Clasificación de perfiles por preferencia de personalidad en redes sociales mediante redes neuronales artificiales aplicando la técnica de retro-propagación

Dr. Adolfo Meléndez Ramírez*, Ing. Miguel Ángel Garrido Gutiérrez, M. en ADN. Juan Manuel Stein Carrillo*, MTE. Carlos Alfonso Trejo Villanueva, Ing. Ruth Anel Gutiérrez González.

Acerca de los autores...

* Académico adscrito a la División de Sistemas Computacionales del Tecnológico de Estudios Superiores de Ecatepec
Adolfo_melendez@tese.edu.mx
jmsteinc@tese.edu.mx
201711670@tese.edu.mx
carlostrejo@tese.edu.mx
201711669@tese.edu.mx

Resumen

El presente trabajo de investigación propone utilizar técnicas de minería de datos para la clasificación de aprendizaje supervisado: Redes Neuronales Artificiales (RNA), Clasificador de Naive Bayes y KNN, con el fin de poder clasificar perfiles de usuarios de redes sociales, para auxiliar a instituciones educativas de nivel superior a ofrecer sus servicios a estudiantes potenciales que tengan el perfil adecuado para sus ofertas educativas. El objetivo principal de la investigación es clasificar dichos

perfiles con base en las preferencias de los usuarios en sus redes sociales. Los resultados obtenidos por la RNA son comparados con los que otorgan otras técnicas de clasificación de minería de datos de aprendizaje supervisado: Clasificador de NaiveBayes y KNN. Al final, se pudo hacer una clasificación satisfactoria con los tres métodos, siendo la RNA la que mostró mejores resultados.

Palabras Clave: Aprendizaje supervisado, Clasificación, Minería de datos, Redes neuronales, Redes sociales.

Abstract

This research paper proposes to use data mining techniques for the classification of supervised learning: Artificial Neural Networks (RNA), NaiveBayes Classifier and KNN, in order to be able to classify profiles of users of social networks, to assist educational institutions higher level to offer their services to potential students who have the right profile for their educational offers. The main objective of the research is to classify these profiles based on the preferences of the users in their social networks. The results obtained by the RNA are compared with those granted by other supervised learning data mining classification techniques: NaiveBayes Classifier and KNN. In the end, a satisfactory classification could be made with the three methods, with RNA showing the best results.

Keywords: *Supervised Learning, Classification, Data Mining, Neural Networks, Social Networks.*

Introducción

En la actualidad, el uso diario de redes sociales se ha incrementado de manera acelerada. En el último año, de 2017 a 2018, el porcentaje mundial de usuarios creció

un 13% (Kemp, 2018) llegando a 3.196 billones de usuarios. En México, hasta el 2017, teníamos 65 millones de usuarios de internet, de los cuales el 75% (48.75 millones) usa al menos una red social (2017). Cada usuario genera gran cantidad de información que es almacenada y puede ser consultada bajo ciertas condiciones, con el fin principal de poder ofrecerles mejor publicidad y ofertas más adecuadas.

Según cifras de la Encuesta Nacional sobre Disponibilidad y Uso de Tecnologías de la Información en los Hogares (ENDUTIH) 2017, del Instituto Nacional de Estadística y Geografía (INEGI) (ENDUTIH 2017, 2018), el grupo de edad de población que más utiliza el internet son los que pertenecen al grupo de entre los 18 a 34 años, seguidos por quienes tienen de 6 a 17 años. Grupos en que la mayoría de los casos se encuentran en el rango de edad para elegir carreras universitarias.

La minería de datos incluye diferentes algoritmos para clasificar con aprendizaje supervisado, entre ellos están las redes neuronales. Se le conoce como aprendizaje supervisado cuando las clases ya se conocen y están determinadas antes de examinar la información (Dunham, 2003).

Las *Redes Neuronales Artificiales* (RNA) son modelos que intentan reproducir el comportamiento del cerebro, y a semejanza de éste, realizan una simplificación, averiguando cuáles son los elementos relevantes del sistema. Una elección adecuada de sus características, más una estructura conveniente, es el procedimiento convencional utilizado para construir redes capaces de realizar una determinada tarea. Este modelo posee dispositivos elementales de proceso, que son el equivalente a las neuronas en el sistema biológico.

Backpropagation o algoritmo de propagación hacia atrás, también conocido como retropropagación, es una regla de aprendizaje que se puede aplicar en redes con más de dos capas de neuronas. Una de sus características es la capacidad de representar el conocimiento en las capas ocultas, logrando así cualquier correspondencia entre entradas y salidas.

El funcionamiento de una red backpropagation consiste en el aprendizaje de un conjunto predefinido de pares, entrada-salida dados como ejemplo, empleando un ciclo propagación-adaptación de dos fases. Primero se aplica un patrón de entrada en la capa de acceso de la red, que luego se propaga hacia las capas superiores hasta generar una salida y se compara el resultado alcanzado en cada neurona de salida con el valor deseado para esa neurona y se obtiene un error para dicha unidad. A continuación, estos errores se transmiten hacia atrás, a todas las neuronas de la capa intermedia que contribuyan directamente a la salida, recibiendo el porcentaje de error aproximado a la participación de la neurona intermedia en la salida original. Este proceso se repite capa por capa, hasta llegar a la entrada y hasta que cada neurona haya recibido un error que describa su aporte al error total. Según el valor del error recibido, se reajustan los pesos de las conexiones entre cada par de neuronas en la red, de manera de que el error total cometido para ese patrón disminuya. [García Martínez *et al*., 2003]

El algoritmo utiliza una función o superficie de error asociada a la red, buscando el estado estable de mínimo error a través del camino descendente de la superficie de error. Por esta razón es que realimenta el error del sistema para efectuar la modificación de los pesos en un valor proporcional al gradiente decreciente de dicha función de error.

Funcionamiento del algoritmo:

Dada una neurona (U_i) y su salida y_i, el cambio que se produce en el peso de la conexión que une la salida de dicha neurona con la unidad $U_j(w_{ij})$ para un patrón de aprendizaje p es:

$$\Delta w_{ij}(t+1) = \alpha \delta_{pj} y_{pj}$$

El subíndice p se refiere al patrón de aprendizaje concreto y α es la constante de aprendizaje o tasa de aprendizaje.

La regla delta generalizada difiere con la regla delta en el valor concreto de δ_{pj}. En las redes multinivel con capas ocultas no se conoce la salida deseada de estas capas para poder calcular los pesos en función del error cometido. Sin embargo, inicialmente podemos conocer la salida deseada de las neuronas de salida. Para la unidad U_j de salida, definimos:

$$\delta_{pj} = (d_{pj} - y_{pj}) f'(net_j)$$

Donde d_{pj} es la salida deseada para la neurona j y el patrón p y net_j es la entrada neta de la neurona j. Si U_j no es de salida, se tiene:

$$\delta_{pj} = \left(\sum_k \delta_{pk} w_{kj} \right) f'(net_j)$$

Donde el rango de k cubre todas las neuronas a las que está conectada la salida de U_j. El error que se produce en una neurona oculta es la suma de todos los errores cometidos por las neuronas a las que está conectada su salida, multiplicados por el peso de la conexión correspondiente.

Adición de un momento en la regla delta generalizada

Se utiliza una amplitud de paso dada por la tasa de aprendizaje α. A mayor tasa, mayor será la modificación de los pesos en cada iteración, y más rápido será el aprendizaje, dando lugar a oscilaciones. Para filtrarlas un término (momento) β,

$$w_{ji}(t+1) = w_{ji}(t) + \alpha \delta_{pj} y_{pj} + \beta \left(w_{ji}(t) - w_{ji}(t-1) \right) =$$

$$\Delta w_{ji}(t+1) = \alpha \delta_{pj} y_{pi} + \beta \Delta w_{ji}(t)$$

donde β es una constante que determina el efecto en $t+1$ del cambio de los pesos en el instante t.

Esto permite la convergencia de la red en menor número de iteraciones, ya que si en t el incremento de un peso era positivo y en $t+1$ también, entonces el descenso por la superficie de error en $t+1$ es mayor. Sin embargo, si en t el incremento era positivo y en $t+1$ es negativo, el paso que se da en $t+1$ es más pequeño, lo que es adecuado ya que significa que se ha pasado por un mínimo y que los pesos deben ser menores para poder alcanzarlo.

Estructura y aprendizaje de la red backpropagation

Presenta una capa de entrada con n neuronas y una capa de salida con m neuronas y al menos una capa neuronas ocultas internas. Cada neurona (menos en la capa de entrada) recibe entrada de todas las neuronas de la capa previa y genera salida hacia

todas las neuronas de la capa siguiente (salvo las de salida). No hay conexiones hacia atrás, feedback, ni laterales o autorecurrentes.

Pasos y fórmulas por utilizar para aplicar el algoritmo de entrenamiento:

Paso 1: Se inicializan los pesos de la red con valores pequeños y aleatorios.

Paso 2: Se presenta un patrón de entrada: $Xp = x_{p1}, x_{p2}, \ldots, x_{pn}$ y se especifica la salida deseada: $d_1, d_2, \ldots, d_M$ (si se utiliza como clasificador, todas las salidas deseadas serán 0, salvo una, que es la de la clase a la que pertenece el patrón de entrada).

Paso 3: Se calcula la salida actual de la red, para ello se presentan las entradas y se van calculando las salidas que presenta cada capa, hasta llegar a la capa de salida, $y_1, y_2, \cdots, y_M$.

Luego

• Se calculan las entradas netas para las neuronas ocultas procedentes de las neuronas de entrada.

$$net_{pj}^{h} = \sum_{i=1}^{N} w_{ij}^{h} x_{pi} + \Theta_j^h$$culta:

Donde el índice h se refiere a magnitudes de la capa oculta (hidden), el subíndice p, al $p-ésimo$ vector de entrenamiento, y la $j-ésima$ neurona oculta.

- Se calculan las salidas de las neuronas ocultas:

$$y_{pj} = f_j^h(net_{pj}^h)$$

- Se realizan los mismos cálculos para obtener las salidas de las neuronas de salida (o capa)

$$net_{pk}^{0} = \sum_{j=1}^{L} w_{jk}^{0} y_{pj} + \Theta_k^o$$

$$y_{pj} = f_k^o(net_{pk}^0)$$

Paso 4: Calcular los términos de error para todas las neuronas.

Si la neurona k es una neurona de la capa de salida, el valor de delta es:

$$\delta_{pk}^{0} = (d_{pk} - y_{pk}) f_k^0(net_{pk}^0)$$

La función f debe ser derivable, para ello existen dos funciones que pueden servir: la función lineal $(f_k(net_{jk}) = net_{jk})$ y la función sigmoidal:

$$f_k(net_{jk}) = \frac{1}{1 + e^{-net_{jk}}}$$

La elección de la función de salida depende de la forma de representar los datos: Si se desean salidas binarias, se emplea la función sigmoidal, (tal como se realizó en el presente artículo) en otro caso es tan aplicable una como la otra.

Para la función lineal tenemos $f_k^{o'} = 1$, mientras que la derivada de una función sigmoidal es:

$$f_k^{o'} = f_k^o(1 - f_k^o) = y_{pk}(1 - y_{pk})$$

Por lo que el término de error para las neuronas de salida lineal queda:

$$\delta_{pk}^o = d_{pk} - y_{pk}$$

Y para la salida sigmoidal:

$$\delta_{pk}^o = (d_{pk} - y_{pk})y_{pk}(1 - y_{pk})$$

Si la neurona j no es de salida se tiene:

$$\delta_{pj}^h = f_j^{h'}\left(net_{j_{pj}^h}\right)\sum_k \delta_{pk}^o w_{pk}^o$$

Donde observamos que el error en las capas ocultas depende de todos los términos de error de la capa de salida. De aquí surge el término de propagación hacia atrás. Para la función sigmoidal:

$$\delta_{pj}^h = x_{pi}(1 - x_{pi})\sum_k \delta_{pk}^o w_{pk}^o$$

Donde k se refiere a todas las neuronas de la capa superior a la de la neurona j. Así, el error que se produce en una neurona oculta es proporcional a la suma de los errores conocidos que se producen en las neuronas a las que está conectada la salida de ésta, multiplicados por el peso de la conexión.

Paso 5: Actualización de los pesos.

Se utiliza el algoritmo recursivo, comenzando por las neuronas de salida y trabajando hacia atrás hasta llegar a la capa de entrada, ajustando los pesos de la forma siguiente:

Para la capa de salida:

$$w_{jk}^o(t+1) = w_{jk}^o(t) + \Delta w_{jk}^o(t+1)$$

$$\Delta w_{jk}^o(t+1) = \alpha\delta_{pk}^o y_{pj}$$

y para los pesos de las neuronas de la capa oculta:

$$w_{ij}^h(t+1) = w_{ij}^h(t) + \Delta w_{ij}^h(t+1)$$

$$\Delta w_{ij}^h(t+1) = \alpha\delta_{pj}^h x_{pi}$$

En ambos casos, para acelerar el proceso de aprendizaje, se puede añadir un término momento de valor en el caso de neurona de salida:

$$\beta\left(w_{jk}^o(t) - w_{jk}^o(t-1)\right)$$

Y en el caso de neurona oculta:

$$\beta\left(w_{ij}^h(t) - w_{ij}^h(t-1)\right)$$

Paso 6: El proceso se repite hasta que el término de error

$$E_p = \frac{1}{2}\sum_{k=1}^{M} \delta_{pk}^2$$

Resulta aceptablemente pequeño para cada uno de los patrones aprendidos (García Martínez *et al*, 2003).

Adicionalmente, es necesario comparar algunos de los métodos de clasificación de aprendizaje supervisado para identificar el más acertado al momento de clasificar con las características que se cuentan en el conjunto de datos.

El método de vecinos más cercanos (KNN, por sus siglas en inglés k-nearest-neighbor) está basado en el aprendizaje por analogías. Cuando se quiere clasificar un nuevo punto o tupla, se hace una comparación de la tupla de prueba a clasificar con tuplas previamente entrenadas y clasificadas que sean similares en sus atributos. Cada tupla representa un punto en un espacio n-dimensional. Cuando se presenta una tupla desconocida, el clasificador KNN ubica la tupla en el espacio dependiendo sus atributos y busca a las tuplas entrenadas más cercanas en términos de distancia métrica. La nueva tupla será clasificada dependiendo el número de vecinos más cercano a la nueva tupla. (Han, Kamber, & Pei, 2012)

Esta técnica se basa, simplemente, en "recordar" todos los ejemplos que se vieron en la etapa de entrenamiento. Cuando un nuevo dato se presenta al sistema de aprendizaje, este se clasifica según el comportamiento del dato más cercano (Aha *et al*., 1991; Moreno, 2004).

A manera de ejemplo, se puede observar cómo es la clasificación del vecino más cercano. Se tienen los datos pertenecientes al conjunto de entrenamiento, tal como se muestra en la Figura 1 (triángulos y cuadrados), y se quiere conocer la etiqueta de un nuevo dato (marcado como *x*). Entonces el procedimiento a seguir con consiste en buscar el ejemplo que esté más cerca de este nuevo dato *x*, y asignarle su etiqueta (triángulo), tal como se señala en la Figura 2.

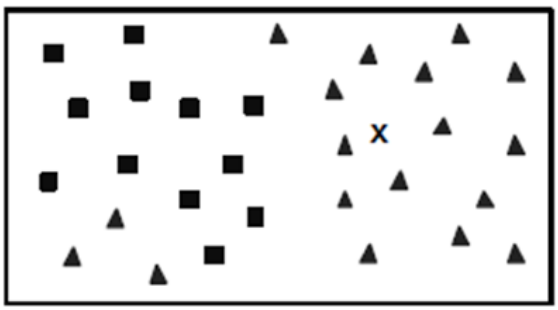

Figura 1

Ubicación de un dato nuevo en KNN

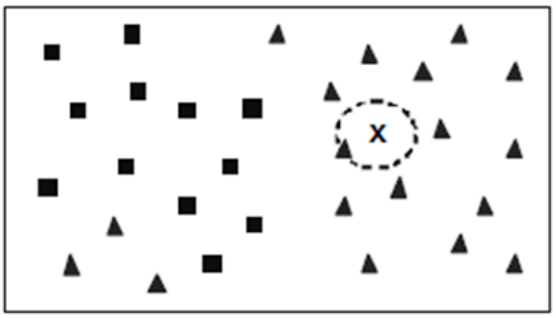

Figura 2

Predicción de la clase de un dato nuevo con KNN.

Ahora, si se considera el caso donde hay un cuadrado dentro de los datos correspondientes a los triángulos (ruido), y se desea clasificar el nuevo dato (x), utilizando el ejemplo más cercano tal como se muestra en la Figura 3, se tiene un posible error.

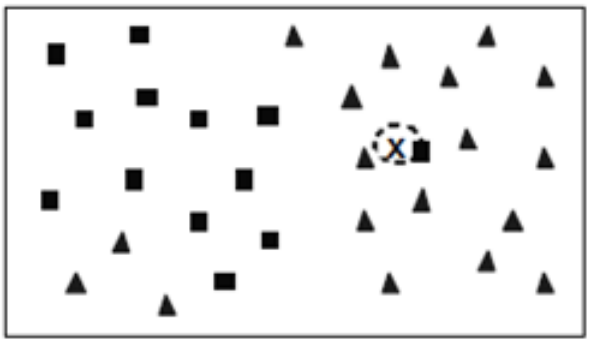

Figura 3

Predicción de la clase de un dato nuevo con KNN, los datos conocidos contienen ruido.

Se puede notar que debido al ruido, el nuevo dato se clasifica como cuadrado. Para considerar el problema del ruido, se puede cambiar el algoritmo de clasificación y utilizar un mayor número de vecinos, y así generar la etiqueta del nuevo dato usando mayoría simple, y no un solo dato. Esta generalización del método se llama *k*vecinos más cercanos (Moreno, 2004). En este caso se hace k=5 y se puede observar que el nuevo dato pertenece a la clase triángulos, tal como gráficamente se muestra en la Figura 4.

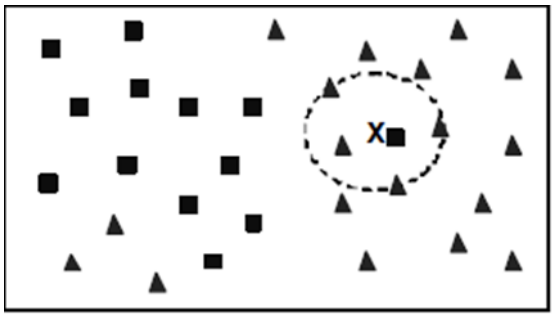

Figura 4

Predicción de la clase de un dato nuevo con KNN, los datos conocidos contienen ruido.

Con este nuevo enfoque se consigue resolver el problema del ruido. Entre más grande es *k*, más robusta es la clasificación ante ruido. Sin embargo, el valor de *k* tiene un límite, si se hiciera máximo cualquier dato nuevo siempre se tendrá la etiqueta de la clase que más datos haya en el conjunto de entrenamiento (Aha *et al*., 1991). Por ejemplo, para el caso presentado en la Figura 4, si se asigna k=31 los datos nuevos siempre serán clasificados como triángulos, debido a que se tienen 18 triángulos y 13 cuadrados.

Para la estimación de la distancia se utilizan dos estrategias conocidas como distancia euclidiana y distancia Mahalanobis, las cuales se presentan en su forma vectorial para una distancia entre dos vectores en (1) y (2), respectivamente.

$$D_{Euclideana}(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^T(\vec{x}-\vec{y})}$$

$$D_{Mahalanobis}(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^T\sum^{-1}(\vec{x}-\vec{y})}$$

Por otra parte, la clasificación Bayesiana está basada en el teorema de Bayes y es llamada Naive Bayes. Este método ha mostrado alta precisión y velocidad cuando se aplica a grandes bases de datos. El clasificador asume que el efecto del valor de un atributo en una clase dada, es independiente de los valores de los otros atributos. Esta suposición es llamada independencia condicional de clase y ayuda a simplificar los cálculos involucrados. (Han, Kamber, & Pei, 2012)

De acuerdo con Malagón (Malagón, 2013), dado un ejemplo x representado por k valores, el clasificador Naive Bayes se basa en encontrar la hipótesis más probable que describa a ese ejemplo. Si la descripción de ese ejemplo viene dada por los valores $(a_1, a_2, .., a_n)$, la hipótesis más probable será aquella que cumpla:

$$v_{MAP} = \arg max_{(v_j \in V)} P(v_j | a_1, a_2, .., a_n)$$

Podemos estimar $P(v_j)$ contando las veces que aparece el ejemplo v_j en el conjunto de entrenamiento y dividiéndolo por el número total de ejemplos que forman este conjunto. Para estimar el término $P(a_1, a_2, .., a_n|v_j)$, es decir, las veces en que para cada categoría aparecen los valores del ejemplo x, debo recorrer todo el conjunto de entrenamiento. Este cálculo resulta impracticable para un número suficientemente grande de ejemplos, por lo que se hace necesario simplificar la expresión. Para ello se recurre a la hipótesis de independencia condicional con el objeto de poder factorizar la probabilidad. Esta hipótesis dice lo siguiente:

Los valores a_j que describen un atributo de un ejemplo cualquiera x son independientes entre sí, conocido el valor de la categoría a la que pertenecen Así la probabilidad de observar la conjunción de atributos a_j dada una categoría a la que pertenecen, es justamente el producto de las probabilidades de cada valor por separado:

$$P(a_1, a_2, .., a_n|v_j) = \prod_i P(v_j|a_i)$$

Se requiere un modelo de clasificación capaz de identificar perfiles en redes sociales en alguno de los grupos definidos para carreras universitarias, basándose en las preferencias de personalidades y gustos en redes sociales de cada uno de los usuarios.

Desarrollo metodológico

Como primera etapa, se recaba la información de los usuarios de redes sociales y sus preferencias en Facebook a través de dos herramientas, como cuestionarios de Google (https://goo.gl/forms/3tIvwkfmhTCf1vnh1) y encuestas impresas personales con exactamente las mismas preguntas.

Al mismo tiempo, a dichos usuarios de redes sociales se les aplica un test de personalidad MBTI (Briggs-Myers, 1985) (Briggs-Myers, 1985) para conocer sus perfiles de personalidad. Todas las muestras se integran en una sola base para evaluarlas.

Una vez que se tiene completa la información en una sola base de datos, se realiza una selección, exploración y limpieza de los datos recabados a través de las encuestas personales, ya que algunas no fueron llenadas en su totalidad y no pueden ser evaluadas correctamente.

Se decide trabajar con una de las cuatro preferencias de personalidad que analiza el estudio MBTI en vez de las 16 personalidades, debido a que si se trabaja con 16 salidas, cada target tiene pocas tuplas representativas, y el entrenamiento y validación no podría realizarse de manera exitosa; en su lugar, si se trabaja con una de las cuatro preferencias, la RNA tendrá sólo dos salidas. Las cuatro preferencias son: preferencia de socialización, preferencia para reunir información, preferencia para tomar decisiones y preferencia para organizar la vida. En este caso, trabajaremos con las preferencias para tomar decisiones: racional o emocional.

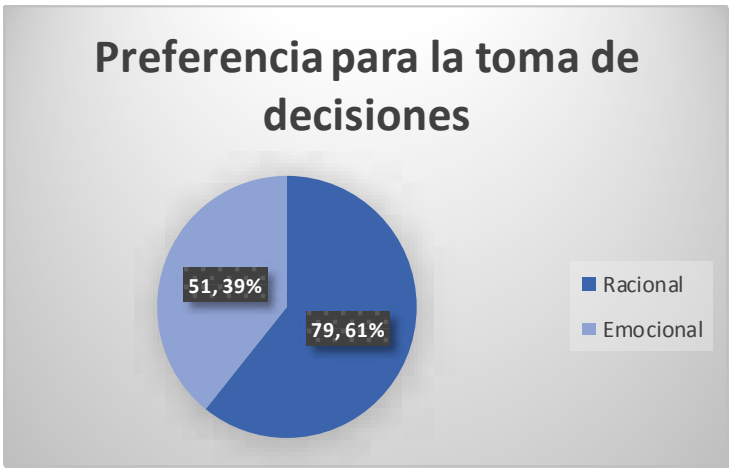

Figura 5

Distribución de la información.

La Tabla 1 nos muestra las mejores carreras de acuerdo con cada personalidad y su preferencia para tomar decisiones (Snyder, 2017)

Tabla 1
Mejores carreras por personalidad y preferencia para tomar decisiones.

Personalidad	Preferencia para tomar decisiones	Mejor carrera
ESTJ	Racional	Chef
ISTJ	Racional	Systems administrator
ESFJ	Emocional	Registered nurse
ISFJ	Emocional	Kindergarten teacher
ESTP	Racional	Military officer
ISTP	Racional	Police officer
ESFP	Emocional	Bartender
ISFP	Emocional	Jeweler
ENTJ	Racional	Physician
INTJ	Racional	Microbiologist

ENFJ	Emocional	Minister
INFJ	Emocional	Veteriniarian
ENTP	Racional	Reporter
INTP	Racional	College professor
ENFP	Emocional	Landscape architect
INFP	Emocional	Fine artist

Una vez terminada la selección y limpieza de datos, se tiene una base informativa con los gustos en redes sociales y los tipos de preferencias según la personalidad de cada usuario.

Con esta información se procede a entrenar, validar y probar la red neuronal.

Preferencias de socialización	Preferencia para reunir información	Preferencia para tomar decisiones	Preferencia para organizar la vid	Cla	Top Comunida	Negoci	Medios de c	Artistas	Marcas	Top Empres	Usuario
1	4	5	7	1457	6	5	4	3	2	1	[redacted]@gmail.com
2	3	6	8	2368	6	4	5	2	1	3	[redacted]@gmail.com
1	4	6	8	1468	5	3	6	1	4	2	[redacted]IL.COM
2	4	5	8	2458	0	3	5	0	6	1	[redacted]@gmail.com
2	3	6	8	2368	6	2	3	4	0	5	[redacted]gmail.com
1	3	5	8	1358	5	3	6	2	4	1	[redacted]@gmail.com
2	4	5	8	2458	6	4	5	1	2	3	[redacted]@gmail.com
1	3	5	7	1357	4	5	6	1	2	3	[redacted]@gmail.com
2	3	5	7	2357	6	2	4	3	0	1	[redacted]gmail.com
2	3	5	7	2357	6	0	0	0	0	0	[redacted]@gmail.com
2	4	5	7	2457	6	0	4	1	2	5	[redacted]hotmail.com
1	4	5	8	1458	4	5	3	1	2	6	[redacted]@hotmail.com
2	3	5	8	2358	2	0	0	6	3	4	[redacted]@hotmail.com
2	4	6	7	2467	4	0	0	6	0	5	[redacted]@gmail.com
2	4	6	8	2468	6	5	3	2	4	1	[redacted]@hotmail.com
1	4	5	8	1458	2	5	6	3	4	1	[redacted]@gmail.com
2	4	6	8	2468	0	4	3	0	1	2	[redacted]romasinformatica.com.mx

Figura 6

Muestra de los datos.

Para la creación de la red neuronal nos apoyamos del software MatLab, en particular del asistente nprtool. Se hacen pruebas de entrenamiento donde los mejores resultados arrojan redes entrenadas con una precisión del 93.8%.

La red neuronal fue diseñada con una arquitectura 6:23:2, lo que quiere decir que consta de:

- 6 neuronas de entradas: las cuales representan los pesos para cada una de las preferencias o tipos de páginas en la red social Facebook.
- 23 neuronas en la capa oculta con una función signoidal.
- 2 neuronas de salida: racional o emocional. Representan su preferencia para la toma de decisiones; si el usuario tiende más a lo racional o lo emocional al momento de decidir.

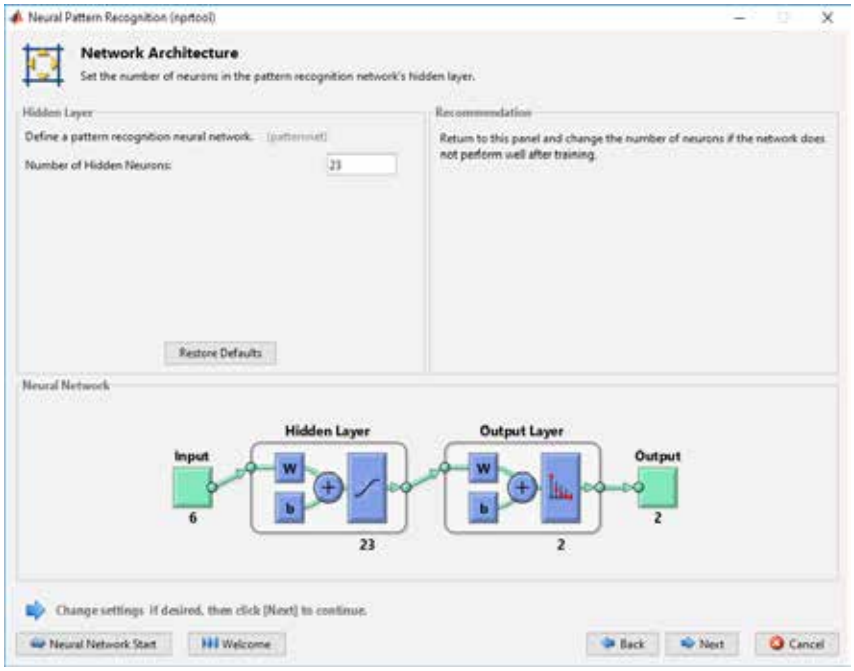

Figura 7

Arquitectura de la RNA.

La red se alimenta con una matriz de 6x130 para sus entradas y con una matriz de 2x130 para los objetivos.

La retro-propagación se aplica en la RNA que aprende procesando de manera iterativa un conjunto de datos de entrenamiento. Para cada tupla los pesos son modificados con el fin de minimizar el error medio cuadrático entre la predicción de la RNA y valor objetivo real. Dichas modificaciones de peso son hechas en dirección "hacia atrás" desde la capa de salida a la capa oculta. (Han, Kamber, & Pei, 2012)

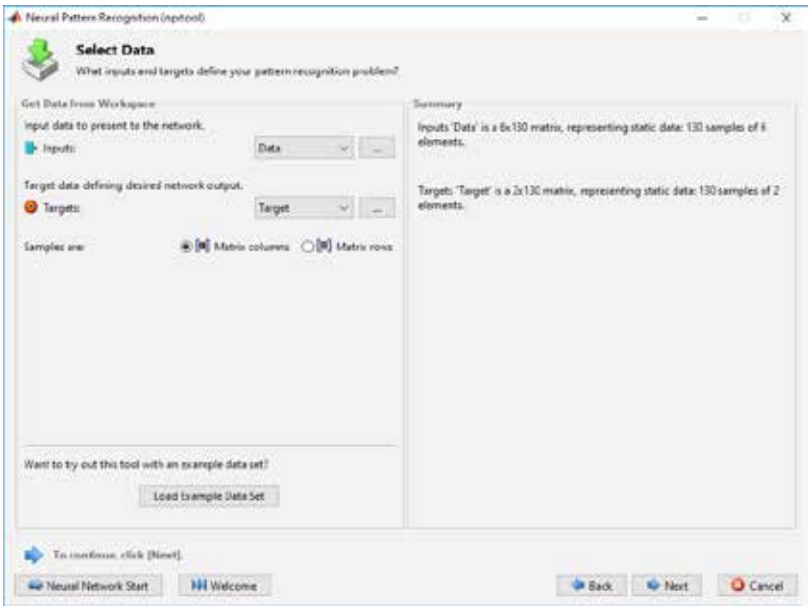

Figura 8

Matrices de entrada y objetivos.

Las muestras de datos son divididas en los siguientes porcentajes para cada una de las etapas de entrenamiento de la RNA.

Tabla 2
Porcentaje de división de muestras

Etapa	Porcentaje	Número de muestras
Entrenamiento	70%	90
Validación	15%	20
Pruebas	15%	20

Una vez entrenada en su totalidad la red neuronal, tenemos que la RNA tiene los siguientes porcentajes de error cuadrático de clasificación: 5.5% en el entrenamiento, 5% en la validación y 10% en las pruebas, obteniendo un porcentaje total de clasificación del 93.8%. Entre menor sea el porcentaje de error cuadrático, mayor es la efectividad de la RNA para clasificar.

En la tabla 3 se muestra que el proceso de aprendizaje se realizó en 18 épocas y en la tabla 4 vemos los algoritmos utilizados.

Figura 9

Matriz de confusión.

Tabla 3
Progreso del entrenamiento de la RNA.

Épocas	18
Performance	0.0341
Gradient	0.0304
Validation Checks	6

Tabla 4
Algoritmos de la RNA

Data Division	Random
Training	Scaled Conjugate Gradient
Performance	Cross-Entropy
Calculations	MEX

En la Figura 10 observamos el comportamiento que tuvo la red neuronal en su entrenamiento con respecto a los falsos positivos.

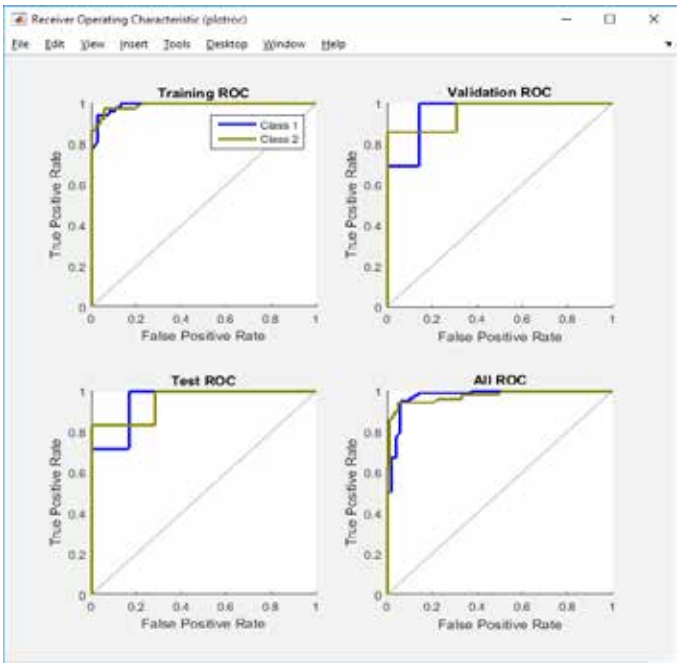

Figura 10

Gráfica ROC.

Una vez terminado el entrenamiento de la RNA se procede a comprobar los resultados, para ello ingresamos muestras de la base de datos. Se empieza primero con tres ejemplos del objetivo 1. Como se muestra en la siguiente figura, los resultados de clasificación son correctos.

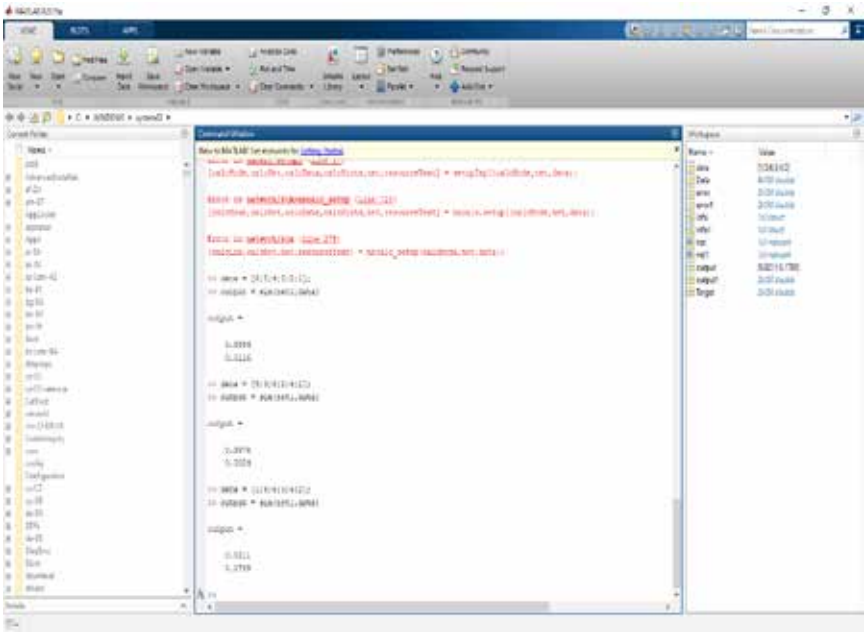

Figura 11

Prueba de clasificación con objetivo 1.

Ahora, se procede a hacer pruebas de clasificación con datos del objetivo 2.

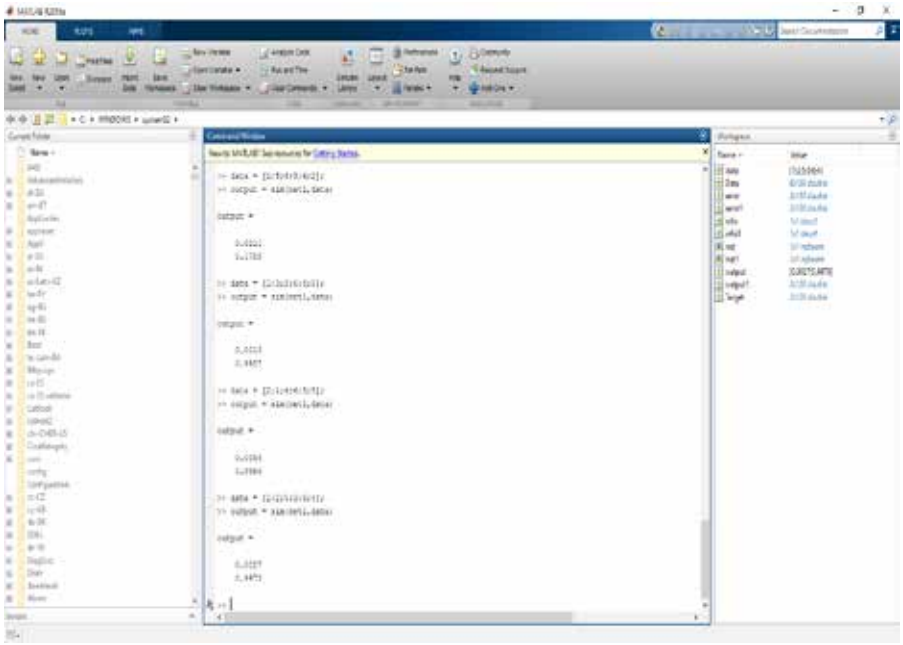

Figura 12

Prueba de clasificación con objetivo 2.

Con estos resultados, podemos determinar que la clasificación de la RNA de los perfiles basados en sus preferencias en redes sociales, es correcta.

Comparativa de métodos de clasificación

Como comparativa de los resultados obtenidos con la RNA, se aplicaron otros dos métodos de clasificación de aprendizaje supervisado desarrollado en Python.

El primer método probado fue el de vecinos más cercanos KNN (por sus siglas en inglés K-Nearest Neighbors), el cuál es suministrado con la misma información que la RNA, es decir, las preferencias de los usuarios en la red social Facebook y la preferencia según su personalidad. Del conjunto total de datos, se usa el 25% para las pruebas.

Los resultados de la clasificación de éste primer método fueron del 95% de efectividad (19 de 20 casos) para el primer objetivo y del 70% (9 de 13 casos) para el segundo.

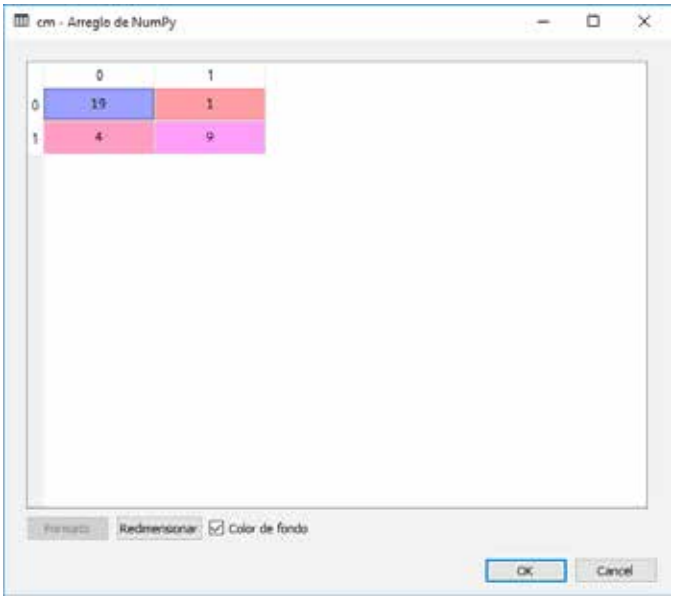

Figura 13

Matriz confusión KNN

El segundo método comparativo es NaiveBayes, al cual también se carga la misma información en la data set y se ocupa el 25% de ella para pruebas.

Los resultados de la clasificación para Bayes fueron del 85% de efectividad (17 de 20 casos) para el primer objetivo y del 85% (11 de 13 casos) para el segundo.

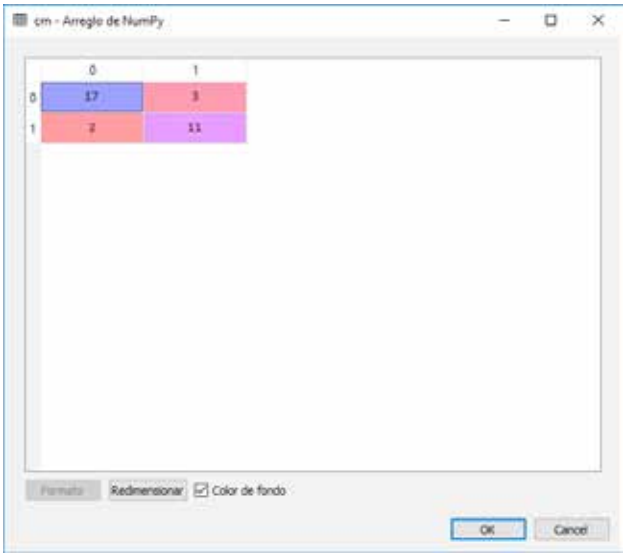

Figura 14

Matriz confusión NaiveBayes

Al final, comparando los tres métodos obtenemos el resumen que se muestra en la Tabla 5.

Tabla 5
Comparativa de métodos de aprendizaje supervisado

Método	% Efectividad Objetivo 1	% Efectividad Objetivo 2	% Efectividad Total
RNA			93.8%
KNN	95	70	82.5%
Naive Bayes	85	85	85%

Conclusión

Como se pudo comprobar en el desarrollo de este trabajo, la RNA se puede entrenar satisfactoriamente para clasificar perfiles de usuarios de acuerdo a sus preferencias en redes sociales y, con ello, brindar información a las instituciones educativas para ofrecer sus servicios a estudiantes potenciales que cuenten con el perfil adecuado a sus ofertas educativas.

El método de clasificación de aprendizaje supervisado que resultó ser más acertado en este trabajo fue la RNA. La retro-propagación ayuda a ir ajustando los pesos de las neuronas, de tal manera que se acerca más a los valores objetivos.

Aunque los métodos de clasificación y minería de datos están diseñados para grandes cantidades de información, notamos que cuando las clases objetivos se diferencian de manera marcada en sus atributos, es decir, que son muy distintas, la clasificación de puntos se realiza de manera eficiente, a pesar de no contar con un gran número de datos en la muestra, como fue el caso de este trabajo.

Trabajos futuros

Durante el desarrollo del trabajo no se pudo lograr una mayor eficiencia de clasificación, por lo que sería pertinente tener presentes dos consideraciones: la primera será incluir otras variables significativas como neurona de entrada que ayuden a distinguir mejor los perfiles de los usuarios. La otra, es obtener un mayor número de muestras.

Un trabajo adicional a futuro, es crear un agente que se ejecute en las redes sociales, para recolectar la información de los perfiles en automático sin necesidad de encuestar a los usuarios directamente; además, aprovechando los modelos de clasificación previamente creados, como el de trabajo, se podrá tener la personalidad de los usuarios, lo cual ayudará a que el número de la muestra obtenida sea mayor.

Agradecimientos

Deseamos agradecer a los profesores y directivos del Tecnológico de Estudios Superiores de Ecatepec por el conocimiento impartido, dedicación y empeño que otorgan y que nos ayuda a llegar a trabajos como el presente.

Así también, se extiende el agradecimiento a los compañeros del posgrado, ya que su esfuerzo y apoyo son una motivación para seguir esforzándose día a día.

Finalmente, a cada una de las personas que integran nuestra familia, que son el principal motivo de superación y felicidad.

Referencias

(2017, mayo 30). (Cómite de la Infraestructura de la asociación de Internet Mx) Retrieved abril 20, 2018, from https://www.asociaciondeinternet.mx/es/noticiasx/2315-adopcion-de-ipv6

Briggs-Myers, I. (1985). *MBTI Manual: A Guide to the Development and Use of the Myers-Briggs Type Indicator*. Consulting Psychologists Press.

Dunham, M. (2003). *Data Mining. Introductory and Advanced Topics*. New Jersey: Pearson Education, Inc.

ENDUTIH 2017. (2018, Febrero 20). (INEGI) Retrieved Mayo 23, 2018, from http://www.beta.inegi.org.mx/contenidos/saladeprensa/boletines/2018/OtrTemEcon/ENDUTIH2018_02.pdf

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining*. USA: Elsevier.

Kemp, S. (2018, enero 30). We are social. Retrieved abril 20, 2018, from https://wearesocial.com/blog/2018/01/global-digital-report-2018

Snyder, B. (2017, June 6). Here's the best job for you based on your personality type. Retrieved from CNBC: https://www.cnbc.com/2017/06/06/the-3-best-jobs-for-you-based-on-your-myers-brigg-personality-type.html

Aha, D., Kibler, D., Albert, M., Instance-based learning algorithms. Machine Learning, Springer Netherlands, 1991. https://link.springer.com/content/pdf/10.1007%2FBF00153759.pdf

Moreno, F., Clasificadores eficaces basados en algoritmos rápidos de búsqueda del vecino más cercano. Departamento de lenguajes y sistemas informáticos. Universidad de Alicante, 2004. https://rua.ua.es/dspace/bitstream/10045/11790/1/Moreno-Seco-Francisco.pdf

García Martínez, R.; Servente, M.; Pasquín, D.; 2003. Sistemas Inteligentes, Capítulo 1: "Aprendizaje Automático", Capítulo 2 "Redes Neuronales Artificiales"; Nueva Librería, Buenos Aires, Argentina.