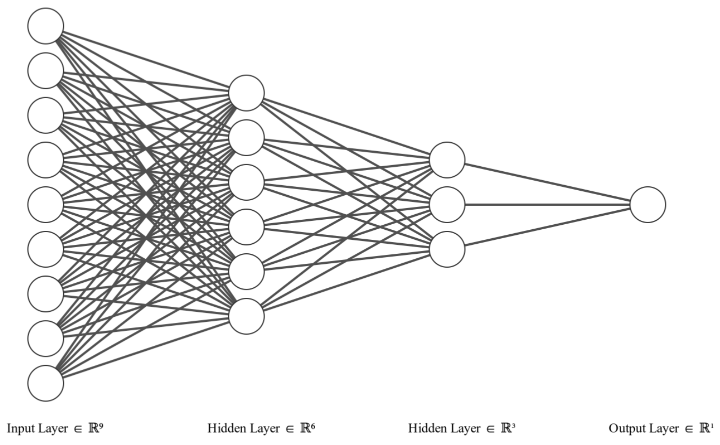
Input Layer ∈ $\mathbb{R}^9$
Hidden Layer ∈ $\mathbb{R}^6$
Hidden Layer ∈ $\mathbb{R}^3$
Output Layer ∈ $\mathbb{R}^1$

Figure 10. Neural Network Structure With Two Hidden Layer.

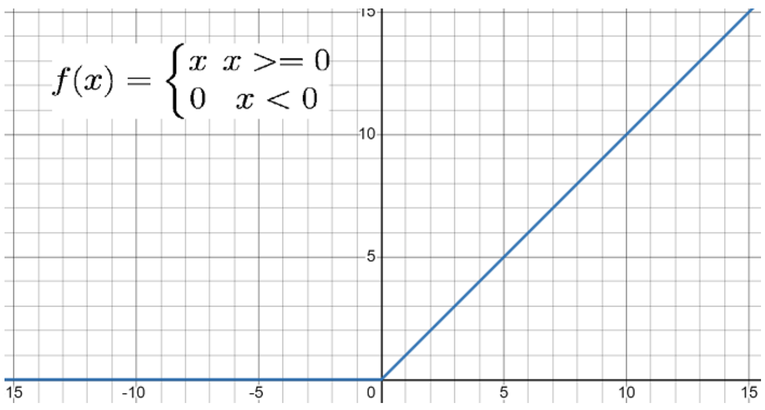

Figure 11. Relu Activation Function.

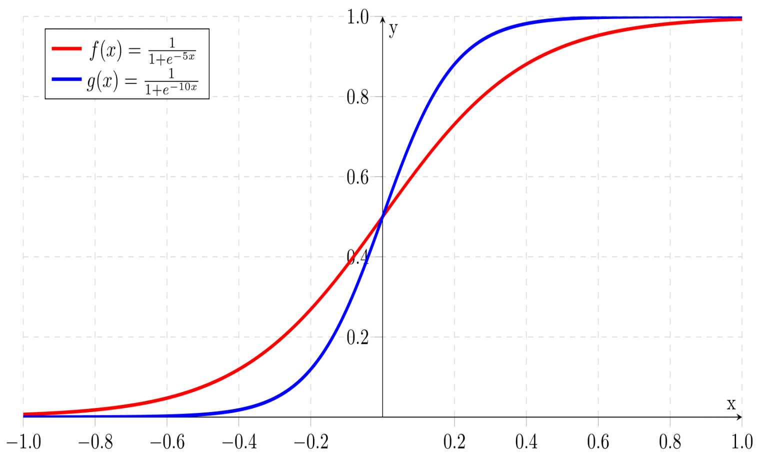

Figure 12. Sigmoid Activation Function.

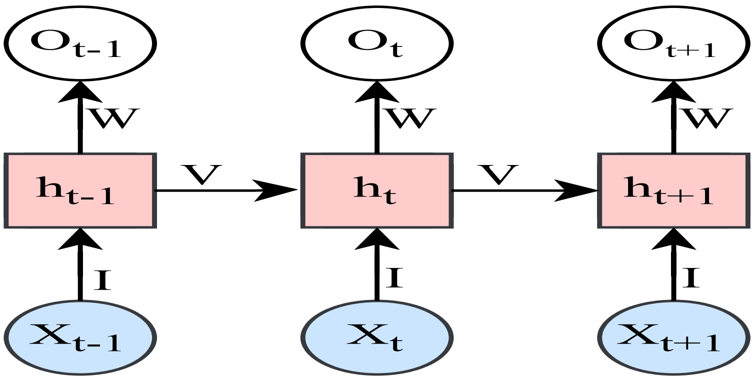

Figure 13. RNN structure.

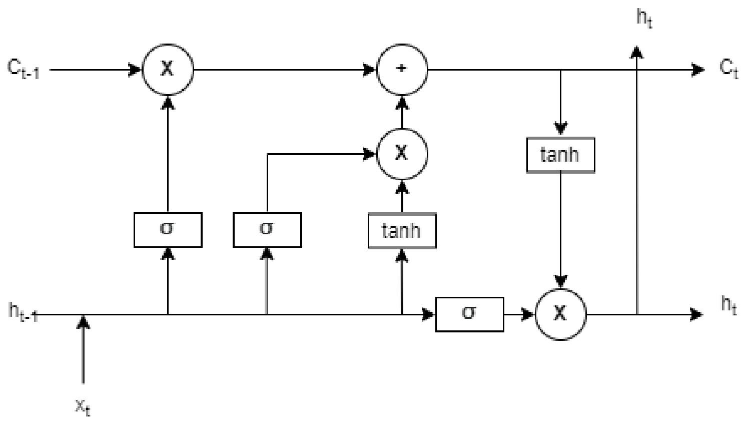

Figure 14. LSTM Cell Design.

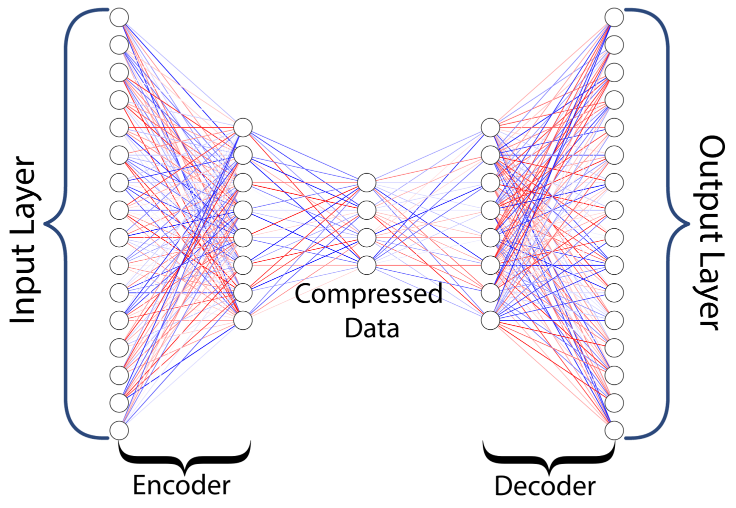

Figure 15. Basic Autoencoder Architecture.

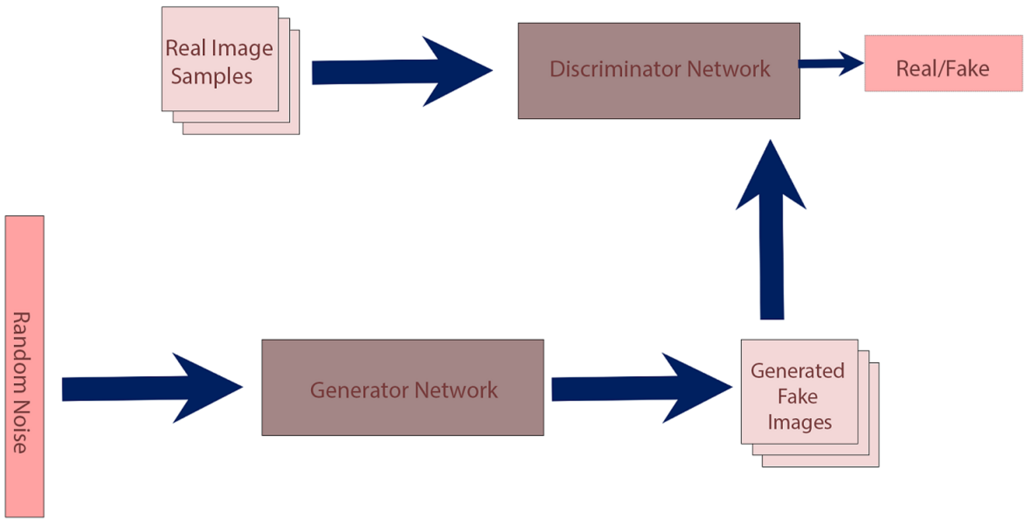

Figure 16. GAN Architecture.

5.2. Clustering

Clustering is an example of unsupervised machine learning. When we have to find out the category but no information regarding the labels is available, clustering is a very beneficial algorithm. The clustering algorithms divide the data points into different groups such that the properties of the points in each group/cluster are similar, that is, intra-cluster properties are maximized but inter-cluster properties are minimized. This mechanism of clustering assists in labeling the data points that are added to the dataset but do not carry any information regarding their labels. The common clustering techniques include K-means clustering, DBSCAN, and OPTICS.

5.2.1. K-Means

K-means clustering is a common top-down clustering algorithm due to its low computational cost and simplicity in implementation.

This step is repeated until there is no change in the centroid position. The demonstration of K-means is shown in Figure 17.

5.2.2. Density Based Clustering

When using density-based clustering, the clusters that are produced are determined by the data points that have the highest density. Outliers are considered to be any points in the data that do not fit into any of the clusters. This sort of clustering suffers from the major shortcoming that it does not function well when there are data of different densities and large dimensions. Two types of density-based clustering algorithms, DBSCAN and OPTICS, are explained in this article.

- Density-based spatial clustering of applications with noise (DBSCAN): DBSCAN starts with any object in the dataset and looks at its neighbors within a certain distance and is mostly denoted by eplison (Eps). If the points around it in that Eps are more than the minimum number of neighbors, then these points are marked as core points. The cluster is formed by core objects and non-core objects. The non-core points are part of the cluster, but they don't extend the cluster, which means they cannot include other non-core points in the cluster. The other objects that are not part of any clusters are marked as outliers. The distance between the points is usually the Euclidean distance. The of DBSCAN is presetned in Figure 18.
  K-means is a centroid-based clustering algorithm, and it starts with the initialization of the number of clusters, followed by assigning a random centroid to each cluster. In the next step, we assign the points to the nearest centroid cluster, and once all the points are assigned, we update the centroid.

- Ordering Points To Identify Cluster Structure(OPTICS): OPTICS is an extended version of DBSCAN. In OPTICS two more hyperparameters are added. (1) Core Distance (2) Reachability Distance. Core distance is defined as the minimum radius value in order to mark the point as a core point, which means it will see if there is any sub-cluster within a cluster that satisfies the minimum neighbor condition. Reachability is the maxima of the core distance and the Euclidean distance between the points. Reachability and core distance are not defined for non-core points. The OPTICS algorithm works well compared to its predecessor, DBSCAN, as it is able to handle data with varying densities.

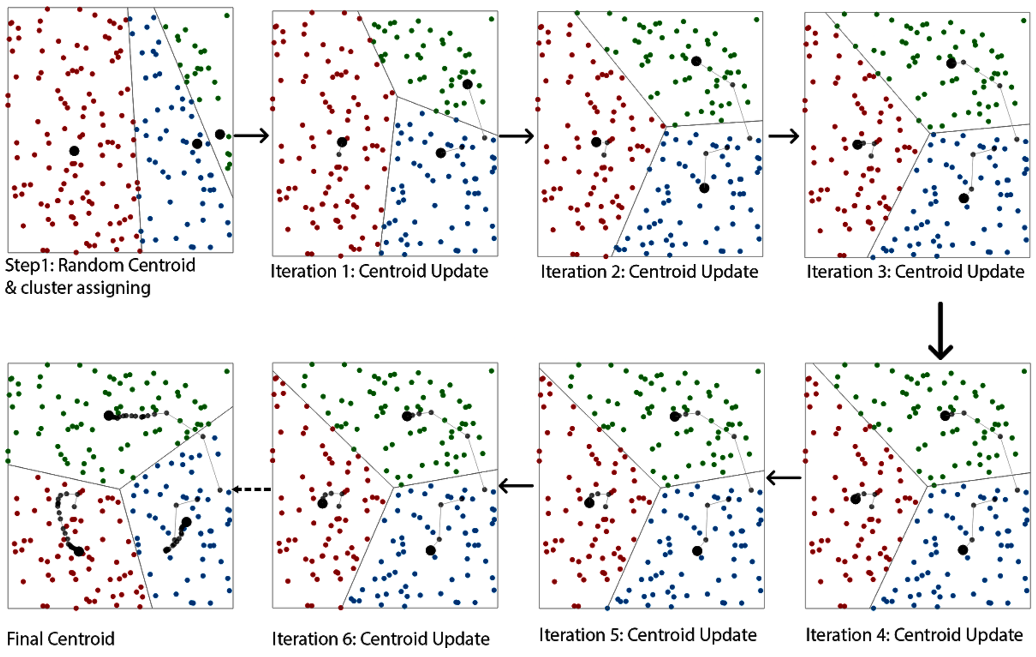
Step1: Random Centroid & cluster assigning
Iteration 1: Centroid Update
Iteration 2: Centroid Update
Iteration 3: Centroid Update
Iteration 4: Centroid Update
Iteration 5: Centroid Update
Iteration 6: Centroid Update
Final Centroid

Figure 17. K-Means Clustering.

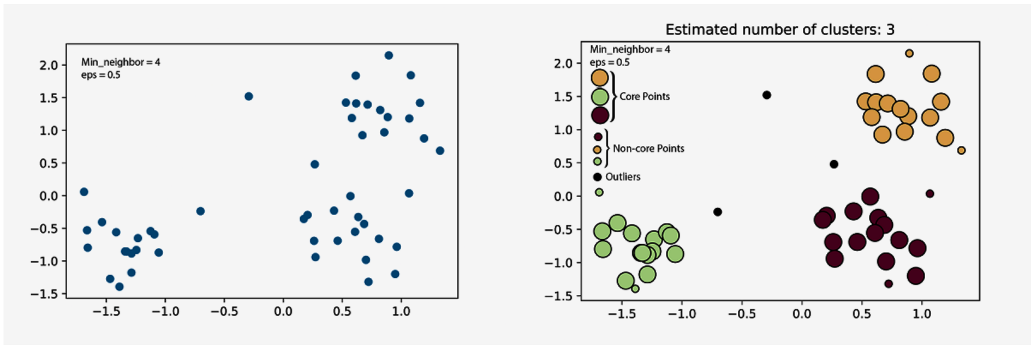

Figure 18. Left: Before Clustering, **Right**: After DBSCAN Clustering.

5.3. *Optimization*

Optimization problems are solved by EA; there are two key implementations of EA: the GA and the PSO. The GA uses the theory of biological natural selection to evolve solutions.

5.3.1. Genetic Algorithm

The GA creates an initial population then loops the following methods:

1. Selection
2. Crossover
3. Mutation
4. Fitness function

The fittest solutions are most likely to be selected for cross-over of genetics, and mutation will alter the genetics of the child. The initial population may be randomized combinations of variables in the solution. Crossover creates new combinations from the parent solutions, showing characteristics from both parents. The mutation is crucial for the introduction of *new genetic material* into the population. In practice, the cross-over loop will produce solutions that find local optimal solutions, whereas mutation will drive the population toward global optimal solutions. In the implementation of GA, there are two rate values α and β, which change the cross-over and mutation rates [40].

5.3.2. Particle Swarm Optimization

The PSO randomly initializes a particle swarm in the decision space, similar to the GA the PSO must be able to evaluate a particle's performance in terms of its solution. Each particle has a speed and position associated with it. The speed and position of a particle are changed in this way so that each particle represents a solution vector that travels through the solution space. A particle's velocity is changed by relating that particle's position vector and velocity to the global best particle. In this way, the swarm will move together through the search space, traversing the solution space, and honing in on the global best solution. Metaheuristics of PSO exist as ant colonies and firefly swarm optimization [41,42].

5.3.3. Vortex Search

The Vortex Search method was developed by Doan and Olmez [43] and inspired by the forms of vortices. VS is a better method for solving single-objective continuous problems.The VS method searches inside the boundaries of a radius adaptively calculated around the vortex's center to achieve a balance between global and local searches. In the early iterations, a global search is conducted while the radius is initially set to a large number; in subsequent iterations, the radius is progressively decreased to enhance the effectiveness of the local search. Hence, the quest for the optimal global solution is increased. The major advantages of VSA are that it handles high-dimensional problems efficiently and is easy to implement as it does not require much parameter tuning. In 2017, a Multi Objection VS(MOVS) was proposed [44]. A few challenges in VSA include getting stuck in a local optimal solution, a slow convergence rate, and low accuracy. To address these issues chaotic vortex search was proposed in 2022 [45].

5.3.4. Simulated Annealing

Simulated Annealing (SA) is a stochastic optimization algorithm inspired by the annealing process in metallurgy, where a material is cooled slowly to reach its optimal crystalline structure [46]. SA explores the search space by accepting solutions based on a probability that depends on the current temperature. As the temperature decreases over time, the algorithm becomes less likely to accept worse solutions, converging to a near-optimal solution [47].

5.3.5. Pattern Search

Pattern Search (PS) is a deterministic optimization algorithm that performs a step-by-step exploration of the search space [48]. The method starts from an initial point and searches in a pattern around it. If a better solution is found, the search moves to the new point, and the pattern is updated. The search continues until a termination criterion is met, such as a maximum number of iterations or when the improvement is below a threshold.

5.3.6. Artificial Bee Colony

Artificial Bee Colony (ABC) is a swarm intelligence-based optimization algorithm that simulates the foraging behavior of honey bees [49]. In ABC, the search agents are "bees" exploring the solution space for "food sources" (candidate solutions). The algorithm consists of three types of bees: employed, onlooker, and scout bees. Employed bees explore the vicinity of their current food source; onlooker bees select and exploit promising food sources based on the information provided by employed bees; and scout bees randomly search for new food sources when a food source is exhausted.

5.4. Hyperparamter Optimization Algorithms

The goal of hyperparameter tuning methods is to optimize the performance of a machine learning model by determining the optimal values for the model's hyperparameters. The most widely used hyperparameter tuning algorithms include:

1. Grid Search: An exhaustive search technique that evaluates all possible combinations of hyperparameter values specified in a predefined grid [50].
2. Random Search: A simple yet effective method that samples random combinations of hyperparameter values from specified distributions, often more efficient than grid search [50].
3. Bayesian Optimization: A model-based optimization technique that uses surrogate models, typically Gaussian processes or tree-structured Parzen estimators, to guide the search for optimal hyperparameters [51].
4. Gaussian Processes: A Bayesian optimization variant that employs Gaussian processes as the surrogate model to capture the underlying function of hyperparameters and the objective [52].
5. Tree-structured Parzen Estimators (TPE): A Bayesian optimization variant that uses TPE to model the probability distribution of hyperparameters conditioned on the objective function values [53].
6. HyperBand: A bandit-based approach to hyperparameter optimization that adaptively allocates resources to different configurations and performs early stopping to improve efficiency [54].
7. Successive Halving: Successive Halving is a hyperparameter optimization approach used to identify the optimal hyperparameter combination for a particular machine learning model. The technique seeks to determine the optimal choice of hyperparameters given a constrained budget, which is often the total number of times the model can be trained and evaluated.
   The primary goal of successive halving is to more efficiently allocate resources by swiftly removing configurations with subpar hyperparameters. This is accomplished by running the model in parallel with various hyperparameter values and lowering the number of analyzed configurations iteratively [55].

6. Major Challenges in Machine Learning and Research Gaps

Machine learning is often difficult to apply or formulate for the problem, requiring iterations of models or variants of models before arriving at a good design for the problem being solved by the ML. Careful attention to detail in the training process must be done by the ML programmer to ensure a good fit of the model to the data. Underfitting or overfitting a model will result in poor performance from the model. Model selection is important and may require some experimentation to compare competing models' performances. Data can also be a challenge, such as the availability of data or missing data. Finally, there is a hardware requirement for ML. To run applications of ML in time to be practical, there must be sufficient computer availability.

6.1. Underfitting

When the training time for a model is low, the model will not learn enough to perform well. For example, if the number of epochs in a neural network training is low, the model will not have sufficient passes over the training data to adjust its weight, and bias variables towards an optimal setting.

Underfitting Prevention Techniques

1. More training data: Increasing the amount of data used in the training process can significantly improve model accuracy [56]. However, it is recommended that the complexity of the model be enhanced in order to get the most out of the data [57].
2. Training Time: If a model has an under-fitting problem, increasing its training time can improve its accuracy [58].
3. Feature Engineering: A poor selection of features or data can mean that regardless of the model chosen, the ML will struggle to learn from the data. Feature engineering can include limiting input data through selection via principle component analysis as well as scaling input data features to the same ranges. Other feature engineering may also be done.
4. Outliers Removal: outliers can produce an unwanted effect on the training of a regression model or yield bad forecast results if taken as true input in a predictor. Filtering outliers can be necessary.
5. Increased Model Complexity: Compared with less complicated models, more sophisticated machine learning models, such as multi-layer perceptrons, can sometimes produce higher levels of accuracy. On the other hand, the training procedure itself needs to be optimized; otherwise, there is a risk of overfitting [59].

6.2. Overfitting

Overfitting is the consequence of training excessively on your dataset and having the model very well fit to the training data; as a result, the model performs worse on test, validation, or real-world data. A typical overfitting curve compared to performance on real data is shown in Figure 19. The problem is identified when the error of the ML on the testing data begins to rise again after having passed a minimum. Over-fitting is a classic problem in ML and can easily occur if the number of training epochs is set very high without a stopping criteria on the performance of the test data set.

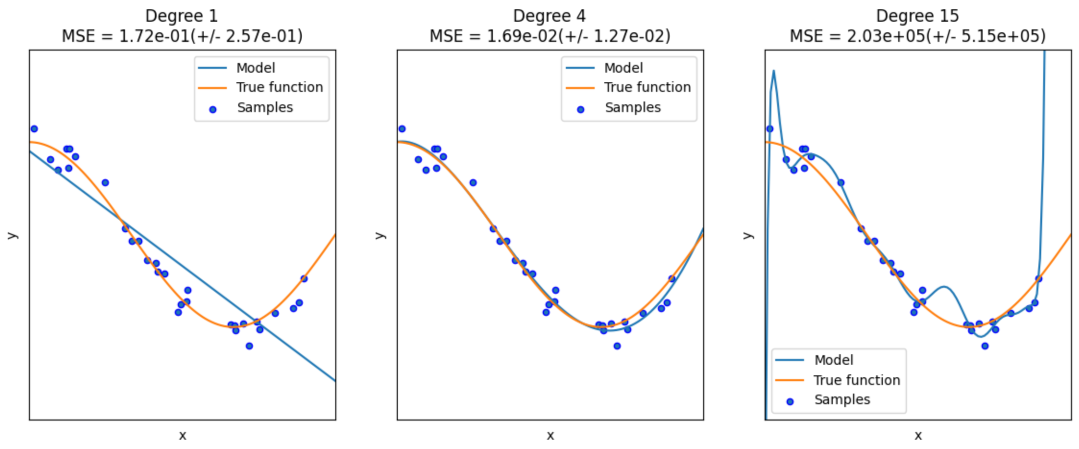

Figure 19. Under, good, and overfitting a polynomial regression model of various degrees.

Overfitting Prevention Techniques

To avoid overfitting and thus train the best ML model that reaches an error minimum on testing data and stops training the model before the error begins to rise again as a result of overfitting on the training data, it is possible to stop training. Monitoring a quantity for early stopping of training is a good approach to prevent overfitting. For example, the error metric can be monitored, and when the minimum is reached, that model is saved and used as it performs the best on test and real data. A typical approach is to monitor validation loss during training and use this metric for early stopping. Another technique is known as cross-validation, and it uses the segmentation of the dataset over various training runs. It moves the target data into the training set and the training data into the target set during training. This maximizes the utility of the training data during training because it reuses the data without fitting it into a small training target set. Rather, the full training data is used in place of the target over consecutive folds. It has the benefit of mixing the target signal and thus limiting the development of a model overfit for a specific target [60–63].

1. More training data
2. Regularisation
3. Cross Validation
4. Early Stopping
5. Dropout Layer
6. Reduce Model Complexity
7. Ensemble Learning

6.3. Model Selection

The machine learning models can be very diverse based on the type of problem. It's not like one model fits all. It has been observed that for the same data, different machine learning algorithms yield different accuracy. Moreover, as the complexity of the model increases, there is an exponential increase in training time and hardware resources. Therefore, for more robust machine learning models, the model accuracy, training time, and resources required should be considered.

In order to find the best model for the problem, prior to model building, data preprocesing and data visualization should be performed in order to see the distribution and other statistical measures. Furthermore, dimensionality reduction and outlier removal should be performed for the most generalized model. Additionally, methods based on information criteria, such as model selection techniques can also be used to compare and choose the best model among a set of candidate models. These criteria balance the goodness of fit of the model with the complexity of the model, penalizing overly complex models that may overfit the data. Some of the widely used information criteria methods are:

1. Akaike Information Criterion (AIC): AIC is a model selection criterion introduced by Akaike [64]. It estimates the relative quality of a set of models and is given by:

$$\mathrm{AIC} = 2k - 2\ln(L) \tag{27}$$

   where k is the number of parameters in the model, and L is the maximized value of the likelihood function for the model.
2. Bayesian Information Criterion (BIC): Also known as Schwarz Information Criterion (SIC), BIC is a model selection criterion introduced by Schwarz [65]. It is similar to AIC but has a stronger penalty for model complexity:

$$\mathrm{BIC} = -2\ln(L) + k\ln(n) \tag{28}$$

   where n is the number of samples, k is the number of parameters in the model, and L is the maximized value of the likelihood function for the model.
3. Minimum Description Length (MDL): MDL is an information-theoretic model selection criterion based on the idea of data compression [66]. It aims to find the model that can represent the data with the shortest description length (the sum of model complexity and data encoding length). MDL works well for dataset model selection. It is better than the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) since it does not require assumptions about data distribution or model prior probability.

6.4. Scalability

Scalability is a major challenge in machine learning because as the size of the dataset and the complexity of models increase, the computational resources and time required for training and inference also grow exponentially. Large-scale datasets and complex models demand high memory capacity, processing power, and storage, which may not be readily available or affordable for all users. Additionally, the need to distribute and parallelize computations across multiple machines or GPUs adds to the complexity of the problem, requiring efficient algorithms and infrastructure to handle communication and synchronization among the computing resources [67].

6.5. Hyperparameter Tuning

Hyperparameter tuning is one of the most challenging task in machine learning due to several reasons. First, the complexity of the hyperparameters themselves can be daunting, as they control various aspects of model architecture and training, such as learning rate, regularization, and network depth. The number and variety of hyperparameters often result in a vast search space, making it difficult to find the optimal combination. Second, the evaluation of each hyperparameter setting can be computationally expensive, as it requires training and validating the model, which can take a significant amount of time and resources, especially for large-scale datasets and complex models. Third, the optimal hyperparameters may differ across datasets, tasks, and even model initialization, making it difficult to generalize the results of tuning for one problem to another. Lastly, many hyperparameter tuning algorithms, such as grid search and random search, can be inefficient as they require testing numerous combinations without considering the interactions between different hyperparameters or leveraging prior knowledge. This has led to the development of more advanced techniques, like Bayesian optimization, to improve the efficiency of hyperparameter tuning. Nevertheless, hyperparameter tuning remains a challenging and time-consuming aspect of machine learning.

6.6. Data availability and Missing Data

Another major bottleneck in machine learning research is a lack of data. In sectors like healthcare, defense, or energy, the data is very sensitive as they have some very confidential information about the stakeholders. Therefore, in such cases, to use the data for machine learning, it has to be anonymized in such a way that there is no exposure of personal information.

The missing data causes model underfitting. Some of the potential causes of missing data are faulty equipment that is responsible for recording data, communication failure that results in failure in sending data to the data storage from where the machine learning model is scraping the data, and power cut-off. Furthermore, there are human causes for missing data, such as an employee who forgot to make the data entry or accidental deletion of data due to an untrained employee.

The missing data problem is tacked on with data imputation. There are many data imputation techniques proposed, such as LSTM-based, mean-based value, sliding window imputation in time series data, and historical mean.

One Model Doesn’t Fit All

The same algorithm cannot be used for all models in machine learning because different models have different characteristics and requirements. The following are a few reasons for this:

1. Different model architectures: Different machine learning models have different architectures and learning algorithms optimized for specific problems.
2. Model objectives: Different models have different objectives, such as minimizing mean squared error, maximizing likelihood, or maximizing accuracy. Therefore, different algorithms are used to optimize each model’s objective function.
3. Model complexity: Different models have varying levels of complexity, and a more complex model requires a different algorithm to optimize its parameters. For example, simple linear models can be optimized using gradient descent, while complex deep learning models require more sophisticated algorithms such as backpropagation.
4. Computational efficiency: Different models have different computational requirements, and some algorithms are better suited for larger datasets or distributed environ-

ments. For example, gradient boosting algorithms can handle large datasets and are suitable for distributed computing, while decision trees can become computationally expensive for larger datasets.

5. Data and problem characteristics: Different machine learning problems have varying data types, feature spaces, noise levels, underlying distributions, and objectives. Therefore, it is important to select or develop a model that is most appropriate for the specific problem at hand.

7. Generalization

While machine learning models have become very good at fitting training data, they often struggle to generalize well to new, unseen data.

Hardware Requirement

Some machine learning models can be very computationally expensive to train as they require a lot of computation during the training process. One such example is image classification. In an image classification problem, as the number of pixels increases, it requires more space and computation power, and when there is continuous object detection throughout a sequence of frames, link timestamp video, or any continuous video, this becomes a more difficult problem to solve. However, to tackle such problem, pretrained models like You Look Only Once(YOLO) and One Shot Object Detection(OSOD), for object detection are very helpful [68–70]. Another technique to handle the limited resource problem is transfer learning. Furthermore, using advanced and high speed hardware such as solid state drives, higher frequency main memory, and a dedicated graphic processing unit locally can increase the performance [71]. Cloud-based model training for large data sets can help alleviate this limitation on hardware [72].

For highly demanding machine learning, there are two strategies for increasing computing power, being parallel computing and distributed computing. Parallel computing utilizes the graphics processing unit of the computer to execute multiple processing threads simultaneously or in *parallel*. This is enabled due to the architecture of a GPU, as they are designed for processing many pixel values simultaneously. More recently, this ability has been leveraged by machine learning algorithm developers in general or scientific applications. GPU-based machine learning is effective in offline applications and applications with real-time constraints. Distributed computing scales the computing power across multiple systems, in other words, cluster computing. A machine learning algorithm can be submitted to the cluster and will be processed; in this case, the application must be less sensitive to latency in the machine learning algorithm [73,74].

8. Machine Learning and the Programming Languages

Across the various programming languages used to implement machine learning, there are some key standouts that are frameworks available across the spectrum of programming languages. Tensorflow, for example, is the ML library available on all the programming tools in this section. Some distributed computing frameworks, such as Apache Spark, also have interfaces for various programming languages and enable accelerated ML implementations.

8.1. Python

Within the machine learning development industry, the choice of Python is common because it has a high level of programming abstraction and some very powerful libraries for writing machine learning programs. This enables ML programmers that use Python to in less time create well designed and powerful models. It is not strongly typed and may not be the best choice for large, object-oriented projects. There is support for all types of ML models in Python; neural networks alone are supported by Keras [75], Tensorflow [76], and PyTorch [77] to name a few. Some libraries have been built off of existing libraries; for example Keras is an abstraction of the Tensorflow library that has since been incorporated into

Tensorflow's own library; another example is science-kit-learn or sk-learn, which provides an even further level of abstraction. The powerful library of sk-learn enables the creation of 'pipelines' which allow the programmer to define the input data, a series of transformations, and a final estimator. A pipeline could, for example, input a 5-dimensional data set, scale the data column-wise between 0 to 1 and train a neural network regression model on the data. Then this pipeline can be saved and loaded as needed, it can be used on new incoming data. Besides powerful ML model development tools, Python also has libraries that are helpful to the ML as auxiliary tools, such as an extensive number library, a plotting library, and a data manipulation library (numpy, matplotlib, and pandas, respectively) [78–80]. Many other libraries support classical ML models in Python, including the Science-Kit libraries [81]. Python also has an open computer vision library for image processing that can be used in ML [82].

8.2. R-Programming Libraries

R-programming has been one of the most popular programming tools for machine learning engineers for quite some time. It was developed by Robert Gentleman and Ross Ihaka at the University of Auckland in the 1990s. It is open-source and there are packages for almost every task, which makes it more lucrative for the programmers. A healthcare data analysis with R is performed in [83]. Some of the machine learning packages available in R includes

- Multivariate Imputation via Chained Sequences (MICE): It is handy tools for data pre-processing like handling of missing data and multivariate data imputation technique as we discussed earlier that missing data is one of the biggest challenge in machine learning models [84].
- Rpart: Its used for recursive partition for classification and regression problem. Rpart is used to build decision trees [85].
- Randomforest: This package is used to implement random forest algorithm for classification and regression task. Additionally it provide relative feature importance for the model [86].
- Caret: The caret (classification and regression training) package in R provides a wealth of resources for creating predictive models from the wide variety of existing R models. The goal of this package is to streamline the process of training and tweaking models using a wide range of modeling approaches. Methods for preparing training data, determining which variables are most important, and visualizing the resulting model are also included.SVM, random forest, bagged trees, neural networks, etc. are just some of the models that may be implemented using caret [87].
- E1071 Naive Bayes, Fourier Transform, Support Vector Machines, Bagged Clustering, etc. are just a few of the algorithms that may be put into action with the help of e1071. The most useful tool that e1071 provides is the use of support vector machines, which allow us to do classification and regression on data that would otherwise be non separable on the given dimension [88].
- Nnet: It is R library that is used to build and train neural network model for classification and regression problem [89].

8.3. Java

The Java programming language is a high-level object-oriented programming language. There are many ML libraries for Java, including the Weka library, which is popular for its graphical user interface to the ML models. To name a few others, there are: Apache Mahout, Deeplearning4j, Mallet, Spark MLlib.

8.4. C++

The C++ language is a high-level object-oriented language that has been around for 37 years and is well known as an incredibly fast and robust programming language. This is the same programming language that our operating systems (Windows, Macintosh, Linux) are written in. This language would require a more trained programmer and more time to create the artificial intelligence code when compared to other languages such as R or Python. The advantages are that C++ will outperform those other languages in its implementations in terms of run time and memory usage. The Tensorflow library for neural networks is implemented in C++. The library MLPack supports other ML models such as Nearest Neighbors, Random Forest, and other classical models, as well as neural networks. Furthermore, there are the OpenNN, Shogun libraries for ML, and OpenCV library for computer vision. Finally (Compute Unified Device Architecture) CUDA is a platform supported on C++ which enables parallel processing on some graphics cards for accelerated run times on ML.

8.5. Research Gaps

Machine learning is undoubtedly a powerful tool that has the potential to explore different sectors from a completely different perspective. There has been a lot of research going on in different sectors to fetch the full power of machine learning.

The main limitation of machine learning is its reliance on data. Despite the fact that it can process large amounts of data in a relatively short amount of time, the data must first be collected and labeled, which can be difficult and time-consuming. Additionally, machine learning models are only as good as the data they are trained on. The model will also be biased or incomplete if the data is biased or incomplete. This can lead to inaccurate or misleading results. However, currently, there is no standard tool or metrics that can ensure the standard of the data by checking for bias and ensuring that the data is free from any kind of contamination prior to feeding it to the algorithms. Therefore, it is highly dependent on the data scientist and his domain knowledge to check that the data is fit for training.

Another domain that still needs to be explored is the challenges in algorithm integration. Standalone machine learning algorithms are not always the best choices; for example, in different time stamps data can have linear relationships and non-linear relationships. These models can be built with python or R programming but require a hefty amount of programming knowledge. However, machine learning tools with algorithm integration can be handy and can significantly reduce the time of model building, with more accurate results.

Machine learning models can be susceptible to malicious attacks. Data contamination attacks can infuse data biasing that can cause a machine-learning model to make incorrect predictions by training the model on biased data. This means that there must be safeguards in place to ensure the security and accuracy of the models. Furthermore, a two-pass filter comprising of a data scrutiny algorithm and an algorithm for the identified problem can reduce the probability of the model getting trained on the contaminated dataset.

Furthermore, it's important to keep personal information safe from any unauthorized access. The government has the responsibility to make rules and guidelines for organizations to follow so that people's personal information is protected and kept private. In Table 2 we presented the data protection laws in various countries [90–99]. Machine learning is dependent on the data, so data anonymization techniques and synthetic data generation techniques are a few other areas of research that need to be explored more.

Table 2. Data Protection Laws in Different Countries.

Countries	**Relevant Act**	**Requirements**
United States	Health Insurance Portability and Accountability Act (HIPAA)	This requires organizations to implement appropriate technical and administrative safeguards to ensure the security and privacy of personal health information(PHI)
	Children's Online Privacy Protection Act (COPPA)	This law applies to the collection of personal information from children under the age of 13 and sets requirements for obtaining parental consent for the collection of such information.
	Electronic Communications Privacy Act (ECPA)	This law governs the interception and disclosure of electronic communications and applies to emails, text messages, and other electronic communications.
	Fair Credit Reporting Act (FCRA)	This law governs the protection of credit information and applies to credit reporting agencies, lenders, and other entities that use credit information for credit decisions.
	California Consumer Privacy Act (CCPA)	This law gives California residents specific rights over their personal information and requires organizations to provide disclosures and notices regarding the collection and use of personal information.
European Union	General Data Protection Regulation (GDPR)	1. Right to access: Individuals have the right to access their personal data and to receive information about how their data is processed. 2. Right to erasure: Individuals have the right to request that their personal data be deleted 3. Right to data portability: Individuals have the right to receive their personal data in a structured and machine-readable format, and to transmit it to another controller. 4. Right to rectification: Individuals have the right to request that inaccurate personal data be corrected. 5. Data protection by design and by default: Organizations must take appropriate technical and organizational measures to ensure that personal data is processed in a secure and confidential manner. 6. Data protection impact assessments: Organizations must conduct data protection impact assessments in certain situations where the processing of personal data is likely to result in high risk to the rights and freedoms of individuals. 7. Notification of data breaches: Organizations must notify individuals and the supervisory authority without undue delay in the event of a personal data breach.
Canada	Personal Information Protection and Electronic Documents Act (PIPEDA)	This governs the collection, use, and disclosure of personal information by organizations in the course of commercial activities. Provisions of PIPEDA (1) Obtaining consent for the collection, use, and disclosure of personal information. (2) Specifying the purposes for which personal information is collected. (3) Limiting the collection of personal information. (4) Ensuring accuracy, protecting personal information with appropriate security measures, and being transparent about policies and practices. (5) Allowing individuals to file a complaint, and ensuring compliance with the law.
	Personal Health Information Protection Act (PHIPA)	This protects the privacy of personal health information while still letting people who are in charge of health information collect, use, and share this information to offer good healthcare. Provisions of PHIPA (1) Obtaining consent for the collection, use, and disclosure of personal health information, specifying the purposes for which personal health information is collected. (2) limiting the collection of personal health information (3) Granting individuals the right to access their personal health information (4) Allowing individuals to file a complaint, and ensuring compliance with the law.

Table 2. *Cont.*

Countries	**Relevant Act**	**Requirements**
Australia	Privacy Act 1988	The Privacy Act 1988 is an Australian law that governs the handling of personal information by Australian government agencies and also private sector organizations. (1) Organizations must obtain consent from individuals for the collection, use, and disclosure of personal information (2) Must only collect personal information that is reasonably necessary for a lawful purpose (3) Individuals have the right to access their personal information held by organizations, and to request correction of any inaccuracies (4) They can also make a complaint to the Office of the Australian Information Commissioner if they believe their privacy rights have been breached.
India	Information Technology (Reasonable security practices and procedures and sensitive personal data or information) Rules, 2011	This provides the guidelines in India that regulate the handling of personal data and information in the information technology sector. Provisions: (1) Organizations must implement reasonable security practices and procedures to protect personal data and information from unauthorized access, alteration, destruction, or disclosure. (2) Organizations must only collect personal data that is necessary for the purpose for for which it was collected and must ensure its accuracy and integrity. (3) Sensitive personal data, such as financial information, biometric information, and health information, is given special protection and must be treated with additional security measures. (4) Organizations must obtain explicit consent from individuals for the collection, use, and disclosure of personal data, including sensitive data. (5) Personal data must be stored in a secure manner, taking into account the nature of the data and the potential risks associated with its loss or unauthorized access. (6) Individuals have the right to access their personal data held by organizations and to request correction of any inaccuracies.

9. Applications of Machine Learning

Machine learning comes with a plethora of applications. However, there are advantages and disadvantages of different algorithms working on the same problem. This behavior is due to the nature of the problem and dataset quality. Table 3 presents a holistic view of various machine learning algorithms along with their advantages and disadvantage. Also, Table 4 shows the different types of applications of machine learning in various domains.

Table 3. Machine Learning Algorithm Comparison.

	Algorithm	Advantages	Disadvantages	References
Supervised Learning Algorithms	Linear Regression	Simple and easy to implement Interpretable Fast training time	Assumes linear relationship between features and target Sensitive to outliers	[100]
	Logistic Regression	Simple and easy to implement Interpretable Provides probability estimates	Assumes linear relationship between features and log-odds of target classes Sensitive to outliers	[101]
	Decision Trees	Interpretable Can handle non-linear relationships Can handle missing values	Prone to overfitting Sensitive to small changes in data	[102]
	Random Forest	Reduces overfitting compared to Decision Trees Handles missing data well	Can be slow to train and predict Less interpretable than Decision Trees	[86]
	Support Vector Machines (SVM)	Effective in high-dimensional spaces	Prone to overfitting in large datasets	[103]
	k-Nearest Neighbors (k-NN)	Simple and easy to implement No training time	Sensitive to feature scaling High memory requirement	[104]
	Neural Networks	Can model complex relationships Suitable for high-dimensional data	Requires large amount of training data Difficult to interpret	[105]
	Gradient Boosting	High predictive accuracy Can handle missing data	Can be slow to train Requires tuning of hyperparameters	[106]
	Naive Bayes	Simple and fast Performs well on small datasets	Assumes independence of features May perform poorly on large or complex datasets	[107]
	Extreme Gradient Boosting (XGBoost)	- High predictive accuracy - Faster and more efficient than traditional Gradient Boosting	Can be prone to overfitting Requires tuning of hyperparameters	[108]
	LightGBM	Fast training and lower memory usage High predictive accuracy	Requires tuning of hyperparameters Can be prone to overfitting	[109]
	Convolutional Neural Networks	Effective for image recognition and other visual tasks, can learn hierarchical representations	Computationally expensive, requires large amounts of data, difficult to interpret	[105,110,111]
	Recurrent Neural Networks	Suitable for sequential data, can learn temporal dependencies	Computationally expensive, prone to vanishing gradients, difficult to interpret	[36,105,110]
	Long Short-Term Memory Networks	Handles long-term dependencies in sequential data, effective for language modeling and other natural language processing tasks	Computationally expensive, requires large amounts of data, prone to overfitting	[110,112]
	Neural Networks	Can model complex relationships Suitable for high-dimensional data	Requires large amount of training data Difficult to interpret	[105]
	Principal Component Analysis (PCA)	Reduces dimensionality of data while preserving most important information, easy to interpret results	Assumes linear relationships between variables, may not perform well on non-linear data	[113]
	t-SNE	Preserves non-linear relationships in high-dimensional data, visually appealing results	Computationally expensive, difficult to interpret results	[114]

Table 3. *Cont.*

	Algorithm	Advantages	Disadvantages	References
Unsupervised Machine Learning Algorithms	Autoencoders	Can learn complex representations of data, useful for dimensionality reduction and anomaly detection	Sensitive to hyperparameters, may overfit on small datasets	[105,110,115]
	Association Rule Mining	Can identify frequent itemsets and interesting rules, useful for market basket analysis and recommendation systems	Limited to binary data, can suffer from the "curse of dimensionality"	[116]
	DBSCAN	Can identify arbitrary shaped clusters, does not require specifying number of clusters beforehand	Sensitive to choice of distance metric and hyperparameters, may not perform well on high-dimensional data	[117]
	Isolation Forest	Useful for outlier detection and anomaly detection, can handle high-dimensional data	May not work well with datasets containing many similar instances, sensitive to choice of hyperparameters	[118]
	Self-Organizing Map (SOM)	Preserves topological properties of input data Good for visualization of high-dimensional data	Requires selection of map size and shape Convergence depends on the learning rate and neighborhood functions	[119]
	Apriori Algorithm	Widely used for association rule learning Easy to implement	- High computational complexity - Inefficient for large datasets	[116]
	OPTICS	Can find clusters of arbitrary shapes Extends DBSCAN by handling varying densities	Higher complexity than DBSCAN Requires tuning of parameters	[120]
	Generative Adversarial Networks	Can generate realistic synthetic data, useful for image and text generation	Prone to mode collapse, can be difficult to train, not suitable for all types of data	[105,110,121]
	Deep Belief Networks	Can learn hierarchical representations of data, effective for unsupervised learning	Computationally expensive, requires large amounts of data, difficult to interpret	[105,110,122]
Reinforcement Learning	Q-learning	Model-free Can learn directly from raw data Converges to optimal policy	Assumes a discrete state and action space Can be slow to converge	[123]
	Deep Q-Network (DQN)	Can handle continuous and high-dimensional state spaces Combines Q-learning with deep neural networks	Requires large amount of training data May be unstable or diverge	[124]
	Policy Gradient Methods	Can handle continuous action spaces Directly optimize the policy	Often suffer from high variance Can be sensitive to step size	[125]

- Energy Sector: As we discussed earlier in this article machine learning has been heavily used in the energy sector. In [126] a comparative study using random forest, support vector, naive Bayes, decision tree, and AdaBoost was performed to predict false data injection attacks in power systems. The experiment performed showed random forest yielded the most accurate results with and without feature selection and is very effective in such types of problems. However, feature selection can affect the performance of machine learning models to a great extent. A similar study was performed in [127] to detect power system faults using a decision tree, K-Nearest Neighbour and SVM. The results showed that the SVM outperformed DT and KNN algorithms with an accuracy of 91.6%. Machine learning is a very handy tool for anomaly detection in different domains, the study performed by the authors of [128] presented a study on autoencoder LSTM, Facebook prophet (time-series forecasting tool), and isolation forest predict anomaly in solar power plants. This study showed AE-LSTM can detect anomalies with high accuracy and can differentiate between anomalous signals and healthy signals.The demand for energy is increasing which is driving more households to install off-grid solar power plants. A study performed in [129] proposed a machine learning framework to maximize the consumption of solar energy using a random forest algorithm. Smart grids are vulnerable to cyberattacks. A detailed survey on various cyberattacks and their defense mechanism has been discussed in [130,131]. Machine learning may be used to identify those attacks. This includes the fake data injection attack, which can have a significant effect on AI-based smart grids [132,133]. An autoregressive integrated moving average(ARIMA) has been proposed in [134] to predict the state of charge for unknown charge and discharge rate. Also in [135] multi-layer perceptron and long short-term based model was proposed to predict battery state of charge. This work shows that the mean squared error for MLP is higher than LSTM model.
- Healthcare: Both the Naive Bayes (NB) classifier and the KNN algorithm can be used for classification problems. In this analysis [136], the authors compared the performance of KNN and Naive Bayes in predicting breast cancer. The correctness of their performance was analyzed using a cross-validation technique. According to the findings, KNN provides the highest degree of accuracy of 97.51% and had the lowest error rate compared to the NB classifier with an accuracy of 96.19%. While in [137] GAN was used to generate synthetic skin lesions. chest x-rays and renal cell carcinoma. The model was trained using 10,000 real images to generate similar synthetic images, and the performance of the models was assessed by training the model on synthetic images and another training using synthetic and real images. In [138] the study was performed on 8066 patients to predict the factors for breast cancer survivors using machine learning. In this study, random forest yielded the most accurate results and showed that the cancer stage classification, tumor size, total auxiliary lymph nodes removal, positive lymph nodes, primary treatment type, and method of diagnosis were the key factors. A comparative study to predict heart disease using logistic regression, KNN, neural network, SVM, naive Bayes, and the decision was performed in [139]. The training was performed on 297 patients sample with 13 attributes. The study showed that the SVM along with feature selection can yield up to 92.37% accuracy.
- Financial sector: It is a sector that generates a humongous amount of data every day. These data vary a lot, it can be transaction data, loan application data, borrowers' personal data, stock data, company information data, and others. Deep learning algorithms can be used to predict stock prices [140]. The base model for the classification problem which is logistic regression can be used to predict the loan defaulter with an accuracy of 81.1% [141]. Similar research done by the authors of [142] showed that the loan default prediction accuracy can be 100% with the AdaBoost model. In the financial sector financial crisis also play a considerable role in economics. Therefore, to predict the financial crisis authors of [143] developed a hybrid model of K-means and genetic ant colony algorithm. To access the model authors used tested using the

Qualitative Bankruptcy dataset [144], Polish dataset [145], and Weislaw dataset [146] data. The proposed model showed an accuracy ranging from 97.55% to 97.93% for different datasets.

- Autonomous Driving: Enhancing the safety of transportation has become feasible as a result of the quickening pace at which artificial intelligence is being incorporated into motor vehicles. It has been demonstrated that safety features such as collision detection, park assists, and lane change assist are particularly beneficial in the prevention of accidents [147–149]. Taking it a step further, machine learning methods for multiple object detection, trajectory prediction, motion, and speed estimation have been crucial in the development of autonomous vehicles. Even if the number of autonomous vehicles is still expanding and they are not entirely self-sufficient, additional research and data are required to demonstrate the effectiveness of such automobiles. Cybersecurity is another issue that needs to be addressed in relation to autonomous vehicles; this is due to the fact that these vehicles are equipped with a large number of sensors that provide driving assistance and that their battery management systems communicate data to the cloud. A detailed survey done in [150] presents various cyber threats and their prevention technique in autonomous vehicles.
- E-Commerce: Another industry that largely relies on machine learning is the e-commerce industry. The forecast of the product demand is done with the help of machine learning. For instance, during the winter months, sales of seasonal products like jackets are at an all-time high, and the demand for products that are used on a daily basis can skyrocket if there is any kind of severe weather alert. This kind of projection contributes to the more efficient management of supply chains [151]. In addition, online retailers often send personalized notifications to customers depending on the users' search histories or the products they have bought in the past. The recommender system is another area in which e-commerce companies invest a lot of money. This system helps retailers propose products that are similar to ones that customers have already purchased or are considering purchasing [152,153]. In the media industry, comparable articles are suggested to users based on their reading histories. An SVM-based recommender system was proposed in ref. [154]. Some of the most prominent challenges in news recommender systems including cold start and data sparsity are discussed in ref. [155]. Furthermore, in the entertainment industry, similar shows are suggested to users depending on whether or not they have seen them or put them on their whitelists. A k-means and KNN algorithm-based recommender system is proposed in ref. [156] Both industries take a similar strategy.
- Satellite Communication: Machine learning (ML) has demonstrated significant potential in improving various aspects of satellite communication systems, making them more efficient, reliable, and adaptable. Some key ML applications in satellite communication include [157–160]:
  - Interference detection and mitigation: ML algorithms can identify and classify interference sources in satellite signals, enabling operators to maintain optimal system performance.
  - Flexible payload configuration: ML techniques, particularly deep learning, can optimize resource allocation within satellite payloads, allowing the system to adapt to changing user demands and improve capacity utilization.
  - Congestion prediction: ML-based forecasting can predict network congestion in satellite communication systems, enabling operators to proactively manage network resources and avoid service disruptions.
  - Spectrum management: ML can assist in dynamic spectrum allocation and interference mitigation, leading to more efficient use of available spectrum and better overall system performance.
  - Anomaly detection: ML algorithms can analyze telemetry data to detect anomalies or potential issues in satellite systems, enabling operators to address problems before they escalate.

– Orbit determination and prediction: ML can improve satellite orbit prediction accuracy, facilitating better planning for satellite-based services, collision avoidance, and space debris tracking.

Table 4. Applications of ML in Different Domains.

Domain	Applications	References
Financial Sector	These applications include automated trading, detection of fraud, financial advisory services for investors, revenue forecasting, and estimate of potential to default on loans.	[161,162]
Social Networks	include the detection of a spammer, link prediction, user classification, friend suggestions, community or cluster identification, and trend analysis.	[163,164]
Energy Sector	Forecasting generation by renewables, economic distribution of energy, energy demand forecasts.	[165–167]
Health Care	Prediction of disease, drug development, medical image processing, outbreak prediction.	[168,169]
Entertainment	Recommender system, Viewer behavior prediction, customized recommendation, target advertisement	[170,171]
Online Shopping	Product Recommendation, Customised offer, shopping pattern analysis, targeted advertisement	[172,173]
Authentication	Face recognition, suspicious activity, shopping pattern analysis, target advertisement	[174–176]
Autonomous Driving	Object Classification, pedestrian detection, trajectory prediction	[177,178]

10. Conclusions

In this article, we have provided a comprehensive analysis of a number of different machine learning algorithms and the domains in which these algorithms may be applied. In the last section of this essay, we went through the different categories of machine learning, and then we moved on to the different categories of problems. Among these problems are regression, classification, object detection, and clustering, amongst others. In the course of our research, we came to the conclusion that there is no such thing as a universal algorithm that fits all; rather, the efficacy of any given model depends not only on the model itself but also on the kinds of data that it uses. In the second section of this article, we discussed the difficulties that can arise when putting machine learning algorithms into practice. These difficulties include the availability of data, the problem of missing data, and the limitations imposed by hardware. We also discussed the precautions that should be taken and the steps that should be taken prior to building a model. And in the third section, we showed the current research that is being done on machine learning in a variety of different fields, such as energy, healthcare, autonomous driving, and e-commerce.Furthermore, we examined the many tools that are now available to fit machine learning algorithms onto data sets.

In general, this article can serve as a stepping stone for those who are just beginning their journey with machine learning, and it can also give experienced users the means to evaluate or construct their own models using the data that they already possess. Furthermore, this article can provide the provision to collect data from the users before using it to build any ML prediction model.

Author Contributions: Conceptualization, S.T. and H.R.; Funding acquisition, A.I.S.; Methodology, S.T., H.R. and M.T.; Investigation, S.T.; Resources, H.R. and M.T.; Supervision, A.I.S.; Writing-original draft preparation, S.T. and H.R.; Writing-review & editing, S.T. and M.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not Applicable.

Data Availability Statement: The study did not report any data.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Li, X.; Li, H.; Pan, B.; Law, R. Machine Learning in Internet Search Query Selection for Tourism Forecasting. *J. Travel Res.* **2021**, *60*, 1213–1231. [CrossRef]
2. Xin, Y.; Kong, L.; Liu, Z.; Chen, Y.; Li, Y.; Zhu, H.; Gao, M.; Hou, H.; Wang, C. Machine Learning and Deep Learning Methods for Cybersecurity. *IEEE Access* **2018**, *6*, 35365–35381. [CrossRef]
3. Handa, A.; Sharma, A.; Shukla, S.K. Machine learning in cybersecurity: A review. *Wires Data Min. Knowl. Discov.* **2019**, *9*, e1306. [CrossRef]
4. Martínez Torres, J.; Iglesias Comesaña, C.; García-Nieto, P.J. Review: Machine learning techniques applied to cybersecurity. *Int. J. Mach. Learn. Cybern.* **2019**, *10*, 2823–2836. [CrossRef]
5. James, D. *Introduction to Machine Learning with Python: A Guide for Beginners in Data Science*, 1st ed.; CreateSpace Independent Publishing Platform: North Charleston, SC, USA, 2018.
6. Jordan, J. Introduction to Autoencoders. 2018. Available online: https://www.jeremyjordan.me/autoencoders/ (accessed on 23 January 2023).
7. Chapelle, O.; Schölkopf, B.; Zien, A. *Semi-Supervised Learning*; The MIT Press: Cambridge, MA, USA, 2006. [CrossRef]
8. Alpaydin, E. *Introduction to Machine Learning*, 3rd ed.; Adaptive Computation and Machine Learning; MIT Press: Cambridge, MA, USA, 2014.
9. Kober, J.; Bagnell, J.A.; Peters, J. Reinforcement learning in robotics: A survey. *Int. J. Robot. Res.* **2013**, *32*, 1238–1274. [CrossRef]
10. Lu, R.; Hong, S.H.; Zhang, X. A Dynamic pricing demand response algorithm for smart grid: Reinforcement learning approach. *Appl. Energy* **2018**, *220*, 220–230. [CrossRef]
11. Ott, R.L.; Longnecker, M.T. *Introduction to Statistical Methods and Data Analysis (with CD-ROM)*; Duxbury Press: Duxbury, MA, USA, 2006.
12. Mathworks.com. Machine Learning with MATLAB. 2022. Available online: mathworks.com (accessed on 22 November 2022).
13. Microsoft. Azure Machine Learning Documentation. 2022. Available online: https://docs.microsoft.com/en-us/azure/machine-learning/ (accessed on 6 February 2023).
14. Python. Python Software Foundation. 2022. Available online: https://www.python.org/psf/ (accessed on 6 February 2023).
15. r–project.org. R: What is R? 2022. Available online: https://www.r-project.org/about.html (accessed on 6 February 2023).
16. Amazon. Cloud Computing Services—Amazon Web Services (AWS). 2022. Available online: https://aws.amazon.com/ (accessed on 6 February 2023).
17. IBM. SPSS Software | IBM. 2022. Available online: https://www.ibm.com/analytics/spss-statistics-software (accessed on 6 February 2023).
18. cs.Waikato.ac.nz. Weka 3—Data Mining with Open Source Machine Learning Software in Java. 2022. Available online: https://www.cs.waikato.ac.nz/ml/weka/ (accessed on 6 February 2023).
19. DataRobot.com. DataRobot AI Cloud—The Next Generation of AI. 2022. Available online: https://www.datarobot.com/ (accessed on 5 February 2023).
20. Gooogle. Cloud AutoML Custom Machine Learning Models. 2022. Available online: https://cloud.google.com/automl (accessed on 4 February 2023).
21. Amazon. Machine Learning—Amazon Web Services. 2022. Available online: https://aws.amazon.com/sagemaker/ (accessed on 5 February 2023).
22. KNIME.com. Open for Innovation. 2022. Available online: knime.com (accessed on 4 February 2023).
23. Alteryx.com. Self-Service Analytics, Data Science & Process Automation | Alteryx. 2022. Available online: alteryx.com (accessed on 4 February 2023).
24. Villegas-Mier, C.G.; Rodriguez-Resendiz, J.; Álvarez Alvarado, J.M.; Jiménez-Hernández, H.; Odry, Á. Optimized Random Forest for Solar Radiation Prediction Using Sunshine Hours. *Micromachines* **2022**, *13*, 1406. [CrossRef]
25. Varma, A.; Sarma, A.; Doshi, S.; Nair, R. House Price Prediction Using Machine Learning and Neural Networks. In Proceedings of the 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, 20–21 April 2018; pp. 1936–1939. [CrossRef]
26. Ho, W.K.; Tang, B.S.; Wong, S.W. Predicting property prices with machine learning algorithms. *J. Prop. Res.* **2021**, *38*, 48–70. [CrossRef]
27. Huynh-Cam, T.T.; Chen, L.S.; Le, H. Using Decision Trees and Random Forest Algorithms to Predict and Determine Factors Contributing to First-Year University Students' Learning Performance. *Algorithms* **2021**, *14*, 318. [CrossRef]
28. Xu, R.; Wunsch, D. Survey of clustering algorithms. *IEEE Trans. Neural Netw.* **2005**, *16*, 645–678. [CrossRef]
29. Gavali, P.; Banu, J.S. Chapter 6—Deep Convolutional Neural Network for Image Classification on CUDA Platform. In *Deep Learning and Parallel Computing Environment for Bioengineering Systems*; Sangaiah, A.K., Ed.; Academic Press: Cambridge, MA, USA, 2019; pp. 99–122. [CrossRef]
30. Géron, A. *Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*; O'Reilly Media: Sebastopol, CA, USA, 2017.
31. Edgar, T.W.; Manz, D.O. *Research Methods for Cyber Security*, 1st ed.; Syngress Publishing: Oxford, UK, 2017.

32. Chapter 11 Random Forests | Hands-On Machine Learning with R. Available online: https://bradleyboehmke.github.io/HOML/random-forest.html (accessed on 4 February 2023).
33. Burkov, A. *The Hundred-Page Machine Learning Book*; Andriy Burkov: Quebec, QC, Canada, 2019.
34. Elman, J.L. Finding Structure in Time. *Cogn. Sci.* **1990**, *14*, 179–211. [CrossRef]
35. Gers, F.A.; Schmidhuber, J.; Cummins, F. Learning to Forget: Continual Prediction with LSTM. *Neural Comput.* **2000**, *12*, 2451–2471. [CrossRef] [PubMed]
36. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
37. Team, G.L. Types of Neural Networks and Definition of Neural Network. 2021. Available online: https://www.mygreatlearning.com/blog/types-of-neural-networks/ (accessed on 4 February 2023).
38. Fengming, Z.; Shufang, L.; Zhimin, G.; Bo, W.; Shiming, T.; Mingming, P. Anomaly detection in smart grid based on encoder-decoder framework with recurrent neural network. *J. China Univ. Posts Telecommun.* **2017**, *24*, 67–73. [CrossRef]
39. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv* **2016**, arXiv:1609.02907.
40. Mallawaarachchi, V. Introduction to Genetic Algorithms—Including Example Code. 2020. Available online: https://www.pinterest.com/pin/introduction-to-genetic-algorithms-including-example-code--656821926880321724/ (accessed on 4 February 2023).
41. Yang, X.S. Chapter 8—Particle Swarm Optimization. In *Nature-Inspired Optimization Algorithms*, 2nd ed.; Yang, X.S., Ed.; Academic Press: Cambridge, MA, USA, 2021; pp. 111–121. [CrossRef]
42. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948. [CrossRef]
43. Doğan, B.; Ölmez, T. A new metaheuristic for numerical function optimization: Vortex Search algorithm. *Inf. Sci.* **2015**, *293*, 125–145. [CrossRef]
44. Özkış, A.; Babalık, A. A novel metaheuristic for multi-objective optimization problems: The multi-objective vortex search algorithm. *Inf. Sci.* **2017**, *402*, 124–148. [CrossRef]
45. Gharehchopogh, F.S.; Maleki, I.; Dizaji, Z.A. Chaotic vortex search algorithm: Metaheuristic algorithm for feature selection. *Evol. Intell.* **2022**, *15*, 1777–1808. [CrossRef]
46. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671–680. [CrossRef]
47. Černý, V. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *J. Optim. Theory Appl.* **1985**, *45*, 41–51. [CrossRef]
48. Torczon, V. On the Convergence of Pattern Search Algorithms. *SIAM J. Optim.* **1997**, *7*, 1–25. [CrossRef]
49. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
50. Bergstra, J.; Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
51. Snoek, J.; Larochelle, H.; Adams, R.P. Practical Bayesian optimization of machine learning algorithms. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 2951–2959.
52. Rasmussen, C.E.; Williams, C.K. *Gaussian Processes for Machine Learning*; MIT Press: Cambridge, MA, USA, 2006.
53. Bergstra, J.; Yamins, D.; Cox, D.D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In Proceedings of the 30th International Conference on Machine Learning (ICML), Atlanta, GA, USA, 16–21 June 2013; pp. 115–123.
54. Li, L.; Jamieson, K.; DeSalvo, G.; Rostamizadeh, A.; Talwalkar, A. Hyperband: A novel bandit-based approach to hyperparameter optimization. *arXiv* **2016**, arXiv:1603.06560.
55. Jamieson, K.; Talwalkar, A. Non-stochastic best arm identification and hyperparameter optimization. *arXiv* **2015**, arXiv:1502.07943.
56. Tufail, S.; Batool, S.; Sarwat, A.I. A Comparative Study Of Binary Class Logistic Regression and Shallow Neural Network For DDoS Attack Prediction. In Proceedings of the SoutheastCon 2022, Mobile, AL, USA, 26 March–3 April 2022; pp. 310–315. [CrossRef]
57. Zhu, X.; Vondrick, C.; Fowlkes, C.C.; Ramanan, D. Do We Need More Training Data? *Int. J. Comput. Vis.* **2016**, *119*, 76–92. [CrossRef]
58. Kim, Y.J.; Choi, S.; Briceno, S.; Mavris, D. A deep learning approach to flight delay prediction. In Proceedings of the 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC), Sacramento, CA, USA, 25–29 September 2016; pp. 1–6. [CrossRef]
59. Bustillo, A.; Reis, R.; Machado, A.R.; Pimenov, D.Y. Improving the accuracy of machine-learning models with data from machine test repetitions. *J. Intell. Manuf.* **2022**, *33*, 203–221. [CrossRef]
60. Tibshirani, R. Regression Shrinkage and Selection via the Lasso. *J. R. Stat. Soc. Ser. Methodol.* **1996**, *58*, 267–288. [CrossRef]
61. Prechelt, L. Early Stopping—But When? In *Neural Networks: Tricks of the Trade*; Orr, G.B., Müller, K.R., Eds.; Springer: Berlin/Heidelberg, Germany, 1998; pp. 55–69. [CrossRef]
62. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
63. Guyon, I.; Elisseeff, A. An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res.* **2003**, *3*, 1157–1182.
64. Akaike, H. A new look at the statistical model identification. *IEEE Trans. Autom. Control.* **1974**, *19*, 716–723. [CrossRef]
65. Schwarz, G. Estimating the Dimension of a Model. *Ann. Stat.* **1978**, *6*, 461–464. [CrossRef]
66. Rissanen, J. Modeling by shortest data description. *Automatica* **1978**, *14*, 465–471. [CrossRef]

67. Bottou, L. Large-Scale Machine Learning with Stochastic Gradient Descent. In Proceedings of the Proceedings of COMPSTAT'2010: 19th International Conference on Computational Statistics, Paris, France, 22–27 August 2010; Lechevallier, Y.; Saporta, G., Eds.; Physica-Verlag HD: Heidelberg, Germany, 2010; pp. 177–186.
68. Jiang, P.; Ergu, D.; Liu, F.; Cai, Y.; Ma, B. A Review of Yolo Algorithm Developments. *Procedia Comput. Sci.* **2022**, *199*, 1066–1073. [CrossRef]
69. Faisal, A.; Kamruzzaman, M.; Yigitcanlar, T.; Currie, G. Understanding autonomous vehicles: A systematic literature review on capability, impact, planning and policy. *J. Transp. Land Use* **2019**, *12*, 45–72. [CrossRef]
70. Fei-Fei, L.; Fergus, R.; Perona, P. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.* **2006**, *28*, 594–611. [CrossRef]
71. Riggs, H.; Tufail, S.; Parvez, I.; Sarwat, A. Survey of Solid State Drives, Characteristics, Technology, and Applications. 2020. Available online: https://www.researchgate.net/publication/339884124_Survey_of_Solid_State_Drives_Characteristics_Technology_and_Applications (accessed on 5 February 2023).
72. Tufail, S.; Qadeer, M.A. Cloud Computing in Bioinformatics: Solution to Big Data Challenge. *Int. J. Comput. Sci. Eng.* **2017**, *5*, 232–236. [CrossRef]
73. Bekkerman, R.; Bilenko, M.; Langford, J. *Scaling up Machine Learning: Parallel and Distributed Approaches*; Cambridge University Press: Cambridge, UK, 2011.
74. Parallel Processing—An Overview. Available online: https://www.sciencedirect.com/topics/computer-science/parallel-processing (accessed on 5 February 2023).
75. Chollet, F. *Deep Learning with Python*; Manning: Shelter Island, NY, USA, 2018; ISBN 9781617294433.
76. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Available online: tensorflow.org (accessed on 6 February 2023).
77. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; pp. 8024–8035.
78. Harris, C.R.; Millman, K.J.; van der Walt, S.J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N.J.; et al. Array programming with NumPy. *Nature* **2020**, *585*, 357–362. [CrossRef]
79. Hunter, J.D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.* **2007**, *9*, 90–95. [CrossRef]
80. pandas-dev/pandas: Pandas. *Zenodo* **2020**, *21*, 1–9. [CrossRef]
81. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
82. Bradski, G. The OpenCV Library. *Dr. Dobb'S J. Softw. Tools Prof. Program.* **2000**, *25*, 120–123.
83. Tufail, S.; Qadeer, M.A. Analysing data using R: An application in healthcare sector. *Int. J. Comput. Sci. Eng.* **2017**, *5*, 249–253. [CrossRef]
84. Wilson, S. MICE Algorithm. 2021. Available online: https://cran.r-project.org/web/packages/miceRanger/vignettes/miceAlgorithm.html (accessed on 14 January 2023).
85. Therneau, T.; Atkinson, B.; Ripley, B. Recursive Partitioning and Regression Trees [R Package Rpart Version 4.1.16]. 2022. Available online: https://rdrr.io/cran/rpart/man/ (accessed on 14 January 2023).
86. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
87. Kuhn, M. Building Predictive Models in R Using the caret Package. *J. Stat. Softw.* **2008**, *28*, 1–26. [CrossRef]
88. Meyer, D.; Dimitriadou, E.; Hornik, K.; Weingessel, A.; Leisch, F.; Chang, C.C.; Lin, C.C.; Meyer, M.D. Package 'e1071'. *R J.* **2019**.
89. Ripley, B.; Venables, W.; Ripley, M.B. Package 'nnet'. *Package Version* **2016**, *7*, 700.
90. Health Insurance Portability and Accountability Act of 1996 (HIPAA). 2022. Available online: https://www.cdc.gov/phlp/publications/topic/hipaa.html (accessed on 6 February 2023).
91. Gaynor, A. Complying with Coppa: Frequently Asked Questions. 2023. Available online: https://www.ftc.gov/business-guidance/resources/complying-coppa-frequently-asked-questions (accessed on 6 February 2023).
92. Electronic Communications Privacy Act of 1986 (Ecpa). Available online: https://bja.ojp.gov/program/it/privacy-civil-liberties/authorities/statutes/1285# (accessed on 6 February 2023).
93. Summary of Your Rights under the Fair Credit Reporting Act. Available online: https://www.consumer.ftc.gov/sites/default/files/articles/pdf/pdf-0096-fair-credit-reporting-act.pdf (accessed on 23 January 2023).
94. Fair Credit Reporting Act—ftc.gov. Available online: https://www.ftc.gov/system/files/ftc_gov/pdf/545A-FCRA-08-2022-508.pdf (accessed on 25 January 2023).
95. de la Torre, L. A guide to the california consumer privacy act of 2018. *SSRN Electron. J.* **2018**. [CrossRef]
96. Mondschein, C.F.; Monda, C. The EU's General Data Protection Regulation (GDPR) in a Research Context. In *Fundamentals of Clinical Data Science*; Kubben, P., Dumontier, M., Dekker, A., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 55–71. [CrossRef]
97. Office of the Privacy Commissioner of Canada. Pipeda Fair Information Principles. 2019. Available online: https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/p_principle/ (accessed on 6 February 2023).

98. Cavoukian, A. Personal Health Information Protection Act—IPC. 2004. Available online: https://www.ipc.on.ca/wp-content/uploads/Resources/hguide-e.pdf (accessed on 8 February 2023).
99. Notani, S. Overview of the Digital Personal Data Protection (DPDP) Bill, 2022—Data Protection—India. 2022. Available online: https://www.mondaq.com/india/data-protection/1255222/overview-of-the-digital-personal-data-protection-dpdp-bill-2022 (accessed on 4 February 2023).
100. Draper, N.; Smith, H. Applied Regression Analysis 2014. Available online: https://www.wiley.com/en-us/Applied+Regression+Analysis%2C+3rd+Edition-p-9780471170822 (accessed on 28 January 2023).
101. Hosmer, D.W.; Lemeshow, S. *Applied Logistic Regression*; John Wiley & Sons: Hoboken, NJ, USA, 2000.
102. Quinlan, J.R. Induction of Decision Trees. *Mach. Learn.* **1986**, *1*, 81–106. [CrossRef]
103. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
104. Cover, T.; Hart, P. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **1967**, *13*, 21–27. [CrossRef]
105. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
106. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
107. Rish, I. An empirical study of the naive Bayes classifier. In Proceedings of the IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, Seattle, WA, USA. 4 August 2001; Volume 3, pp. 41–46.
108. Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
109. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Liu, T.Y. Lightgbm: A highly efficient gradient boosting decision tree. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 3146–3154.
110. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
111. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; Volume 25, pp. 1097–1105.
112. Graves, A.; Schmidhuber, J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Netw.* **2005**, *18*, 602–610. [CrossRef]
113. Jolliffe, I.T. *Principal Component Analysis*; Wiley Online Library: Hoboken, NJ, USA, 2002.
114. van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
115. Hinton, G.E.; Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science* **2006**, *313*, 504–507. [CrossRef]
116. Agrawal, R.; Imielinski, T.; Swami, A. Mining association rules between sets of items in large databases. In Proceedings of the ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, 26–28 May 1993; pp. 207–216.
117. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the 2nd international conference on knowledge discovery and data mining (KDD'96), Portland, OR, USA, 2–4 August 1996; pp. 226–231.
118. Liu, F.T.; Ting, K.M.; Zhou, Z.H. Isolation forest. In Proceedings of the 2012 11th International Conference on Machine Learning and Applications (ICMLA), Boca Raton, FL, USA, 12–15 December 2012; pp. 450–456.
119. Kohonen, T. The self-organizing map. *Proc. IEEE* **1990**, *78*, 1464–1480. [CrossRef]
120. Ankerst, M.; Breunig, M.M.; Kriegel, H.P.; Sander, J. OPTICS: Ordering points to identify the clustering structure. In Proceedings of the ACM Sigmod Record, Philadelphia, PA, USA, 12–17 June 2022; ACM: New York, NY, USA, 1999; Volume 28, pp. 49–60.
121. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; Volume 27, pp. 2672–2680.
122. Hinton, G.E.; Osindero, S.; Teh, Y.W. A fast learning algorithm for deep belief nets. *Neural Comput.* **2006**, *18*, 1527–1554. [CrossRef]
123. Watkins, C.J. *Learning from Delayed Rewards*; University of Cambridge: Cambridge, UK, 1989.
124. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Petersen, S. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
125. Sutton, R.S.; McAllester, D.A.; Singh, S.P.; Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In Proceedings of the Advances in Neural Information Processing Systems, Denver, CO, USA, 20 June 2000; pp. 1057–1063.
126. Kumar, A.; Saxena, N.; Choi, B.J. Machine Learning Algorithm for Detection of False Data Injection Attack in Power System. In Proceedings of the 2021 International Conference on Information Networking (ICOIN), Jeju Island, Republic of Korea, 13–16 January 2021; pp. 385–390. [CrossRef]
127. Goswami, T.; Roy, U.B. Predictive Model for Classification of Power System Faults using Machine Learning. In Proceedings of the TENCON 2019—2019 IEEE Region 10 Conference (TENCON), Kochi, India, 17–20 October 2019; pp. 1881–1885. [CrossRef]
128. Ibrahim, M.; Alsheikh, A.; Awaysheh, F.M.; Alshehri, M.D. Machine Learning Schemes for Anomaly Detection in Solar Power Plants. *Energies* **2022**, *15*, 1082. [CrossRef]
129. Gautam, M.; Raviteja, S.; Mahalakshmi, R. Energy Management in Electrical Power System Employing Machine Learning. In Proceedings of the 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 27–29 November 2019; pp. 915–920. [CrossRef]

130. Tufail, S.; Parvez, I.; Batool, S.; Sarwat, A. A Survey on Cybersecurity Challenges, Detection, and Mitigation Techniques for the Smart Grid. *Energies* **2021**, *14*, 5894. [CrossRef]
131. Tyav, J.; Tufail, S.; Roy, S.; Parvez, I.; Debnath, A.; Sarwat, A. A comprehensive review on Smart Grid Data Security. In Proceedings of the SoutheastCon 2022, Mobile, AL, USA, 26 March–3 Apri 2022; pp. 8–15. . 9764139. [CrossRef]
132. Tufail, S.; Batool, S.; Sarwat, A.I. False Data Injection Impact Analysis In AI-Based Smart Grid. In Proceedings of the SoutheastCon 2021, Mobile, AL, USA, 26 March–3 April 2022; pp. 1–7. [CrossRef]
133. Riggs, H.; Tufail, S.; Khan, M.; Parvez, I.; Sarwat, A.I. Detection of False Data Injection of PV Production. In Proceedings of the 2021 IEEE Green Technologies Conference (GreenTech), Denver, CO, USA, 7–9 April 2021; pp. 7–12. [CrossRef]
134. Khalid, A.; Sundararajan, A.; Sarwat, A.I. An ARIMA-NARX Model to Predict Li-Ion State of Charge for Unknown Charge/Discharge Rates. In Proceedings of the 2019 IEEE Transportation Electrification Conference (ITEC-India), Bengaluru, India, 17–19 December 2019; pp. 1–4. [CrossRef]
135. Khalid, A.; Sundararajan, A.; Acharya, I.; Sarwat, A.I. Prediction of Li-Ion Battery State of Charge Using Multilayer Perceptron and Long Short-Term Memory Models. In Proceedings of the 2019 IEEE Transportation Electrification Conference and Expo (ITEC), Detroit, MI, USA, 19–21 June 2019; pp. 1–6. [CrossRef]
136. Amrane, M.; Oukid, S.; Gagaoua, I.; Ensarİ, T. Breast cancer classification using machine learning. In Proceedings of the 2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT), Istanbul , Turky, 18–19 April 2018; pp. 1–4. [CrossRef]
137. Chen, R.J.; Lu, M.Y.; Chen, T.Y.; Williamson, D.F.K.; Mahmood, F. Synthetic data in machine learning for medicine and healthcare. *Nat. Biomed. Eng.* **2021**, *5*, 493–497. [CrossRef]
138. Ganggayah, M.D.; Taib, N.A.; Har, Y.C.; Lio, P.; Dhillon, S.K. Predicting factors for survival of breast cancer patients using machine learning techniques. *BMC Med Inform. Decis. Mak.* **2019**, *19*, 48. [CrossRef] [PubMed]
139. Li, J.P.; Haq, A.U.; Din, S.U.; Khan, J.; Khan, A.; Saboor, A. Heart Disease Identification Method Using Machine Learning Classification in E-Healthcare. *IEEE Access* **2020**, *8*, 107562–107582. [CrossRef]
140. Nikou, M.; Mansourfar, G.; Bagherzadeh, J. Stock price prediction using DEEP learning algorithm and its comparison with machine learning algorithms. *Intell. Syst. Account. Financ. Manag.* **2019**, *26*, 164–174. [CrossRef]
141. Sheikh, M.A.; Goel, A.K.; Kumar, T. An Approach for Prediction of Loan Approval using Machine Learning Algorithm. In Proceedings of the 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2–4 July 2020; pp. 490–494. [CrossRef]
142. Lai, L. Loan Default Prediction with Machine Learning Techniques. In Proceedings of the 2020 International Conference on Computer Communication and Network Security (CCNS), Xi'an, China, 21–23 August 2020; pp. 5–9. [CrossRef]
143. Uthayakumar, J.; Metawa, N.; Shankar, K.; Lakshmanaprabu, S.K. Intelligent hybrid model for financial crisis prediction using machine learning techniques. *Inf. Syst. -Bus. Manag.* **2020**, *18*, 617–645. [CrossRef]
144. Martin, A.; Uthayakumar, J.; Nadarajan, M. UCI Machine Learning Repository. 2014. Available online: https://archive.ics.uci.edu/ml/datasets/Qualitative_Bankruptcy (accessed on 8 February 2023).
145. Maciej, Z.; Sebastian, K.T.; Jakub, M.T. Polish Companies Bankruptcy Data Data Set; UCI Machine Learning Repository. 2016. Available online: https://archive.ics.uci.edu/ml/datasets/polish+companies+bankruptcy+data (accessed on 8 February 2023).
146. Pietruszkiewicz, W. Dynamical systems and nonlinear Kalman filtering applied in classification. In Proceedings of the 2008 7th IEEE International Conference on Cybernetic Intelligent Systems, London, UK, 9–10 September 2008; pp. 1–6. [CrossRef]
147. LeBeau, P. New Report Shows How Many Accidents, Injuries Collision Avoidance Systems Prevent. 2017. Available online: https://www.cnbc.com/2017/08/22/new-report-shows-how-many-accidents-injuries-collision-avoidance-systems-prevent.html (accessed on 14 January 2023).
148. Nanda, S.; Joshi, H.; Khairnar, S. An IOT Based Smart System for Accident Prevention and Detection. In Proceedings of the 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 16–18 August 2018; pp. 1–6. [CrossRef]
149. Uma, S.; Eswari, R. Accident prevention and safety assistance using IOT and machine learning. *J. Reliab. Intell. Environ.* **2022**, *8*, 79–103. [CrossRef]
150. Kim, K.; Kim, J.S.; Jeong, S.; Park, J.H.; Kim, H.K. Cybersecurity for autonomous vehicles: Review of attacks and defense. *Comput. Secur.* **2021**, *103*, 102150. [CrossRef]
151. Carbonneau, R.; Laframboise, K.; Vahidov, R. Application of machine learning techniques for supply chain demand forecasting. *Eur. J. Oper. Res.* **2008**, *184*, 1140–1154. [CrossRef]
152. Fernández-García, A.J.; Iribarne, L.; Corral, A.; Criado, J.; Wang, J.Z. A recommender system for component-based applications using machine learning techniques. *Knowl.-Based Syst.* **2019**, *164*, 68–84. [CrossRef]
153. Addagarla, S.K.; Amalanathan, A. Probabilistic Unsupervised Machine Learning Approach for a Similar Image Recommender System for E-Commerce. *Symmetry* **2020**, *12*, 1783. [CrossRef]
154. Fortuna, B.; Fortuna, C.; Mladenić, D. Real-Time News Recommender System. In Proceedings of the Machine Learning and Knowledge Discovery in Databases, Ghent, Belgium, 14–18 September 2020; Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 583–586.
155. Raza, S.; Ding, C. News recommender system: A review of recent progress, challenges, and opportunities. *Artif. Intell. Rev.* **2022**, *55*, 749–800. [CrossRef] [PubMed]

156. Ahuja, R.; Solanki, A.; Nayyar, A. Movie Recommender System Using K-Means Clustering AND K-Nearest Neighbor. In Proceedings of the 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 10–11 January 2019; pp. 263–268. [CrossRef]
157. Vázquez, M.Á.; Henarejos, P.; Pappalardo, I.; Grechi, E.; Fort, J.; Gil, J.C.; Lancellotti, R.M. Machine Learning for Satellite Communications Operations. *IEEE Commun. Mag.* **2021**, *59*, 22–27. [CrossRef]
158. Ortiz, F.; Monzon Baeza, V.; Garces-Socarras, L.M.; Vásquez-Peralvo, J.A.; Gonzalez, J.L.; Fontanesi, G.; Lagunas, E.; Querol, J.; Chatzinotas, S. Onboard Processing in Satellite Communications Using AI Accelerators. *Aerospace* **2023**, *10*, 101. [CrossRef]
159. Ferreira, P.V.R.; Paffenroth, R.; Wyglinski, A.M.; Hackett, T.M.; Bilen, S.G.; Reinhart, R.C.; Mortensen, D.J. Reinforcement learning for satellite communications: From LEO to deep space operations. *IEEE Commun. Mag.* **2019**, *57*, 70–75. [CrossRef]
160. Fourati, F.; Alouini, M.S. Artificial intelligence for satellite communication: A review. *Intell. Converg. Netw.* **2021**, *2*, 213–243. [CrossRef]
161. Choi, D.; Lee, K. An Artificial Intelligence Approach to Financial Fraud Detection under IoT Environment: A Survey and Implementation. *Secur. Commun. Netw.* **2018**, *2018*, 5483472. [CrossRef]
162. Lei, H.; Cailan, H. Comparison of Multiple Machine Learning Models Based on Enterprise Revenue Forecasting. In Proceedings of the 2021 Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS), Shenyang, China, 22–24 January 2021.
163. Ganguli, R.; Mehta, A.; Sen, S. A Survey on Machine Learning Methodologies in Social Network Analysis. In Proceedings of the 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 4–5 June 2020.
164. Koggalahewa, D.; Xu, Y.; Foo, E. An unsupervised method for social network spammer detection based on user information interests. *J. Big Data* **2022**, *9*, 1–35. [CrossRef]
165. Chowdhury, B.H.; Rahman, S. A review of recent advances in economic dispatch. *IEEE Trans. Power Syst.* **1990**, *5*, 1248–1259. [CrossRef]
166. Wang, H.; Lei, Z.; Zhang, X.; Zhou, B.; Peng, J. A review of deep learning for renewable energy forecasting. *Energy Convers. Manag.* **2019**, *198*, 111799. [CrossRef]
167. Tan, M.; Yuan, S.; Li, S.; Su, Y.; Li, H.; He, F. Ultra-Short-Term Industrial Power Demand Forecasting Using LSTM Based Hybrid Ensemble Learning. *IEEE Trans. Power Syst.* **2019**, *35*, 2937–2948. [CrossRef]
168. Big Data and Machine Learning in Health Care | Clinical Decision Support | JAMA | JAMA Network. Available online: https://jamanetwork.com/journals/jama/article-abstract/2675024 (accessed on 14 January 2023).
169. Frid-Adar, M.; Diamant, I.; Klang, E.; Amitai, M.; Goldberger, J.; Greenspan, H. GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing* **2018**, *321*, 321–331. [CrossRef]
170. Machine Learning Applications in Drug Development. Available online: https://www.sciencedirect.com/science/article/pii/S2001037019303988 (accessed on 22 January 2023).
171. Yannakakis, G.N.; Maragoudakis, M.; Hallam, J. Preference Learning for Cognitive Modeling: A Case Study on Entertainment Preferences. *IEEE Trans. Syst. Man-Cybern.-Part Syst. Humans* **2009**, *39*, 1165–1175. [CrossRef]
172. Wu, C.; Wang, Y.; Ma, J. Full article: Maximal Marginal Relevance-Based Recommendation for Product Customisation. *Enterp. Inf. Syst.* **2021**, 1–14.
173. Rausch, T.M.; Derra, N.D.; Wolf, L. Predicting online shopping cart abandonment with machine learning approaches. *Int. J. Mark. Res.* **2022**, *64*, 89–112. [CrossRef]
174. CEEOL—Article Detail. Available online: https://www.ceeol.com/ (accessed on 14 January 2023).
175. Verma, K.K.; Singh, B.M.; Dixit, A. A review of supervised and unsupervised machine learning techniques for suspicious behavior recognition in intelligent surveillance system. *Int. J. Inf. Technol.* **2019**, *14*, 397–410. [CrossRef]
176. Li, L.; Mu, X.; Li, S.; Peng, H. A Review of Face Recognition Technology. *IEEE Access* **2020**, *8*, 139110–139120. [CrossRef]
177. Kohli, P.; Chadha, A. Enabling Pedestrian Safety Using Computer Vision Techniques: A Case Study of the 2018 Uber Inc. Self-driving Car Crash. In *Advances in Information and Communication: Proceedings of the 2019 Future of Information and Communication Conference (FICC), San Francisco, CA, USA, 14–15 March 2019*; Arai, K., Bhatia, R., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 261–279.
178. Gupta, A.; Anpalagan, A.; Guan, L.; Khwaja, A.S. Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. *Array* **2021**, *10*, 100057. [CrossRef]