ISSN 1870-4069

Predicción de victoria en videojuego competitivo con Deep Learning

Juan Alberto González Gutiérrez, Cristian Ramón Hernández Ordoñez, Marco Antonio Vidal Flores, Melina Alexia Zárate Zenteno

Universidad Autónoma del Estado de México,
Centro Universitario UAEM Nezahualcóyotl,
México

{alexzarate0007, cristian.her.ord,
rapsoryt, shsjuan}@gmail.com

Resumen. League of Legends es un videojuego de estrategia competitivo 5 vs 5, se puede observar un amplio margen de personajes (más de 150) disponibles para escoger; existe un interés importante en conocer el resultado de una partida antes de comenzarla, y aunado a ese interés una creencia popular, "Las partidas se pueden ganar desde la selección de personajes, ya que existen tipos de personajes que son mejores contra otros". Abriendo la posibilidad de poder predecir el resultado de una partida con solo saber sus condiciones iniciales, basándose en los personajes. Aquellas personas que pudieran acceder a esta información antes de comenzar a jugar tendrían una ventaja crucial. Para corroborar esta hipótesis se recurrió al uso de Deep Learning con el modelo de red neuronal "Perceptrón Multi Capa". Mediante una base de datos con un total de 62 mil partidas, las cuales fueron obtenidas directamente de la API oficial de la desarrolladora del videojuego RIOT GAMES, de esta forma se buscó como resultado que el algoritmo aprenda y finalmente realice una predicción acerca de qué equipo es vencedor en el estilo de juego clásico.

Palabras clave: API, Deep Learning, inteligencia artificial, red neuronal, videojuegos, predecir, perceptrón multicapa.

Prediction of Victory in a Competitive Video Game with Deep Learning

Abstract. A system developed in Python capable of predicting the victory of a competitive game, which is League of Legends, will be implemented. Through a large amount of data which will be obtained from the official API of the video game developer. The aim is to train a neural network, in this way the algorithm

will learn and finally make a prediction about which team is the winner in a specific style of play.

Keywords: Artificial intelligence, API, neural network, videogames, deep learning, predict, multilayer perceptron.

1. Introducción

1.1. Inteligencia artificial y League of Legends

La inteligencia artificial en pocas palabras podría definirse como "desarrollo de métodos que permitan comportarse a las computadoras de modo inteligente" [1], en otras palabras, es hacer computacional el conocimiento humano por procedimientos simbólicos o conexionistas. En términos coloquiales la IA se usa cuando una máquina es capaz de imitar las funciones cognitivas propias de la mente humana [2]. De hecho, los videojuegos y la inteligencia artificial son dos elementos que siempre han estado fuertemente ligados. Desde el inicio de la computación, la idea de conseguir que un ordenador tenga un comportamiento similar al de una persona humana autónomamente, ha supuesto un gran reto [3].

Los videojuegos son un mundo enorme, repletos de campos de investigación fascinantes, los cuales tienen posibilidades amplias para ser explorados por la gran cantidad de interacciones con varios procesos psicológicos. Los mundos virtuales allí presentes simbolizan un nuevo contexto en el cual las personas pueden interactuar y relacionarse de maneras que en el mundo real serían poco accesibles. Por otra parte, en los videojuegos se pueden experimentar situaciones que requieren diversas habilidades tanto motoras como cognitivas para la consecución de un objetivo determinado.

Cabe destacar que los usuarios de este tipo de tecnologías no solo interactúan con agentes de inteligencia artificial, sino que lo pueden hacer con otras personas permitiendo una amplia gama de actividades psicológicas [4] y cuando se trata de videojuegos competitivos, estos aspectos se llevan al límite. Se han realizado muchas estadísticas de los juegos más populares en 2021, los cuales, si se clasifican, el juego que más resalta y con diferencia, es League of Legends. Llegando a tener alrededor de 100 millones de jugadores activos cada mes. League of Legends (por sus siglas LoL) pertenece a una categoría de juegos llamada arena de combate multijugador en línea, más conocida como M.O.B.A (Multiplayer Online Battlefield Arena) [5].

Al ser tan competitivo las personas buscan adquirir la mayor cantidad de información posible, existen muchos servicios que ofrecen estadísticas a los jugadores, tanto del juego como de sus contrincantes en tiempo real, pero ninguna es capaz de ofrecer algo más complejo. La primera etapa de toda partida consiste en la selección de personajes; los jugadores tienen a su disposición 159 opciones, dejando miles de millones de posibles escenarios diferentes. Aunado a esto existe la creencia común de que las partidas se empiezan a ganar desde que se seleccionan los personajes, ya que algunos pueden tener fortalezas o debilidades dependiendo de a que otros se enfrentan.

A pesar de que es una creencia muy arraigada, aun no existe un sistema o método establecido para conocer realmente si se tiene o no ventaja conforme al equipo enemigo. Bajo este contexto, se plantea realizar una inteligencia artificial capaz de predecir qué equipo será el vencedor, solamente conociendo las características de los personajes seleccionados en el estilo clásico del juego. Información que de ser posible obtener, resultaría en una gran ventaja para aquellos que la aprovechen.

1.2. Redes neuronales artificiales y Deep Learning

Por definición, Deep Learning es un subconjunto dentro del campo del Machine Learning, el cual predica con la idea del aprendizaje desde el ejemplo. Su uso es excelente extrayendo patrones a partir de datos en bruto. Esperando que con el tiempo esos modelos sean capaces de solucionar el problema de forma extremadamente precisa, gracias a dicha capacidad de extraer patrones. Las Redes Neuronales Artificiales (RNA) se crearon para simular los componentes y las funciones del cerebro humano.

Las propiedades de las RNA permiten aplicarlas en problemas de reconocimiento de patrones y clasificación, dado que son capaces de descubrir relaciones no aparentes entre variables, y por la tanto, dan significado a los datos. Se describe un agente inteligente cuyo elemento de desempeño son modelos supervisados de RNA [6]. Como algoritmo de aprendizaje se hará uso del perceptrón multicapa [7] ya que se requiere la clasificación de dos clases (victoria y derrota), es uno de los algoritmos de aprendizaje más estudiados y utilizados en las redes neuronales artificiales, pero por su misma naturaleza presenta notables deficiencias, como por ejemplo la lentitud en su proceso de aprendizaje o su nula flexibilidad al ingreso de nuevas características una vez entrenada.

Con la ayuda de las Redes Neuronales Supervisadas se logrará hacer que la Red Neuronal aprenda y clasifique conforme a la base de datos previamente obtenida. El Algoritmo Backpropagation (BP), es tan solo un Entrenamiento Supervisado. En este algoritmo de Backpropagation se considera una etapa de funcionamiento donde se presenta, ante la red entrenada, un patrón de entrada y éste se transmite a través de las sucesivas capas de neuronas hasta obtener una salida y, luego una etapa de entrenamiento o aprendizaje donde se modifican los pesos de la red de manera que coincida la salida deseada con la salida obtenida por la red [8].

1.3. Trabajos relacionados

La predicción utilizando sistemas de aprendizaje profundo (comúnmente redes neuronales), ha sido manejada en diferentes áreas, como el clima, economía, medicina, industria, etc. Sin embargo, en el videojuego League of Legends no hay formalmente un desarrollo para saber predicciones utilizando Deep Learning, todo lo que tenemos son predicciones basadas en sistemas de Machine Learning y Reglas Fijas impuestas por estadísticas registradas. No obstante, se tiene en cuenta que en donde existen datos masivos, se puede trabajar para obtener predicciones con redes neuronales (Deep

Learning). Lo más que podemos obtener de trabajos previos son estadísticas históricas muy elaboradas referentes al videojuego. En sitios como:

- OP.gg [9]. Proporciona perspectivas a los jugadores de League of Legends acerca de su jugabilidad y sobre el juego en general.
- League Of Graphs [10]. Alternativa similar a OP.gg, proporcionando estadísticas avanzadas y patrones de jugabilidad.
- LolProfile [11]. Página para ver datos históricos sobre jugadores.
- Porofessor.gg [12]. Estadísticas por jugador en tiempo real.
- MOBAlytics [13]. Aplicación de estadísticas históricas y en tiempo real.
- League of Legends Wiki [14]. Proporciona información en general sobre el juego.
- Entre otros…

2. Método

2.1. Obtención de datos

Se tuvo que crear una cuenta en la API [15] de RIOT [16] la cual es la desarrolladora del videojuego. Una vez creada, RIOT asigna una "API Key" [17] por defecto que es necesaria para realizar consultas y obtener los datos.

A través del lenguaje de programación Python [18], se implementó un programa capaz de hacer consultas a la API con base a los datos que se requieren obtener.

En League of Legends existen múltiples modos de juego, en esta aplicación solo se centrará en el estilo clásico de juego ya que este es el modo competitivo.

2.2. Procesamiento de datos

Durante el proceso surgieron varias complicaciones. Una de ellas fue la limitación de consultas a la API en un determinado tiempo.

Por lo tanto, la solución fue que el programa estuviera en ejecución por horas en múltiples computadoras para obtener la cantidad suficiente de datos para un análisis bien sustentado de las partidas. Consiguiendo un total de 60,410 registros.

Los datos a obtenidos se dividieron en dos bases de datos. La primera base de datos se utiliza para almacenar a los campeones (personajes del juego), los cuales tienen sus propias características necesarias que los diferencian unos a los otros como: ataque, armadura, velocidad, magia, dificultad, etcétera.

Tabla 1. Configuraciones propuestas para la red neuronal. [Creación propia].

No.	Atributo	Definición	Descripción
1-10	Champ1-10	Personajes	Se refiere a los 10 personajes utilizados en la partida
11-12	Attack1-2	Ataque	Indica las sumatorias de ataque de los personajes de cada equipo
13-14	Defense1-2	Defensa	Indica las sumatorias de la dificultad de los personajes de cada equipo
15-16	Difficulty1-2	Dificultad	Indica las sumatorias de la defensa de los personajes de cada equipo
17-18	Magic1-2	Magia	Indica las sumatorias de la magia de los personajes de cada equipo
19-20	Armor1-2	Armadura	Indica las sumatorias de la armadura de los personajes de cada equipo
21-22	AttackDamage1-2	Daño de ataque	Indica las sumatorias del daño de ataque de los personajes de cada equipo
23-24	AttackRange1-2	Rango de ataque	Indica las sumatorias del rango de ataque de los personajes de cada equipo
25-26	AttackSpeed1-2	Velocidad de ataque	Indica las sumatorias de la velocidad de ataque de los personajes de cada equipo
27-28	Health1-2	Vida	Indica las sumatorias de la vida de ataque de los personajes de cada equipo
29-30	MagicResist1-2	Resistencia mágica	Indica las sumatorias de la resistencia mágica de los personajes de cada equipo
31-32	Mana1-2	Maná	Indica las sumatorias del maná de los personajes de cada equipo
33-34	MoveSpeed1-2	Velocidad de movimiento	Indica las sumatorias de la velocidad de movimiento de los personajes de cada equipo
35	Win	Victoria	Indica el equipo que ganó la partida
36	Duration	Duración	Indica la duración de la partida

La segunda base de datos es referente a la información de las partidas, requirió un proceso mayor ya que cada uno de los 10 personajes en partida cuenta con sus propias estadísticas, se estarían teniendo 24 características por personaje lo que al acumular los 10 personajes nos deja con 250 datos en cada registro, teniendo un total de 15,102,500 datos, saturar de información irrelevante a una red neuronal puede llevar a resultados ambiguos o erróneos, por lo que se realizó un procesado más profundo de los datos resumiendo la información de los personajes de cada equipo en sumatorias de sus características en común.

Cada equipo contará con las respectivas sumatorias de las características de los campeones seleccionados y finalmente el resultado de la victoria y duración de la partida.

Quedando así la base de datos, normalizada y lista para entrenar la red neuronal. Los patrones de entrada de la arquitectura (1 - 34) y las clases o salidas (35 - 36) quedaron organizados con la siguiente estructura: Previo a la implementación de la red neuronal, se realizó una limpieza a la base de datos para eliminar todas aquellas partidas con una

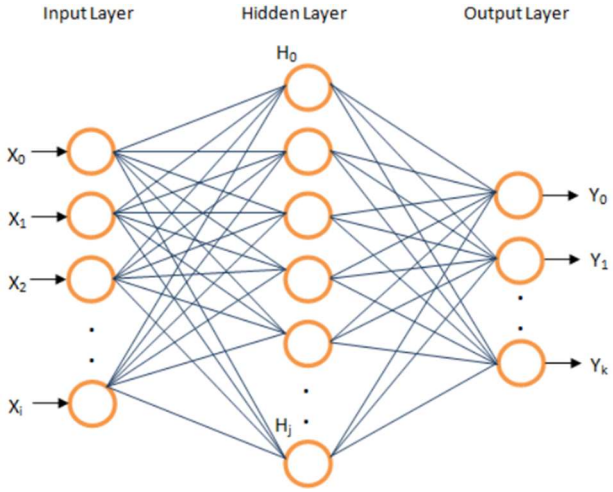


Fig. 1. Ejemplificación del perceptrón multicapa. (Tipo Red Neuronal utilizada) [19].

duración menor a 15 minutos, ya que estas suelen ser partidas que finalizan por factores externos al propio juego como el abandono de jugadores, mantenimiento, perdidas de conexión o errores en los servidores y hacer uso de dichas partidas en el entrenamiento no harían más que generar ruido y por ende un aprendizaje deficiente. Quedándonos con un total de 43,908 registros de 36 datos cada uno (1,580,688 datos en total).

2.3. Desarrollo de la red neuronal

Se utilizó Python como lenguaje de programación ya que el desarrollo de aplicaciones relacionada a la ciencia de datos es más eficiente por la amplia variedad de librerías relacionadas al campo de investigación. La estructura de un perceptrón multicapa modelo luce de la siguiente forma: Se hizo uso múltiples módulos de la librería *scikit-learn* [20], librería especializada en Machine Learning y Deep Learning, específicamente se utilizaron los siguientes módulos:

Classification_report: Este módulo nos permite obtener análisis de resultados complejos. Se tienen 3 datos que son de vital importancia:

Accuracy: Es la relación de predicciones correctas de forma global. Se obtienen al dividir el número total de aciertos entre el número total de registros [21]:

$$\text{Accuracy} = \frac{\text{Verdaderos positivos} + \text{Verdaderos Negativos}}{\text{Todas las muestras}}. \quad (1)$$

Recall: Es la relación de predicciones correctas que hay por cada clase. Se obtiene al dividir los aciertos de cada clase entre la cantidad de valores existentes de esa clase en el conjunto de prueba [21]:

$$\text{Recall} = \frac{\text{Verdaderos positivos}}{\text{Todas los positivos reales}}. \quad (2)$$

Tabla 2. Configuraciones propuestas para la red neuronal. [Creación propia].

N° de iteraciones	N° de neuronas	N° de capas ocultas
10,000	10	7
10,000	10	5
15,000	50	5

Tabla 3. Propiedades de la función de activación con figura [22].

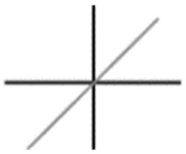
Función de activación	Representación matemática	Figura
Identity	$f(x) = x$	

Tabla 4. Resultados completos de configuración 1 [Creación propia].

	Accuracy	Recall 1	Recall 2
1	47.58	9.17	95.35
2	48.73	2.77	99.19
3	52.02	60.11	41.95
4	55.28	99.17	0.68
5	51.92	52.85	50.76
6	53.06	67.27	35.39
7	55.49	86.71	16.65
8	55.86	75.83	28.79
9	55.03	85.14	17.58
10	51.69	42.24	63.45

Train_test_split: Nos permite dividir rápidamente nuestros datos en dos conjuntos, prueba y entrenamiento. Esto debido a que no se puede hacer uso de los datos de entrenamiento para comprobar la eficacia de la red neuronal ya que se estarían prediciendo datos vistos previamente. Por la gran amplitud de datos que poseemos, se escogió un 88% (53,161 registros) de los datos de manera aleatoria, para realizar el entrenamiento, y un 12% (7,249 Registros) para la fasede pruebas.

Neural_network MLPClassifier: Este módulo contiene herramientas para un rápido desarrollo de la red neuronal Perceptrón Multicapa. Entre los aspectos más importantes al momento de desarrollar este tipo de redes neuronales está la correcta selección de un porcentaje del conjunto de prueba el cuál para este caso fue del 12% de patrones y la cantidad de capas ocultas, iteraciones de entrenamiento, función de activación y neuronas por capa oculta que tendrá el perceptrón. La red neuronal puede ser muy sensible a los cambios de dichos parámetros, por lo que se usaron 3 diferentes configuraciones.

Cabe destacar que en cada configuración se hizo uso de la función de activación identidad, esto con la intención de hacer un análisis individual para cada caso predictivo y así obtener un porcentaje preciso de posible resultado, más sin en cambio, en las

Tabla 5. Resultados resumidos de configuración 1 [Creación propia].

	Accuracy	Recall 1	Recall 2
Mínimo de aciertos	47.58%	2.77%	0.68%
Máximo de aciertos	55.86%	99.17%	99.19%
Promedio	**52.666%**	**58.126%**	**44.979%**

Tabla 6. Resultados completos de configuración 2 [Creación propia].

	Accuracy	Recall 1	Recall 2
1	55.39	97.56	2.93
2	50.99	47.03	55.91
3	48.41	10.68	95.35
4	44.8	1.57	98.59
5	45.64	8.04	92.41
6	45.43	2.7	98.59
7	55.94	92.29	10.73
8	55.03	82.09	21.37
9	45.35	2.25	98.78
10	46.36	8.42	93.56

Tabla 7. Resultados resumidos de configuración 2 [Creación propia].

	Accuracy	Recall 1	Recall 2
Mínimo de aciertos	44.8%	1.57%	2.93%
Máximo de aciertos	55.94%	97.56%	98.78%
Promedio	**49.334%**	**35.263%**	**66.822%**

salidas finales los resultados se tornan a sus extremos, dando un resultado de victoria o derrota:

3. Resultados

Si bien el desempeño de la red neuronal suele ser inconsistente se realizaron pruebas con las configuraciones propuestas para la red neuronal (ver tabla 2) para confirmar la consistencia de los datos obtenidos y asegurar la mínima variabilidad al cambiar los factores de influencia.

Para cada configuración fueron realizadas 10 redes neuronales con inicialización de pesos aleatorios para evitar cualquier influencia en los resultados.

Configuración 1: (Número de iteraciones = 10000, número de neuronas = 10, número de capas ocultas = 5). Los resultados completos se muestran en las Tablas 4 y 5. La primera red neuronal mostró un promedio de Accuracy muy cercano al 50% lo que no queda muy lejos del azar y los aciertos en las predicciones de victoria y derrota de igual manera están muy alejados unos de otros por lo que pareciera no existir relevancia en las predicciones.

Tabla 8. Resultados completos de configuración 3 [Creación propia].

	Accuracy	Recall 1	Recall 2
1	55.38	85.34	18.1
2	48.35	9.34	96.89
3	47.35	7.36	97.1
4	49.74	19.51	87.35
5	55.45	90.89	11.37
6	55.53	98.45	2.12
7	55.39	88.92	22.4
8	53.82	72.61	30.45
9	51.43	37.11	69.25
10	52.68	53.02	52.25

Tabla 9. Resultados resumidos de configuración 3 [Creación propia].

	Accuracy	Recall 1	Recall 2
Mínimo de aciertos	47.35%	7.36%	2.12%
Máximo de aciertos	55.53%	98.45%	97.1%
Promedio	**52.512%**	**56.255%**	**48.728%**

Configuración 2: (Número de iteraciones = 10000, número de neuronas = 10, número de capas ocultas = 7). Los resultados completos se muestran en las Tablas 6 y 7.

Configuración 3: (Número de iteraciones = 15000, número de neuronas = 50, número de capas ocultas = 5). Los resultados completos se muestran en las Tablas 8 y 9. Como se puede observar los resultados con diferentes redes neuronales se mantienen con un margen de mínima alteración, rondando el 50% de precisión.

4. Conclusiones y trabajo a futuro

Se partió de la hipótesis de que los personajes seleccionados en una partida clásica tenían la capacidad de influir sobre el resultado de esta respecto a qué equipo sería el ganador, creencia que es muy popular entre la comunidad y tomada casi como una verdad absoluta. Realizando el análisis de las 43,908 partidas con redes neuronales no ha sido posible el obtener más allá de un 51.504% de precisión al intentar predecir la victoria con esta información.

Esto es prácticamente equivalente a tirar una moneda al aire para conocer el resultado de una partida clásica en el videojuego League of Legends. Conocer esta información nos resulta relevante ya que no solo demuestra que dicha creencia popular es falsa, sino que, el cómo concluye cada enfrentamiento en el juego depende de las decisiones y acciones que ejecute cada persona y no de la herramienta que decida utilizar, en este caso sus personajes. Actualmente existen herramientas que antes de

empezar una partida proporcionan estadísticas de los personajes brindando una supuesta ventaja de conocimiento a los jugadores.

Lo cual queda demostrado como una pérdida de tiempo valioso en la preparación de una partida, ya que este tiempo no supera los dos minutos. Al demostrar que los resultados de las partidas están influenciados completamente por el comportamiento de los jugadores podría resultar viable no analizar los personajes sino a los mismos jugadores, datos como: tiempo de experiencia en el juego, victorias y derrotas de partidas recientes, experiencia con el personaje, tendencias de jugabilidad entre otros.

Este tipo de datos podría proporcionar resultados basados en comportamientos y tendencias humanas cosas como el cansancio del jugador, su humor o incluso su grado de concentración. También queda la opción a futuro de no solo analizar las victorias ya que se sabe que los jugadores son la principal influencia en el juego se puede intentar predecir la duración de las partidas y datos relevantes respecto al comportamiento que tendrán los jugadores durante la partida otorgando una ventaja competitiva a aquellos que posean dicha información.

Referencias

1. Oracle México: ¿Qué es la inteligencia artificial (Artificial Intelligence, AI)? (2022)
2. Pérez Orozco, B., Rentería Rodríguez, M. E.: Inteligencia Artificial. INCyTU. Oficina de Información Científica y Tecnología para el Congreso de la Unión. Nota-INCyTU, no. 12, pp. 1 (2018)
3. Utande, P.: Aplicación de Inteligencia Artificial en videojuegos. Uso de la variante del algoritmo MINIMAX PODA Alpha-Beta para su desarrollo. Escuela Técnica Superior de Ingeniería en Sistemas Informáticos. Universidad Politécnica de Madrid. Madrid, España. pp. 5 (2017)
4. Leyva-Rodríguez, J. E., Varela-García, J. D.: El videojuego League of Legends y su efecto en memoria de trabajo visual y solución de problemas. Programa de Psicología. Escuela de Medicina y Ciencias de la Salud. Universidad del Rosario. Bogotá, Colombia. pp. 6 (2016)
5. League of Legends. Interaxion group (2019)
6. Mariño, S. I., Primorac, C. R.: Diseño de un agente inteligente basado en una red neuronal artificial supervisada. Validación en un dominio botánico. Revista de la Escuela De Perfeccionamiento en Investigación Operativa, vol. 29, no. 49, (2021)
7. IBM. Perceptrón multicapa. Documentación de IBM (2021)
8. Hidalgo-Cajo, I. M., Yasaca-Pucuna, S., Hidalgo-Cajo, B. G., Hidalgo-Cajo, D. P., Latorre-Benalcázar, N. B.: Estudio comparativo de los algoritmos BackPropagation (BP) y multiple linear regression (MLR) a través del análisis estadístico de datos aplicado a redes neuronales artificiales. Revista Boletín Redipe, vol. 9, no. 3, pp. 144–152 (2020) doi: 10.36260/rbr.v9i3.939.
9. OP.gg. LoL Stats, Record Replay, Database, Guide (2022)
10. LeagueOfGraphs. LoL Champions, Summoners Stats & Rankings (2022)
11. LolProfile. League of Legends: Summoner Search & Stats (2022)
12. Porofessor.gg. Búsqueda de partidas en directo de League of Legends, y estadísticas de jugadores en tiempo real. (2022)

13. MOBAlytics. League of Legends: Summoner Stats, Match History and Champions Builds (2022)
14. League of Legends Wiki. Resource for League of Legends by a Wiki Fandom (2022)
15. Red Hat. ¿Qué es una API?. El concepto de las interfaces de programación de aplicaciones (2017)
16. Riot Games. Riot Developer Portal. League of Legends developer community with access to game data (2021)
17. Surática Software. ¿Qué es una API Key?. Diccionario (2021)
18. Python Software Fundation. Python 3.10.3 documentation (2022)
19. Neural Networks and MLP. Red Neuronal Perceptron Multicapa. DotNetLovers (2020)
20. scikit-learn. Machine Learning in Python (2022)
21. Lawtomated. 4 Things you need to know about AI: accuracy, precision, recall and F1 scores (2019)
22. Karakaya-Ulucan, D., Ulucan, O., Turkan, M.: Electronic Nose and Its Applications: A Survey. International Journal of Automation and Computing. Vol. 17, pp. 179–209, (2019) doi: 10.1007/s11633-019-1212-9.
23. Guerrero, S.: Machine Learning, el futuro de la Inteligencia Artificial (2019)
24. Neural Networks and MLP. Red Neuronal Perceptron Multicapa. DotNetLovers (2020)