Revista Científica del Centro Universitario de la Guardia Civil
Revista Lógos Guardia Civil

Juan Luis Castro Peña
Professor of Computer Science and Artificial Intelligence.
University of Granada

ARTIFICIAL INTELLIGENCE APPLIED TO SECURITY

ARTIFICIAL INTELLIGENCE APPLIED TO SECURITY

Abstract: This article reviews the state of the art of the most useful Artificial Intelligence techniques for the development of applications in the field of security. We will focus on the aspects that we consider essential in understanding the possibilities, problems, difficulties and aspects to take into account in order to assess possible concrete projects and to approach them in the most effective way.

Resumen: En este artículo se realiza una revisión del estado del arte de las técnicas de Inteligencia Artificial más útiles para el desarrollo de aplicaciones en el ámbito de la seguridad. Nos centraremos en los aspectos que consideramos esenciales para entender las posibilidades, los problemas, las dificultades y las cosas a tener en cuenta para valorar los posibles proyectos concretos, y para abordarlos de la forma más eficaz.

Keywords: Artificial Intelligence, Machine Learning, Deep Learning, AI applications, Security.

Palabras clave: Inteligencia Artificial, Aprendizaje Automático, Aprendizaje Profundo, Aplicaciones de la IA, Seguridad

1.- INTRODUCTION

In the field of *Artificial Intelligence* (AI), we are experiencing an immense explosion of scientific and social interest that has transcended the academic, scientific or technological worlds, even reaching the level of everyday conversations. This has led to a significant increase in the resources devoted to the application of AI. Numerous private companies and public entities are investing human and financial capital to try to incorporate AI into their systems. This is leading to a very positive increase in the development of AI applications in virtually all fields. However, in many cases, the rigour required for the correct incorporation of scientific and technological advances is not being scrupulously followed, potentially leading to a host of problems.

Many of these developments are being carried out by teams that have not studied the subject in sufficient depth. The proliferation of a large number of introductory courses that do not go into the deeper aspects is resulting in the emergence of what could be called "*user-level AI experts",* responsible for the application of this technique, without the relevant training required for any engineering application. Courses are imparted by companies that offer tools for the application of these techniques and whose natural interest is that their services are contracted. Other courses are offered as *unofficial* courses by academic centres that have found a strong market for this subject. In any case, based on their duration and objectives, many of these courses do not address aspects relevant to the correct application of AI. In this way, advanced AI tools are being used to apply them to specific problems by teams that do not understand some of the key points for the correct application of these techniques. A typical case is the use of machine learning algorithms, using libraries that facilitate easy and rapid development, such as the *Deep Learning* libraries of Python or R, or the tools offered by large service companies such as Google, Microsoft, Amazon or IBM. The problem is that these techniques and tools are being applied without understanding the essential aspects of their operation for their correct application, such as the meaning and relevance of the parameters or alternatives in order to choose the most appropriate values in each specific case, or the fundamental aspects to be taken into account for the system to learn correctly, such as the analysis of the datasets used, and the pre-processing, adjustment or validation procedures. They are limited to following user manuals which, given their conception and length, do not address these issues. In these circumstances the result can be misleading, for example, models with a very high success rate during validation tests (even above 95%), but which, when applied in the real world, are far from even coming close to that rate of effectiveness.

The danger is twofold. On the one hand, when it becomes apparent that the system is flawed in actual use (many more than the "laboratory" validation predicted), this can lead to disappointment and frustration, even product failure, and a consequent low return on investment. An illustrative example is the case of the numerous tools that were developed to diagnose and forecast the evolution of COVID-19, for which a considerable amount of resources were allocated during 2019 and 2020. The analyses made of them are quite enlightening; such is the case of the report issued by the prestigious research institute *The Alan Turing Institute* (von Borzyskowski, 2020) on the impact of the UK Data Science and Artificial Intelligence community's response to the fight against COVID-19. It was concluded that, although there had been a broad response by developing numerous projects, the results had not really been useful, suggesting several areas for improvement so that in a future pandemic a different outcome could be obtained. Along the same lines, an article (Wynantsl, 2020) published in the *British Medical Journal* analysed 731 predictive tools for the diagnosis and prognosis of COVID-19, concluding that none of them are suitable for clinical use. Only two of the proposals analysed are indicated as sufficiently promising to be considered for further analysis. A third exploratory paper, (Roberts, 2021) published in *Nature Machine Intelligence*, concludes along the same lines. The paper analyses 415 Machine Learning-based tools for predicting patient outcomes from chest X-rays and CT scans. The conclusion is again discouraging: None have been considered suitable for clinical use.

But there is potential for even worse. The potential for accepting whatever the model learned says as correct, distorting our view of reality, since the mistakes that the algorithm makes would be accepted as correct. The controversy over the use of COMPAS

(Correctional Offender Management Profiling for Alternative Sanctions) illustrates this point. COMPAS is a system developed by *Northpointe* to predict the likelihood that someone who has committed a crime will reoffend. The system is being used in several states in the United States, such as Florida, California, New York and others, as a system to assist in decision-making on issues related to defendants and prisoners, such as pre-trial detention or parole. The system is designed to specifically predict the risk of reoffending in the following two years based on a broad set of factors, using statistical machine learning models. Accepted as a very useful and reliable tool initially, one study (Angwin, 2016) raised concerns about its functionality. This study looked at how it had worked in one county in Florida where it was used as an aid in deciding on pre-trial detention pending trial. The case of 7214 defendants was monitored during 2013 and 2014. 39% of defendants who were rated by the system as highly likely to reoffend in the next two years (considered highly likely when the system scored above 7 out of 10) had not reoffended during that period of time. In almost 4 out of 10 cases, the algorithm's positives had turned out to be false positives, which was far from what was expected based on the studies provided by the developers. However, what was really striking about the study was the bias the algorithm had shown towards people of colour. Forty-five percent of defendants of colour who did not reoffend during the following two years were rated as having a high probability of reoffending, while for white defendants, this figure stood at just 23 percent. The algorithm used race as a determining factor, something that can happen because learning algorithms only look for a model that minimises a certain error function in the examples they are fed with and cannot take ethical aspects into account.

All this shows the need for a serious approach when analysing the possibilities of applying AI to a particular problem. AI has enormous potential, and we cannot afford to give up on it; on the contrary, we should try to make the most of it. However, this requires realism in terms of what it can deliver, and under what conditions. In the face of the exaggerated expectations that are being generated in society, it is necessary that those who have to decide on the policies and strategies to be followed, or on whether a specific project is worth tackling, be aware of the excellent possibilities that exist, while remaining realistic, weighing up the different issues to be taken into account in order to achieve success. Furthermore, system developers must be rigorous in applying AI techniques to get the right results. It is difficult to optimally adjust a model if you do not know how and why it works, and thus its key points, its limitations, the difficulties in applying it and its actual possibilities. If this is not taken into account and not applied correctly, it can lead to situations such as the examples mentioned above, and thus partly waste the investment made. This is particularly relevant in the case of the application of AI to the field of security, where the reliability of results and the consistent justification of decisions taken are necessary in many situations. The latter means that it is necessary in some cases for these decisions to be explained and justified, which in turn imposes a deeper analysis of the techniques to be applied. In this article, we will review the state of the art on the practical applications of AI in security, focusing on aspects that should be taken into account by those who apply it.

2.- WHAT IS IA?

The term Artificial Intelligence was coined in 1956 at the Dartmouth Summer Research Project on Artificial Intelligence, where it was defined as "the science and ingenuity of making intelligent machines, especially intelligent computing programs". The early years of this science focused on making machines think like humans; indeed, in 1985 Haugeland considered AI as *"The new and exciting endeavour to make computers think...machines with minds in the broadest literal sense"*. However, if we compare computers with the human brain, the two have different characteristics in terms of computational capacity. In some respects, these features are beneficial for computers, such as in processing speed and the ability to use and process a huge amount of information. However, in other respects, the computational capacity of the human brain outperforms computers, such as the sheer number of processors working in parallel (up to 100 trillion neurons), and the sheer number of parameters that can be adjusted to learn and adapt (each neuron is connected to some 10,000 neurons via a synaptic connection that are continuously adjusted through learning and adaptation). This led to the idea of AI incorporating systems that think rationally, but not necessarily as humans do; thus, Charniak and McDermott define AI as *"The study of mental faculties through the use of computational models"*. On the other hand, some authors argued that AI should try to make machines perform every activity that humans do, considering AI as the development of systems that act like humans: In 1991, Rich and Knight defined AI as "*The study of how to get computers to perform tasks that, at present, humans do best"*. It was proposed that AI should tackle complex problems in an intelligent way, performing even better than humans, i.e. as systems that act rationally to solve a problem. Thus, in 1998 Poole defined AI without relating it to human intelligence: "*Computational intelligence is the study of the design of intelligent agents"*.

3.- DEDUCTIVE SYSTEMS

Initially, the AI models proposed were deductive systems, i.e. systems that deduce or reason in order to make decisions or solve problems considered particularly complex. The research focused on *Automatic Reasoning* and the *Knowledge Representation* models needed to this end. These are models that: a) formally define a problem, b) represent the knowledge needed to solve it, and c) incorporate *inference engines* to implement automatic reasoning to find the solution. There was even the possibility of creating a generic system that could solve any problem *(General Problem Solver)*, but the analysis of reality showed that this was not possible, and that this had been a very naive, even deluded, approach. We may be repeating this somewhat deluded attitude today with the advance of new AI models. At the time, it was realised that the focus should be on tackling specific problems, developing *Expert Systems* or *Knowledge-Based Systems* specific to these problems.

These systems pose two problems: 1) human knowledge and reasoning are largely imprecise; and 2) the enormous combinatorial explosion that the system has to explore to find the solution. Thus, numerous models have been proposed to represent and reason with *Uncertainty Models*, such as *Non-Monotonic Reasoning* models, *Probabilistic Reasoning* models and *Fuzzy Logic* models, to name just a few. On the other hand, different *Heuristics* and *Metaheuristics* have been developed to try to solve the problem of finding a solution in a huge search space. The main difficulties encountered in applying this type of system to a real problem are: a) the need to represent a large amount of knowledge, with the consequent effort of the development team to acquire that knowledge and represent it, and b) that the deductive system is able to efficiently find a solution in such a huge search space. We can find various techniques to represent and reuse knowledge. *Rule-Based Systems* of the if-then type make it possible to represent knowledge in a simple and understandable way, but they are very simple and need a large number of rules to represent complex knowledge. On the other hand, *Ontologies* are complex models, which require specialised work for their design, but have the advantage of representing knowledge in a way that is reusable and understandable by both machines and people. Both systems have standard reasoning algorithms with fairly optimised implementations.

Regarding the other difficulty, several techniques have been developed to try to find an efficient solution to the search problem in the huge space of options, and they are often combined to obtain more effective systems. One technique consists of designing *Multi-agent Systems*; in other words, systems with several intelligent agents, where each agent is in charge of a specific, less complex task, and which interact and collaborate with each other in such a way that this joint collaborative work achieves an adequate solution to the problem. These systems reduce computational complexity by dividing the problem into subproblems and less complex subtasks. Furthermore, they allow the most appropriate AI technique to be chosen for each agent according to the task to be performed, thus optimising the performance of the system.

4.- APPROXIMATE SOLUTIONS

The combinatorial explosion makes it impossible to achieve a foolproof search algorithm, both in terms of time and space required for an exhaustive search. A problem as formally simple as 3-SAT, which is to decide on the consistency of a set of clauses, even reduced to each clause containing just 3 literals, turned out to be the first problem demonstrated to be NP-complete (Cook, 1971), and thus assumed to have no efficient algorithmic solution.

Therefore, AI cannot set out to find algorithms that always get it unmistakably right, since, if the problem addressed is really complex, these algorithms will not exist; AI is dedicated to finding approximate algorithms that get it right in as many cases as possible or to provide solutions that are close to the optimum. This is an issue that must be highlighted. It will not be possible to find perfect or infallible solutions, because in the vast majority of cases the problems addressed are NP-hard (Garey, 1979). When a problem is NP-hard, it is demonstrated that either it has no efficient solution (the option assumed by the scientific community), or all NP-complete problems would have an efficient solution (an option that is ruled out because, although these problems have been and are being studied by the entire scientific community, no one has found an efficient solution for any of them).

The fact that AI can only provide approximate solutions means that with each proposal, it is essential to perform rigorous validation and evaluation studies and analyses, measuring how good the proposed solution is, and what results can be expected from applying that solution. In fact, with each problem addressed by AI, standard measures have been established and are used to evaluate the proposed solutions. When AI techniques are to be applied, it is essential to understand these measures and to be aware of the results that the various optional AI algorithms that can be used are generating for that type of problem. As a result, this provides an estimate of what can be expected from the development, and what the most appropriate techniques would be. For further information on any aspect related to AI techniques, such as publications, journals, technologies, applications, researchers, etc., https://aitopics.org/ is an excellent source of information and data.

5.- INDUCTIVE SYSTEMS. MACHINE LEARNING.

As well as attempting to model rational reasoning or behaviour, AI has also addressed the incorporation of the main capability of the human brain with respect to computational systems, learning and adaptation. This is about modelling learning based on experience. Typically, models are developed from a data set or examples, which usually must be very large to generate good results.

Some systems use the example base itself as a component of the predictive system. This gives rise to the term *Exemplar-Based Models*. The *k-NN (k-Nearest neighbour)* algorithm is quite representative of such systems. As part of this algorithm, a distance measurement is defined and a decision is made based on the response of the k examples in the nearest example base. Another example is *Case-Based Reasoning* systems, where the solution is obtained by adapting the solution of the example from the nearest example base. To obtain good results, these models require an example base that is fairly representative of the possible input cases and, in the case of K-NN and its variants, well distributed. The choice of an appropriate distance function is critical, and indexing of the example base is required to reduce the computational complexity of having to go through the entire example base to solve each new case. When the example base is very large, the computational cost of the predictor is very high, and this makes it necessary either to reduce the number of examples used (e.g. by a selection of prototypes), or to structure them so that the search for the closest examples is limited to a subset of examples. One of the advantages of this type of model is its ability to learn, as the system improves as new examples are added, although the increase in computational complexity that this entails must be taken into account. For example, when evaluating behaviour and include corrected errors as new examples, the system will learn from its own mistakes and improve with use. On the other hand, the algorithm is sensitive to errors in the example base, as an error can affect all cases that are nearby because of the distance function. Nevertheless, K-NN is still an algorithm that can offer very good results if a suitable set of examples is available and a proper indexing of the example base is performed (Nagarka 2021); since it is included in practically all machine learning libraries, it can be applied quickly.

Other systems do not directly apply the example base to the system, but use it to adjust or learn a model. These are *Machine Learning* algorithms. The general purpose of these algorithms is to define complex models, either because they have many parameters, or because they have a structure that needs to be instantiated, or both. These models are tuned by a learning algorithm that learns the value of the parameters, and fixes each of the components of the structure to make the model as responsive as possible for that set of examples.

6.- IMPORTANCE OF THE EXAMPLES BASE.

Taking into account the general scheme of how these learning algorithms work, we can conclude that there are several considerations to take into account when applying them correctly (Dorfman, 2022):

a) If we want the algorithm to be able to learn to solve complex problems, we need the model to be complex, i.e. to have a large number of parameters or a structure with many components.
b) If the model has a large number of parameters or structure components, we will need a huge number of examples to be able to adjust them correctly. In any case, the number of parameters and the number of examples will have to be in line with one another to achieve correct learning.
c) The quality and distribution of the examples is crucial, as the model is adjusted to respond correctly to these examples. If the examples are biased or there are situations where the model is to be applied without a sufficient number of examples, the system learned will incorporate that bias and will not work correctly in those situations.
d) There is a danger of over-learning, i.e. the system specialises in responding well to the examples it has been trained on, but when applied to different cases it does not respond correctly.

Analysing a possible application can illustrate this. Imagine you want to design a system to detect whether email is *phishing,* and to do so you want to use an AI model using machine learning. We will need a set of examples, both positive and negative cases, i.e. a *set of examples* consisting of a set of mails each labelled as "no_phishing" or "phishing". If we only have 10 examples of phishing and 100 of non-phishing, and the algorithm is going to adjust, for example, 200 parameters, the probability that the learned model will be correct when applied to other cases is very low, because the model is being adjusted with very little information in relation to the number of parameters. Adjusting a 10-parameter model would probably give a much better result, although the amount of data being trained is very small and the model is too simple, so good results are not to be expected. On the other hand, even if you have a large amount of data, if in the dataset, the number of non-phishing examples is much higher than the number of phishing examples, i.e. if the set is not balanced, but there are many more negative than positive cases, to minimise the error, the system will adjust itself not to make mistakes in the more abundant cases; it will tend to reduce the false positives, although more errors will be allowed in the phishing cases, i.e. it will tend to have more false negatives. This is a consequence of the fact that, by default, learning algorithms seek to reduce the error in the training set of examples, and do not take into account the distribution of error. But what would be of most interest in this problem would be not having false negatives. It would therefore be appropriate to apply data pre-processing techniques for unbalanced cases, and to weight each type of error differently. Suppose next that, in order to generate many positive examples, we use a Chat GPT-style AI, automatically generating phishing emails. We could get a huge base of examples, and with many positive ones, but the vast majority in the same style, that of the AI used to generate them. The learning algorithm could learn a fairly complex model, as it would have a good number of examples, but it would be specialised in recognising only the phishing generated by that AI, recognising the way it generates phishing emails, but it would fail badly when it had to decide on phishing emails other than those generated in that way. Surely, this may explain some of the results obtained by the tools studied in (Roberts, 2021) to predict COVID-19 evolution from chest X-rays and CT scans, and to some extent the bias found in (Angwin, 2016) when analysing the performance of the COMPAS system. In short, achieving an adequate sample base, both in terms of number, quality, diversity and distribution, and carrying out an adequate prior analysis, including debugging and proper pre-screening (Baheti, 2023), will be critical to the success of the learned model.

7.- VALIDATION PROCESS.

To assess how good a learning algorithm is, we consider different features and measurements that serve as an estimate of how well the model learned will perform (Breck et al., 2019). This estimation consists of a *validation process* that tests the algorithm on untrained cases. It is usually trained with some of the examples available, *training examples*, leaving the rest as *validation examples*, to see how it will behave in cases where it has not been trained. The division of the available examples between training and validation is usually done randomly. To ensure that this random process does not distort the results, *cross-validations* are usually performed by dividing the set of examples into *n* chunks and repeating the model adjustment process *n* times, using one of the chunks each time as part of the validation process. The average value obtained in these n learning processes carried out during the cross-validation is then considered.

As for the measurements, they are calculated on the basis of the confusion matrix. We will illustrate this for the case of a binary classification problem. In case of more than two classes, we would use the concepts of positives and negatives per class. This matrix collects the VP values of true positives (examples indicated as positive by the algorithm and labelled as positive in the example base), VN of true negatives, FP of false positives (indicated as positive by the algorithm but labelled as negative in the example base), and FN of false negatives. The total number of examples will therefore be Total = VP+FP+VN+FN, while the number of examples indicated as positive by the algorithm will be iP = VP+FP, those indicated as negative by the algorithm will be iN = VN+FN, the number of true positive examples will be rP=VP+FN, and the number of true negative examples will be rN=VN+FP. The most commonly used measurements are:

- *Accuracy*: Accuracy = (VP+VN) / Total = (VP+VN) / (VP+FP+VN+FN)
- *Precision*: Precision = VP / iP = VP / (VP+FP)
- *Recall*: Recall = VP / rP = VP / (VP+FN)
- *Specificity*: Specificity = VN / rN = VN / (VN+FP)

If the set of examples is 90% positive and 10% negative, the $\textit{Naive}_0$ classifier, which does not take into account the features and simply classifies always as the majority class of the example base, would return the following: Accuracy=0.9, Precision=0.9, Recall=1 and Specificity=0. Expressed as a percentage, 90% of total hits, 90% of detected positives are correct, 100% of positives are detected, and 0% of negatives are detected. To properly assess what has been learned, it is useful to consider the data that would be obtained by the simplest algorithms (Veljanovska, 2017), such as the *Naive* Bayes algorithm, for the problem in question.

8.- GENERALISATION, OVER-TRAINING AND SPECIALISATION.

A learning model will have good *generalisation capacity* when it obtains good measures on the validation examples, with which it has not been trained. In proper training, these should be close to those obtained with the training examples, and in both cases high. The system will be considered as having learned when these measurements are significantly higher than the *Naive Bayes* algorithm.

If the model returns excellent measurements on the training set but poor measures on the validation set, the system will have over-trained itself (specialising in getting the examples it has been trained on right), but it will not have learned to solve the problem, as it does not perform well on the examples it has not been trained on. This is one of the main mistakes when applying machine learning algorithms (Pothuganti, 2018). The model is trained without taking into account how it behaves with the validation examples. The model gets better and better results with the training examples, so it is left to train for a long time, in the belief that it is learning more and more; however, at the end of the learning process, when it is applied to the examples it has not been trained with, the result is disappointing, as the system has been over-trained: It has trained itself intensely to improve its response using the specific training examples that it fails with the remaining ones. To avoid this, always bear in mind how the learned model behaves with the validation examples, and stop the learning process when it starts to deteriorate with these examples.

A more subtle and therefore more difficult to detect variant is *specialisation* to the overall set of examples, not only to the training examples. This would be the case in the example of the phishing detector trained with examples generated by a specific AI. As both the training and validation examples share a common feature in the way that AI generates text, the system could use this feature to optimise the model and it would not be detected by the standard validation process. This may always be the case, as the examples are obtained during a specific period, with a specific context and circumstances, and will therefore have characteristics corresponding to that period. However, when the learned model is applied, it is applied to a subsequent period, where some of these characteristics may have changed. It is a bias that will always be present to some degree. To recognise and measure the extent to which this specialisation has taken place, further validation processes can be designed, preferably in a real environment, or at least by collecting new examples from different contexts. When, for a complex problem, we obtain a model with a very high percentage of success (both in training and validation), for example above 95%, we should suspect a possible specialisation of the example base used for training, and consider a subsequent validation in a real context, or at least a different one. As this specialisation bias always occurs to some degree, it is generally desirable to include processes of adaptation and adjustment of the model that would learn to correct for it.

Let's look at another possible application to illustrate this issue. Suppose we are going to design a system to detect violent profiles on social networks using machine learning models. We would need to have a large set of examples of profiles labelled as violent or non-violent. These profiles would feature the contents of these profiles up to the present time. Therefore, they will address topics that have been topical during that period and will use the terms and jargon of that period. By the time we apply the model we have learned, new issues will most probably have arisen, others will no longer be topical, and new terms may be used. As the model has learned to classify profiles according to the original examples, it will not incorporate these thematic and language developments, and may even use some of the aspects that are no longer relevant. If the metrics are analysed on a current case basis, the results are likely to be worse than those obtained in the development study. Faced with such a rapidly evolving problem, further evaluation in a real context is very necessary, and it is highly desirable to build processes for adapting the model to the evolution of the problem into our system, e.g. by reinforcement learning.

9.- CLASSICAL MACHINE LEARNING MODELS

There are many models that have been used for *Machine Learning (*Veljanovska, 2017). Classical models include *Decision Trees*, *Random Forest* Decision Trees, or *Support Vector Machines (SVM*).

Decision-tree learning algorithms are non-parametric learning algorithms, which learn a tree structure to classify examples based on feature values. The tree divides the set of examples until, at the last level of the tree, the leaves, all examples are of the same class, or at least there is one predominant class. They are simple and fast-learning algorithms, which would be very successful if all possible cases were available as examples. They also offer the advantage of being easy to interpret, as it is equivalent to a rule-based system. The decision made at each leaf of the tree is equivalent to a rule, which is perfectly interpretable as long as it has a reasonable number of antecedents. One problem with decision trees is that they are very sensitive to examples, as small variations in the set of examples produce very different trees. However, the main drawback is that its generalisation capacity is very limited.

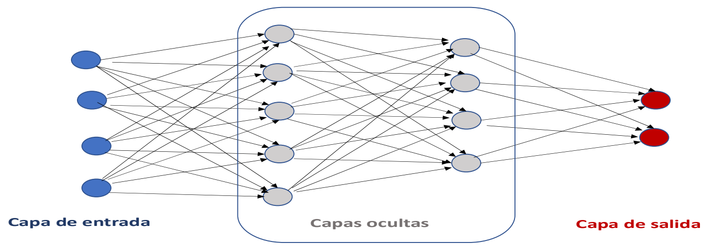
Randomised decision forests combine the response of multiple decision trees to obtain the final decision. This reduces the problem of sensitivity to examples and improves the generalisation capacity, at the cost of obtaining more complex models, reducing the interpretability of the model.

Support Vector Machines (SVMs) develop the idea that when a space has a very large dimension, examples will be far apart and linear classifiers perform very well. What they do is calculate the maximum margin linear classifier that would be obtained by transforming the input space into a very high-dimensional space. Instead of building the transformation, which would have an unaffordable computational cost, *kernel functions* are used to calculate the distance between the examples in this hypothetical space, so that the classifier can be adjusted. As the linear classifiers only depend on the points closest to the regression hyperplane, it only depends on those examples, the so-called *support vectors*. This makes the system insensitive to variations in the set of examples, as in many cases the support vectors will not change when examples are added or modified. Their main advantage is that they can perform well on very high-dimensional problems, even if very few examples are available (Soms, 2015). Although, in general, the models obtained with SVMs are not interpretable, for certain kernels, interpretable models are obtained (Castro, 2007). On the other hand, they can be very sensitive to errors, if any support vector is wrong it will greatly affect the model. Moreover, its effectiveness depends on the *kernel* function chosen and the scale of the data, so it is advisable to study different *kernel* functions in order to obtain the best results. Unfortunately, it is common to observe studies where, in order to compare a proposal, it is confronted with only the

SVM that uses the default parameters of the library being used, including the default *kernel*.

10.- NEURAL NETWORKS

The most widely used machine learning model in practical applications is *Artificial Neural Networks*, mainly because of their generalisation capacity compared to other models. One of the first neural network models was the *Perceptron (*Rosemblat, 1956), which simulates a fairly simple neural network based on the functioning of a biological neuron. This artificial neuron is connected to the inputs of the problem, which would be the neuron's stimuli, and the parameters to be adjusted are the weights of these connections between the inputs of the system and the neuron. These weights are used to weight the stimulus transmitted by the corresponding input. A supervised learning algorithm makes it possible to adjust the weights from the set of examples. *Perceptron* sparked a great deal of interest in the possibility of developing artificial brains using connectionist models, leading to the study of artificial neural networks as a sub-area of AI. However, a study of the capabilities and limitations of the perceptron (Minsky, 1969) demonstrated that this model was only capable of solving extremely simple problems, linearly separable problems. This made research on artificial neural networks all but disappear. The next impulse was the appearance of *backpropagation* (Rumelhart, 1986), which allowed for training systems with several layers of perceptrons, known as *multilayer perceptron, or feedforward network*, and which has ultimately been the most widely used neural network model. The figure below shows the structure of a multilayer perceptron with 4 inputs and 2 outputs. It has been designed with two hidden layers, the first with 5 nodes and the second with 3 nodes. In the multilayer perceptron, the nodes of one layer are connected to all the nodes of the next layer, so that if layer t has n_t nodes and layer t+1 has n_{nt+1} nodes, there will be n_t*n_{t+1} connections between both layers, each with a weight to be adjusted by the learning algorithm. It has been demonstrated that *multilayer perceptrons* with only one hidden layer are universal approximators, whether no activation function is used in the output layer (Hornik, 1989) or, as usual, such an activation function is used (Castro, 2000). This implies that for any problem for which the solution is a continuous function with well-defined limits, and for any desired level of approximation, there is always a *multilayer perceptron* with only one hidden layer that solves that problem at that level of approximation. This fact, together with the fact that the *backpropagation* algorithm under a suitable choice of training parameters converges to a local optimum, made the *multilayer perceptron* a widely used model in AI applications, with a large number of applications to a large number of domains. In fact, when it is stated that a neural network is applied, without specifying the structure, it is understood that a *multilayer perceptron* is being used.



Multi-layer perceptron with 2 hidden layers of 5 and 4 nodes respectively

Neural networks have a very good generalisation ability when trained on a large number of examples. The large number of parameters to be set must be taken into account, so if you do not have a large number of examples, it will be difficult to learn correctly. They have the disadvantage that they require somewhat manual and therefore costly application engineering. There are no clear criteria for choosing the number of hidden layers, the number of neurons in each layer, or the value of the most appropriate learning algorithm parameters. Moreover, converging to a local optimum means that you have to run it several times and keep the best option to try to obtain the global optimum. Thus, for the correct application, many tests have to be carried out with different values for the options, and the model with the best result must be chosen. Other drawbacks that were initially highlighted were that what the neural networks learned was not interpretable, they learned to respond correctly, but could not justify the reason for that response in a way that was understandable to a human being. In 1997, we showed that multilayer perceptrons with only one hidden layer could be translated into a fuzzy rule-based system (Castro, 1997), although the equivalence was as a mathematical function, and the rules could lie outside the problem domain. In 2002, we found an equivalence where the rules were within the domain (Castro, 2002), which made them interpretable to humans.

11.- DEEP LEARNING

Both SVMs and neural networks can be seen as systems that perform a complex transformation of the input space and then a linear classification on that transformed space. In the case of neural networks, the entire process up to the last hidden layer would be in charge of the transformation, and the output layer would be in charge of the linear classification. In the case of SVMs, the choice of the kernel corresponds to the choice of a complex transformation. In both cases, part of the transformation is not automatic, but chosen by the system designer. Although in neural networks the concrete parameters of the transformation, i.e. the weights of the connections reaching neurons in the hidden layers, are also adjusted by the learning algorithm, the decision on the number of hidden layers and the number of nodes in each hidden layer must be made manually.

The initial idea of deep learning is to design models that learn transformations to represent the input space. But rather than being transformations specialised in trying to solve a specific problem, they focus on trying to collect the most relevant features of the input space, on encoding the input space. Deep learning is known as such because it is designed in multiple layers, making the final network very complex. This idea has been used to improve learning outcomes and has led to a dramatic leap in terms of the things that can be done by AI systems, to the extent that when we now talk about an AI model it is taken to mean a model developed by machine learning and, more specifically, deep learning.

In general, these models use the encoder-decoder structure. Parametric models are designed to encode the input as a vector (encoder), together with models to decode (decoder) that vector and obtain the input values. To train these models, we do not need to have labelled data, so we can have as much data as we want without having to make the effort of labelling it, and thus do massive training.

The simplest case is that of *autoencoders*, where a neural network with a single hidden layer is trained to predict the input itself (Dong, 2018). The values of the neurons in the hidden layer is the encoding of the input, the encoder is the calculation of these values using the connections from the input to the hidden layer, and the decoder is the calculation of the output values from the hidden layer values. A directly applicable example of this model is the case of problems for which we have few labelled examples in relation to the dimensionality of the input, but we can collect a large number of unlabelled cases. For example, this has been used to detect credit card fraud (Lin, 2021). For the network to learn well, we will need the problem to have a limited number of parameters to adjust, according to the number of labelled examples we have. We can use the autoencoder procedure to reduce the dimension of the space layer by layer. We use autoencoder to encode the previous layer space to a lower dimensional space. In this way, we generate hidden layers that are trained step by step with a set of possible cases that do not need to be labelled, until we obtain a final hidden layer of a dimension according to the number of labelled examples we have. We will only train that last part of the network using the labelled examples. The improvement of the results with respect to adjusting all parameters of the complete network with these labelled examples is usually very significant.

12.- NATURAL LANGUAGE PROCESSING

One area where deep learning has made spectacular progress is with natural language models. In this case, the input space is a text and the objective is to train an encoder that transforms a sentence into a vector, or numerical matrix, representing the meaning of that sentence, for which coding-decoding systems are used. As millions of texts are available worldwide, and the capacity of computers has increased dramatically, it has been possible to train various systems that have learned the meaning, and which have proved tremendously effective, revolutionising intelligent natural language processing in just a few years. The *BERT* coder (Devlin, 2019), designed by Google, and which is freely available for download and use, has become a generally used standard model.

Initially, embedding models were introduced, to encode each word as a vector, so that nearby words in vector space had similar semantics. For example, the *Word2vec* model (Mikolov 2013) was obtained by training a network with a hidden layer to predict which words came before and after the input word. The vector of neurons in the hidden layer was the vector encoding the input word. In addition to the fact that words with similar meanings were encoded as proximate vectors, the operations on the vectors encoding the words had a semantic meaning. For example, by calculating (vector(woman) - vector(man)) + vector(king) we obtained a vector whose nearest significant point is vector(queen).

Subsequently, the encoding of sentences, sequences of words, was addressed with recurrent neural networks. In general, these are treated as a sequence of tokens, as some words are broken down into tokens applying a process of "*tokenisation*" (e.g. Breaking down into root plus ending). The encoding output of the already processed sequence chunk is added as an input to the encoder, which uses it to obtain the code of the next token. This is why we speak of models with memory, because they store what they have encoded so far as data in their memory to be taken into account for the next encoding process. The decoding process is modelled in a similar way. In the decoding process, attention learning (attention model) was also incorporated. Attention is responsible for deciding for each token which of the remaining tokens in the sequence are relevant for decoding the token. In this way, you simultaneously learn how to use the context.

The next step forward was the use of *Transformers* (Vaswani 2018). Transformers are models that take a text as input and generate an output text through an encoding-decoding process. Their design naturally allows them to be used for machine translation, by entering a text as input and obtaining the corresponding translation as output. Parallel encoding and decoding of tokens is possible in Transformers, which has proved to be essential for training them with a huge amount of data. For the encoding of each token, we take into account: a) the position of the token in the sequence and, b) the rest of the tokens in the sequence, by means of a self-attention model. The final coding values are calculated by a neural network. With the self-attention model, you learn to use the context to encode better. A sequential process is used for decoding where the next output token is generated at each step. In order to generate the next token, the following are taken into account: a) the position of the token to be generated, b) attention on the input tokens, and c) auto-attention on the tokens already generated. For the final decoding process, a neural network is used again, to which a softmax layer is added. The result is that transformers are very complex models, where many parameters have to be adjusted: those of the encoding neural network, those of the encoder's self-attention model, those of the decoder's attention and self-attention models, and those of the decoding neural network. Progress with the development of hardware with high parallel processing power, especially video card processors, has made it possible to train increasingly complex *transformer* models. Furthermore, many companies offer cloud computing services with tools to train these models. When models are very complex, the computational power required is so high that it can only be realised on the high-performance computing equipment of very powerful companies, or at supercomputing centres.

As indicated above, the encoder that has become the standard is *BERT,* developed by Google*,* which was obtained by training a model of transformers. It was initially trained using a huge number of examples obtained from Wikipedia and Google books. First, it was trained to learn to generate the full sentence when a word was deleted; and then to learn to decide whether a second sentence was actually the next sentence in the text. The model generated spectacular results, so much so that it is now widely used as a basis for any problem with natural language data. It uses the pre-trained general model and adjusts it using the examples to solve the specific problem, known as *fine-tuning*. Even starting from a pre-trained model, if a complex system is designed for *fine tuning*, high computational power would also be required. The final model can be used on less demanding equipment, but learning the model will require a great deal of computational effort. Pre-trained models exist in more than 100 languages, and open source Bert libraries can be found already tuned for many natural language processing problems.

To illustrate the leap in performance that Bert has made, let's look at the results with the SWAG *dataset*. This is a *dataset* with 113,000 questions for which 4 alternatives are offered. Each question is a description of a video scene, the 4 options are descriptions of 4 possible continuations of the scene, and the correct one is the one that describes what actually happens in the video. This consists of a task that requires understanding the language and deducing by common sense which of the 4 options would be the most reasonable. The best pre-Bert models did not exceed 60% accuracy, either in training or during validation. When Bert was trained, a success rate of over 86% was achieved in both training and validation. The success rate of one human, who was given 100 questions, was 85%.

13.- AUTOMATIC LANGUAGE GENERATION

In a similar way as general models that encode natural language have been developed, focusing on the encoding part of *transformers*, language generators have been developed taking advantage of the decoding part. The best known model is the GPT (*Generative Pre-Training,* Radford, 2018) model by OpenAI, which has now been developed in GPT-4 version. The GPT-2 version is available as open source, while the GPT-3 version (Brown 2020) is only available as an API.

The model has been obtained by training a *transformer* structure to generate text. The transformer decoder's self-attenuation model causes the system to look at the generated tokens that are most relevant for generating the next token. This ensures syntactic correctness and concordance of the text. The power of the model is based on the complexity of the individual components of the *transformer*, and on the tremendous training carried out: OpenAi indicates that GPT-3 is a 175 million parameter model and that 700GB of text has been used to train it. This is the model used as the base in Chat-GPT.

Chat-GPT incorporates a number of components from GPT-3. First, *fine tuning* by supervised learning has been incorporated to adapt the model for different tasks in response to text input: summarising, generating documents, answering questions, etc. This has resulted in the model that OpenAI refers to as GPT-3.5. In addition, it incorporates a number of alternative strategies to generate different responses to the same input, so that the policy to be followed to generate the response can be adapted. Chat GPT incorporates a reinforcement learning system to optimise the policy and generate progressively better responses. For this learning to take place automatically, a model has been trained to calculate the reward of a response to an input. To do this, a set of examples has been generated by taking inputs, generating different answers according to different strategies and manually evaluating each answer. Once a large set of evaluation examples has been generated, the model that automatically calculates the reward has been trained. In this way, the system self-evaluates and optimises the policy to be followed in order to generate better and better responses.

14.- IMAGE PROCESSING

Another problem to which AI has been applied and which has seen a breakthrough in recent years is image processing. This isdone bymeansof *convolutional neural networks (CNNs),* which have revolutionised the image recognition sector (Radzi, 2015). Convolutional networks use copies of the same feature detector for different areas of the image by convolution with a filter or kernel. This reduces the number of weights to be adjusted, as only the detector weights need to be adjusted. In addition, as different detectors can be used, different characteristics can be learned. Furthermore, the number of features increases, resulting in high dimensionality spaces. To reduce this dimensionality, each convolution layer is usually combined with a *pooling* layer, which reduces the dimension. By repeating the process of a convolution layer, followed by a *pooling* layer*,* a deeper and deeper network is generated using higher level features. In the final layer, a neural network is in charge of responding by using the features extracted by the previous layers.

LeNet (Lecun, 1998) can be considered as the forerunner of the convolutional networks used today. It was designed to sort handwritten digits. The network has two convolutional layers with poolingfollowed by a 3-layer neural network. It was the first major success in the use of RNC. Subsequently, increasingly complex models emerged. Alexnet (*Krizhevsky,* 2017) is a network with a similar structure to Letnet but with many more parameters, 61 million compared to Letnet's 62,000. VGGNet (Simonyan 2015) uses blocks instead of simple convolutional layers, i.e. combinations of convolutional layers that are applied repeatedly. It was a revolution as the blocks allowed high-level features to be learned. In addition, when using blocks, using convolutions based on small filters gives better results.

A good way of measuring the progress of RNCs in image classification is the error rate they achieve in the ImageNet challenge. The goal is to classify images into one of 10,000 possible categories, using 1.2 million images for training and 50,000 for validation. It is considered an error only if the correct ranking is not among the top 5 indicated by the network. In 2012, the lowest error rate was 26%, AlexNet reduced it to 17% and VGGNet in its successive evolutions to 8%. Human error for this challenge is 5%.

To tackle image processing problems, the convolutional part of a VGGNet-type model is often used as a pre-trained model, after having trained it with the Imagenet examples. Examples of these models are VGG-16 and VGG-19. 16 and 19 refer to the number of network layers that have been pre-set. The final model is obtained by adding a classification neural network and performing *fine tuning* by supervised training.

A typical example of application of these models in the field of security are facial recognition systems or the recognition of alert situations, which can be used for suspect identification, terrorist alert detection or access control, among others.

15.- AUTOMATIC IMAGE GENERATION

Generative Adversarial Neural Networks (GANs, Goodfellow, 2001) are the models currently used to generate artificial images. They can be used to generate images for simulation systems, or to generate fake images to try to deceive. GANs take on two neural networks, a "generator" and a "discriminator". The generator network produces samples of what we want to create (images, texts, sounds, etc.), while the discriminator network decides whether what is created by the generator network fits what is being searched for. It is a process where each network improves and learns from its opponent. In the original proposal, the generator network takes randomly generated noise as input, and generates images from that noise as output. To train it, the response of the discriminator for these generated images is considered. The extent to which the discriminator indicates that the images are not correct is the error that is used to correct the generator weights. To train the discriminator, we need a set of real images. The training examples of the discriminator will be both the images from that set of real images, which are labelled as correct, and the images from the generator, which are labelled as incorrect. During training, the improvement of one net serves to improve the other. The better the discriminator, the better the images generated by the generator must be for them not to be considered as incorrect; and the better the generator, the better the discriminator has to be in order to distinguish the generated images from the real ones.

If, instead of using an image generator from random noise, a certain type of input (an image, a text, etc.) is used, the system learns to translate that type of input into an image of the type defined by the set of real images. For example, if we use images of people as an input for the generator, and images of animals in the set of real images, once trained, our generator will translate the image of a person into the image of an associated animal. A number of specific architectures have been developed to translate images of one type into images of another type, such as DCLGan (Dual Contrastive Learning for Unsupervised Image-to-Image Translation, Han 2021).

16.- THE PROBLEM OF INTERPRETABILITY

As we have indicated, the use of deep learning has represented a leap forward in the different areas of AI in which it has been applied, with remarkable improvements in a very short time. This has led society, noticing this rapid improvement, to believe that this growth will be maintained until reaching results that are somewhat implausible. We must remain realistic. This leap has been motivated by the emergence of this technique, deep learning, and the improvement brought about by this technique will be limited. We must harness this improvement to make the most of it and obtain systems that help us with different problems; this is a very exciting challenge, but we must be aware of the difficulties and the limits that we will come up against. For example, it is unfeasible to think that models of the size of those developed for natural language can be built in other domains. Simply because there is not, and does not seem to be, such a huge set of examples as the amount of text that has been made available for training language models. In the face of specific problems, it is unimaginable that such an enormous number of examples could be available to us. This is not to say that they cannot be tackled and excellent results obtained, but they require effort and a thorough understanding of how the models work in order to get the most out of the examples available to us.

On the other hand, deep learning models, which have made such huge strides forwards when it comes to effectiveness, have come at the cost of a decline in the interpretability or explainability of the model. Decision trees, when the depth and number of nodes is not very high, can be translated into rules understood by humans. Therefore, it is possible to know what the model has actually learned and on the basis of what criteria it decides. In the case of SVMs, we have already shown that for certain kernel functions this is also the case (Castro, 2007), although for most kernels, we do not know how to interpret the model. In the case of neural networks, we demonstrated (Castro, 1997 and 2002) that multilayer perceptrons with a hidden layer could be interpreted as a set of fuzzy rules, namely one rule for each neuron in the hidden layer. But in deep learning models, with a large number of layers and each layer of a different type, this interpretability has been lost. Obtaining an interpretation of the characteristics obtained after so many different kinds of transformations of space appears very complicated. In short, with deep learning we learn very complex models, capable of predicting or classifying quite well, but we do not know how to describe, in a way that can be understood by humans, what it is that has been learned, nor with what criteria it is predicted or classified. In that sense, they are said to be black box models. Currently, this is one of the issues that the scientific community is tackling the most, trying to interpret the models. Different approaches and ways of dealing with the issue are emerging, but, for now, we are far from obtaining interpretations or explanations that can be considered understandable by a human being. As a result, if AI is to be applied to a problem where the model is required to explain decisions, we are limited to using simpler models (such as decision trees or neural networks with a single hidden layer) and with a reduced number of parameters, which means that we lose a lot in terms of the effectiveness we can expect from the model.

17.- APPLICATIONS OF IA IN THE SECURITY FIELD

An idea of the application areas that can be addressed with AI techniques in the field of security are:

17.1.- Predictive crime analysis

From the data records, machine learning techniques can be used to perform predictive analytics. Models can be designed to detect where crime is most likely to occur, and even the types of threats that are likely to occur. Although the system will make mistakes, it can be very helpful in preventing crime. This article (Cinelli 2019) provides a European comparison of predictive analytics in police intelligence. It indicates that there is growing interest, and that in 2019 these tools were already being used in 10 European countries, with around 20 different programmes. Predictive modelling is one of the typical applications of AI and Big Data. In the field of economics, where predictive models are commonly used, AI-based models have been incorporated into most tools, so it is foreseeable that the same will happen in the field of security.

17.2.- Automatic threat or crime detection

AI systems can be designed to detect suspicious persons or situations. The field of application is enormous, as specialised systems would probably have to be designed according to the type of threat. Facial recognition systems have achieved a fairly high level of development and are directly applicable tools. Interpol already uses facial recognition systems to identify persons of interest to an investigation (https://www.interpol.int/es/Como-trabajamos/Policia-cientifica/Reconocimiento-facial).

The identification of suspicious situations from images is more complex, to which advanced image classification techniques and models can be applied. Given the progress being made in the performance of deep learning image processing models, there is considerable promise in tackling concrete problems of this type. They would have a cost in terms of work and development time to be considered, and require a proper approach, but it must only be a matter of time before projects and tools along these lines start to emerge.

On the other hand, more specific systems can be developed, designed to detect various specific suspicious situations. For example, suspected false reports, such as the VeriPol system used by the Spanish National Police (Quijano-Sánchez 2018), or of possible fraudulent use of credit cards (Strelcenia 2022), or of possible tax fraud (Rodríguez-Pérez 2018). Well-applied machine learning techniques can work well in these cases. In general, these are problems where the type of error must be considered differently, with a preponderance of false negatives to be avoided, and where the set of examples will be highly unbalanced. The direct application of standard learning and validation models can be greatly improved by further study, and especially by incorporating reinforcement learning models so that the system can adapt to the evolution of the problem.

17.3.- Prevention of possible crimes

Another promising area for the development of tools and projects is the prevention of potential crime. AI systems can be applied to detect anomalies or behaviour that should be investigated to prevent a potential crime. Research is already underway on some of these problems. For example, in the detection of fake profiles in social networks (Breuer 2020), or the detection of fraudulent websites, or impersonation (Monaro 2021). In these cases, the aim is not to detect a crime, rather, behaviour that raises suspicions that criminal activities may be hidden behind a "site", "profile" or "identity". The problem is more complex than the direct detection of crime, but the benefit to the system is greater because of the prevention factor. On the other hand, if the aim is to investigate preventively, it seems reasonable that the system should provide arguments to justify it. This implies the use of AI models that are interpretable or explainable, something that as we have indicated reduces the possibilities to more classical machine learning models, with a reduced size. In many cases, public sites where positive cases are reported are available and can serve as a source of examples. On the other hand, this type of problem evolves very quickly, as profiles, websites and identities are constantly being changed to avoid being discovered, making these examples obsolete very quickly. If we train machine learning models from the direct features, by the time they are to be applied they may have changed enough to significantly reduce the hit rate. It is therefore necessary to use other types of features that are less sensitive to these changes, and with greater semantic power to make the model more explainable. In (Francisco, 2022b) we address the problem of detecting radical jihadist profiles on social networks. As a limited number of examples were available, a tool, Nutcracker (Francis 2023), was developed to generate tagged databases with minimal effort, using behavioural dynamics, the *deep relationships* between users, rather than more direct aspects, which could be more easily modified (Francis 2022a). In addition, we used discourse analysis techniques to extract less camouflageable, and more semantically powerful, features such as intentionality or emotion from the messages. Nutcracker is currently being used as part of a new project to detect fake news.

18.- CONCLUSIONS

In recent years, deep learning has been a revolution in the advancement of AI applications. Complex language models for both recognition, BERT, and language generation, GPT, have been obtained and can be trained by fine-tuning to tackle classification problems from text. In addition, complex pre-trained convolutional models, VGG-16 and VGG-19, have been obtained, which can also be trained by fine-tuning to tackle image classification problems. These advances mean that there is a broad spectrum of security problems where AI-based tools can be developed. Automatic threat or crime detection from video image analysis is an area where effective tools have already been developed, and where new ones are likely to emerge. It also seems promising to use deep learning to address the detection of specific security concerns, as is already the case for the detection of false reports. One application for which the prospects are very good is the detection of specific types of fraud, such as tax fraud, credit card fraud, cryptocurrency fraud, etc. Another area where AI techniques look promising is in predictive crime analysis. There are a number of tools being used, and the trend is for a progressive increase, with AI-based models to be incorporated into these tools in the foreseeable future. As far as preventive systems are concerned, they represent a field of application of undoubted added value with additional difficulties due to the rapid evolution of the problem and the need in some cases to use interpretable models. There is a great deal of interest in them, with a number of projects worth keeping an eye on. In conclusion, there is a wealth of possibilities for AI applications in security, which researchers, technologists and organisations have the opportunity, almost the obligation, to address as effectively as possible.

19.- BIBLIOGRAPHY

Angwin, J. Larson, J. (2016). "Machine Bias. *ProPublica*. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing. Retrieved 26 February 2023.

Aznarte, JL; Benitez, JM; Castro, JL (2007). Smooth transition autoregressive models and fuzzy rule-based systems: Functional equivalence and consequences. Fuzzy Sets and Systems, 158-24, p. 2734-2745.

Baheti, P. (2023) A Simple Guide to Data Preprocessing in Machine Learning. Retrieved 13 April 2023, https://www.v7labs.com/blog/data-preprocessing-guide

Breuer, A.; Eilat, R.; Weinsberg, U. (2020). Graph-Based Early Detection of Fake Accounts on Social Networks. Proceedings of The Web Conference 2020, p. 1287–1297

Brown, T; Mann,B; Ryder,N.; etc. ; Dario Amodei, D. (2020). Language Models are Few-Shot Learners. Retrieved 5 December 2022. https://arxiv.org/abs/2005.14165

Breck, E.; Polyzotis, N.; Roy, S; Whang, S.E.; Zinkevich M. (2019). Data Validation for Machine Learning. *Proceedings of the 2nd SysML Conference*, Palo Alto, CA, USA,

Castro, JL , Benitez, JM; Requena, I (1997) Are artificial neural networks black boxes? IEEE Transactions on Neural Networks, 8 (5) , pp.1156-1164

Castro, JL; Mantas, CJ; Benitez, JM. (2000) Neural networks with a continuous squashing function in the output are universal approximators, Neural Networks, 13 (6) , pp.561-563.

Castro, JL; Mantas, CJ; Benitez, JM. (2002) Interpretation of artificial neural networks by means of fuzzy rules. IEEE Transactions on Neural Networks, 13-1, p. 101-116

Castro, JL; Flores-Hidalgo, LD; Mantas, CJ; Puche, JM. (2007) Extraction of fuzzy rules from support vector machines. Fuzzy Sets and Systems, 158-18, p. 2057-2077.

Cinelli, V; Marnrique Gan, A. El uso de programas de análisis predictivo en la inteligencia policial: Una comparativa europea. Revista de Estudios en Seguridad Internacional. Vol.5, No. 2, pp 1-19.

Cook, S.A. (1971) The complexity of theorem proving procedures, Proceedings, Third Annual ACM Symposium on the Theory of Computing, ACM, New York, 151-158

Devlin, J; Chang, M; Lee, K; Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of the 2019

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1

Dong G.; Liao, G.; Liu, H.; Kuang, G. (2018). A Review of the Autoencoder and Its Variants: A Comparative Perspective from Target Recognition in Synthetic-Aperture Radar Images. IEEE Geoscience and Remote Sensing Magazine, 6. P. 44-68

Dorfman, E. (2022). How Much Data Is Required for Machine Learning? Retrieved 20 February 2023 https://postindustria.com/how-much-data-is-required-for-machine-learning/

Francisco, M; Castro, JL (2022a) A fuzzy model to enhance user profiles in microblogging sites using deep relations. Fuzzy Sets and Systems, 401 , pp.133-149.

Francisco, M; Benitez-Castro, MA; Hidalgo-Tenorio, E; Castro, JL (2022b) A semi-supervised algorithm for detecting extremism propaganda diffusion on social media. Pragmatics and Society 13 (3), pp.532-554.

Francisco, M; Castro, JL (2023). A Methodology to Quickly Perform Opinion Mining and Build Supervised Datasets Using Social Networks Mechanics. IEEE Transactions on Knowledge and Data Engineering. To be published soon, available at https://ieeexplore.ieee.org/document/10057085

Garey, M.; Johnson D. (1979) *Computers and Intractability; A Guide to the Theory of NP-Completeness*, 1979. ISBN 0-7167-1045-5

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Bengio, Y. (2014). Generative Adversarial Nets. In Advances in Neural Information Processing Systems, Editors: Z. Ghahramani and M. Welling and C. Cortes and N. Lawrence and K.Q. Weinberger. Curran Associates, Inc.
https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Han,J.; Shoeiby, M.; Petersson, L.;Armin, M.A. (2021) "Dual Contrastive Learning for Unsupervised Image-to-Image Translation," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW).*Nashville, TN, USA, 2021, p. 746-755,

Hornik, K.; Stinchcombe M.; White, H. (1989). Multilayer feedforward networks are universal approximators. Neural Networks 2-5, p. 359-366.

Krizhevsky, A.; Sutskever, I.; Hinton, G. E. (2017). "ImageNet classification with deep convolutional neural networks" Communications of the ACM. 60 (6): 84–90.

Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. (1998). "Gradient-based learning applied to document recognition" Proceedings of the IEEE. 86(11): 2278–2324

Lin, T; Jiang, J (2021) Credit Card Fraud Detection with Autoencoder and Probabilistic Random Forest. *Mathematics* 2021, *9*(21)

Mikolov, T., Chen, K., Carrado, G. and Dean, J.,(2013). *Efficient Estimation of Word Representations in Vector Space*. 1st ed. [ebook]. Retrieved 20 November 2015. <http://arxiv.org/pdf/1301.3781.pdf>

Minsky, M.; Papert, S. (1969). *Perceptrons.* M.I.T. Press.

Monaro M, Zampieri I, Sartori G, Pietrini P, Orrù G (2021). The detection of faked identity using unexpected questions and choice reaction times. Psychol Res. Sep;85(6):2474-2482.

Nagarka, P. Bhattacharya, A.; Jafari, O (2021). Exploring State-of-the-Art Nearest Neighbor (NN) Search Techniques. CODS-COMAD '21: Proceedings of the 3rd ACM India Joint International Conference on Data Science & Management of Data (8th ACM IKDD CODS & 26th COMAD), 443–446

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D., Sutskever, I. (2018) Language Models are Unsupervised Multitask Learners. Retrieved 18 January 2020. https://paperswithcode.com/paper/language-models-are-unsupervised-multitask

Radzi, F.; Khalil-Hani, M.; Imran, H.; …; Yan C. (2015). Generalizing convolutional neural networks for pattern recognition tasks. ARPN Journal of Engineering and Applied Sciences. 10. 5298-5308.

Rodriguez-Perez, E (2018). Análisis de detección de fraude fiscal mediante técnicas de Aprendizaje Automático. Tesis de Máster, Universidad Politécnicas de Madrid. Retrieved 25 March 2023. https://oa.upm.es/55006/1/TFM_EDUARDO_RODRIGUEZ_PEREZ.pdf

Strelcenia E; Prakoonwit, S. Generating Synthetic Data for Credit Card Fraud Detection Using GANs, *2022 International Conference on Computers and Artificial Intelligence Technologies (CAIT)*, Quzhou, China, 2022, p. 42-47

Pothuganti, S. (2018). Review on over-fitting and under-fitting problems in Machine Learning and solutions. International Journal of Advanced Research in Electrical Electronics and Instrumentation Engineering. 7.

Quijano-Sánchez, L.; Liberatore, F.; Camacho-Collados, J.; Camacho-Collados, M. (2018) Applying automatic text-based detection of deceptive language to police reports: Extracting behavioral patterns from a multi-step classification model to understand how we lie to the police. *Knowledge-Based Systems* 149.

Roberts, M.: Driggs, D.; Thorpe, M.; *et al.* (2021) Common pitfalls and recommendations for using machine learning to detect and prognosticate for COVID-19 using chest radiographs and CT scans. *Nat Mach Intell* 3, 199–217.

Rumelhart, D., Hinton, G. & Williams, R. (1986). Learning representations by back-propagating errors. Nature 323, 533–536.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, 65*(6), 386–408.

Simonyan K.; Zisserman, A.; (2015) Very Deep Convolutional Networks for Large-Scale Image Recognition. International Conference on Learning Representations, 2015. Retrieved 8 December 2018. https://arxiv.org/abs/1409.1556

Soms, N. (2015). A Review on Support Vector Machine: A Classification Method. National Conference on Recent Advances in Design, Automation and Intelligent Systems 2015

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J. Jones, L.; Gomez, A. N.; Kaiser, L.; Polosukhin, Illia (2017-06-12). "Attention Is All You Need". Retrieved 10 February 2018. https://arxiv.org/abs/1706.03762

Veljanovska, k. (2017). Machine Learning Algorithms Comparison. International Journal of Engineering}, vol 7-11, p. 60-64

von Borzyskowsky, I.; Mazumder, A.; Mateen, B.; Wooldridge, M. (2020). Reflections on the response of the UK's data science and AI community to the COVID-19 pandemic. Retrieved 20 February 2023 from https://www.turing.ac.uk/news/publications/data-science-and-ai-age-covid-19-report.

Wynants, L.; Van Calster, B.; Collins, G.S.; et al. (2020). Prediction models for diagnosis and prognosis of Covid-19: systematic review and critical appraisal. British Medical Journal, BMJ 2020;369:m1328.