ISBN: 978-628-95207-4-3. ISSN: 2414-6390. Digital Object Identifier: https://dx.doi.org/10.18687/LACCEI2023.1.1.1332

Machine learning application in university management: Classification model Dropping out of engineering students in Peru.

Paul Tocto, Doctor[1], Gloria Teresita Huamaní, Doctora[2], Luis Zuloaga, Magister[3]
[1,2,3]Universidad Nacional de Ingeniería, Perú, ptocto@uni.edu.pe, gloria.huamani.h@uni.edu.pe, zuloaga_luis@uni.edu.pe

***Abstract*– We apply artificial intelligence to various cases of university management, with a proactive approach. In this study, we apply machine learning to classify whether a student will drop out or not, considering certain variables from the SITUATION DATA, based on the relevant attributes of the students. The results of this study would serve decision markers as a basis of inquiring about the high cost of not finishing the degree and adopting retention strategies. On the other hand, students can perceive the benefits of education, especially in engineering careers by having expectations, and value or assessment of it. The model developed is a neural network with 8 internal layers, in addition to the input and output layers, 225 training iterations have been considered, obtaining as a result an accuracy of 67.10%.**

***Keywords.* Machine learning, neural network, classification, student dropout**

Digital Object Identifier: (only for full papers, inserted by LACCEI).
ISSN, ISBN: (to be inserted by LACCEI).
DO NOT REMOVE

Aplicación aprendizaje automático en la gestión universitaria: Modelo de clasificación de la deserción de los estudiantes en Ingeniería en el Perú.

Paul Tocto, Doctor[1], Gloria Teresita Huamaní, Doctora[2], Luis Zuloaga, Magister[3]
[1,2,3]Universidad Nacional de Ingeniería, Perú, ptocto@uni.edu.pe, gloria.huamani.h@uni.edu.pe, zuloaga_luis@uni.edu.pe

Resumen–. Aplicamos inteligencia artificial a diversos casos de la gestión universitaria, con sentido de proactividad, en este estudio aplicamos el aprendizaje automático para clasificar si un estudiante dejará de estudiar o no considerando ciertas variables de la DATA-SITUACIÓN, en función de los atributos relevantes de los estudiantes. Los resultados de este estudio servirían a los tomadores de decisiones, como base para indagar acerca del alto costo que significa no terminar la carrera y adoptar estrategias de retención. Por otra parte, los estudiantes pueden percibir los beneficios de la educación en especial en las carreras de ingeniería, al tener expectativas y valoración sobre la misma. El modelo hallado es una red neuronal de 8 capas internas, además de la capa de entrada y de salidas, se ha considerado 225 iteraciones de entrenamiento, obteniéndose como resultado una precisión de 67.10%.

Palabras clave. Aprendizaje automático, red neuronal, clasificación, deserción.

I. Introducción

Garantizar la utilización de las tecnologías de la Inteligencia artificial (IA) en el contexto educativo bajo los principios fundamentales de inclusión y equidad, es uno de los objetivos de la UNESCO, por ello, "está decidida a ayudar a los Estados Miembros para que saquen provecho del potencial de las tecnologías de la IA con miras a la consecución de la Agenda de Educación 2030". [1] En esa línea, proponemos la inteligencia artificial en la gestión universitaria. Acerca de la oferta educativa, está previsto completar los estudios en 5 años, sin embargo, otros estudios dan cuenta de que en un porcentaje se demoran más tiempo [2][3]. En esta oportunidad nuestro estudio consiste en diseñar un modelo de aprendizaje automático de clasificación, para detectar los potenciales alumnos que podrían dejar de estudiar. El aprendizaje automático sirve para clasificar si un estudiante dejará de estudiar o no, en función de los atributos relevantes de dichos estudiantes. A partir de los resultados obtenidos, la proactividad de las autoridades y tomadores de decisiones, podrían tomar acciones sobre los potenciales alumnos que dejarrían de estudiar, considerando que cada alumno universitario es una inversión económica que el estado está realizando, y si deja los estudios la inversión realizada se perdería. Los beneficios de la educación para el estudiante pueden depender también en buena medida de la calidad de la educación impartida. En la línea de [4][5] es posible analizar de forma detallada las causas de la deserción: aspectos económicos, laborales, familiares, personales, percepciones sobre la oferta educativa, expectativas y valoración de la carrera profesional. Sin perder de vista las preguntas referidas por [6][7] las mismas que servirían para futuras investigaciones: ¿Cuáles son las características de los jóvenes que abandonaron el sistema educativo respecto de aquellos que aún asisten a la Universidad? ¿En qué momento, en qué semestre abandonan los estudios?, ¿existen diferencias en las causas de la deserción asociadas a género con el abandono de los estudios? ¿El nivel educativo de los padres puede ser un indicador de la valoración que éstos tienen hacia la educación que reciben sus hijos? En este proyecto buscamos predecir si un alumno es un potencial desertor.

Deserción estudiantil

Para este estudio sobre la deserción de los estudios de ingeniería, se considera al acto de dejar un programa de ingeniería antes de completar los requisitos para el título, acorde a los datos de los estudiantes de la Facultad de Ingeniería Industrial y de Sistemas, en donde aparece en el rubro de "situación" ver Tabla II, "la suspensión voluntaria", que lo consideramos como deserción, a estos datos se denomina DATA-SITUACIÓN.

Una vez diseñado el modelo, podríamos explorar y tomar nota del diagnóstico, la identificación de los factores determinantes de la deserción estudiantil como las que señalan otras publicaciones. Respecto a la proporcionalidad es necesario conocer la tasa de graduados y tratar de comparar con otros trabajos. Entre 2011 y 2012, según [8] <<la tasa de graduación después del primer año de estudio fue del 80%, por lo tanto, se tiene una tasa de abandono del 20%. La tasa de deserción entre los estudiantes fue del 20% para los niños y del 15% para mujeres>>. Según [9]: "La tasa de deserción en universidades privadas puntuó 22.3% en el ciclo 2020-1 y 18.9 % en el ciclo 2020-2, según el sistema de información de la educación superior del Minedu". Adicionalmente "La tasa de deserción universitaria llegó al 18,6% en el país, seis puntos porcentuales más respecto al 2019 que son atribuibles a la pandemia del COVID-19. [10] Según el Minedu, el Estado pudo evitar un mayor número de retiros mediante el otorgamiento de becas y distribución de chips. Desde hace un tiempo PRONABEC entrega becas a los estudiantes tanto de universidades públicas como privadas [11], desde el 2012, PRONABEC ha entregado un total de 71,205 becas 18, en sus ocho modalidades. Sin embargo, 17,825 jóvenes (25%) no concluyeron la carrera. Según Cotler [12] "los becarios son

conscientes de los impactos que tiene este programa de becas tanto en términos económicos como sociales. Sin embargo, la forma en que se materializa el objetivo o "fin último" que reconocen, más allá de los objetivos formales del programa, varían de forma considerable".

A. Aprendizaje automático

La detección temprana de una posible deserción de estudiantes, y tomar algunas medidas correctivas para retener y mejorar su desempeño académico es necesario como afirman Pasic y Kukac [13] "Los estudiantes que serían clasificados por el modelo como no exitosos recibirán una recomendación para otro programa de estudio".

En el entrenamiento de datos no se considera la influencia de las variables socioeconómicas, entorno familiar en el desempeño académico, como si lo hacen García y Skarita [14]. Por otra parte <<los modelos de clasificación para usuarios de sistemas web se apoyan en la definición de grupos a priori para que a través de un modelo de aprendizaje automático y con entradas de información conocidas se pueda clasificar la unidad de estudio en un grupo determinado.>> afirma De La Hoz et al. [15] Para Contreras et al. [16] implementaron algoritmos de clasificación a través del lenguaje de programación Python como árbol de decisión, K vecinos más cercanos, perceptrón y otros, los cuales son comparados para conocer el mejor resultado de predicción. Por otra parte, según Sharma y Yadav [17] [18] la solución propuesta, basada en Machine Learning (aprendizaje profundo) tiene la ventaja de adoptar una medida proactiva, de modo que la facultad y los padres de familia pueden identificar

Las limitaciones del caso han sido la obtención de datos y tener un tiempo limitado para explorar, extraer los datos y prepararlos. La elección y configuración del aprendizaje automático de clasificación ha sido tratado en 8 capas. Dada la experiencia de anteriores estudios, facilitó el entrenamiento y la evaluación o validación del modelo seleccionado.

En un inicio contamos con 3141 datos de alumnos universitarios, organizados en campos y el que sirve a nuestro propósito es la DATA SITUACIÓN, explicado en la sección II.

Además de esta sección presentamos la colección de datos y definición de variables, metodología, experimento, análisis de resultados, conclusiones y recomendaciones.

II. COLECCIÓN DE DATOS Y DEFINICIÓN DE VARIABLES

Los datos corresponden a todos los alumnos ingresantes desde el año 2010 hasta el 2022, que denominaremos DATA SITUACIÓN, de las dos especialidades (Ingeniería Industrial e Ingeniería de Sistemas) de la Facultad de Ingeniería Industrial y de Sistemas (FIIS), de la Universidad Nacional de Ingeniería, del Perú. De un total de 3141 registros de alumnos, se realizó la depuración de los valores atípicos usando el diagrama botplox, ver Fig. 1 donde se visualiza el análisis de una de las variables ("tipo de colegio"). Después de la depuración quedaron 2638 registros para realizar el entrenamiento de la red neuronal, La Tabla I especifica las 10 variables independientes y la variable dependiente "Situación del alumno" de la DATA SITUACIÓN.

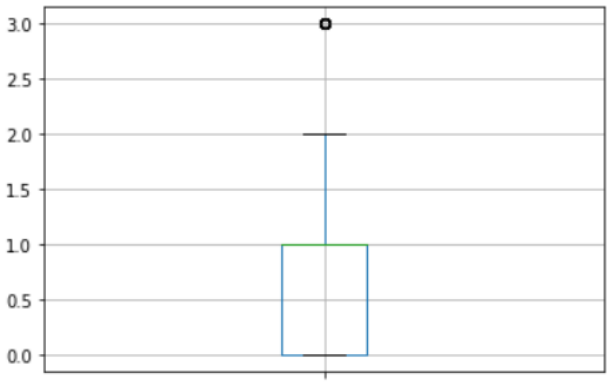

Fig. 1 Botplox de la variable independiente, tipo de colegio.

TABLA I
VARIABLES DEL MODELO

Nro	Atributo	Descripción
1	ESPECIA	Especialidad (I1, I2)
2	SEXO	Sexo (MASCULINO, FEMENINO)
3	NACIONALIDAD	Nacionalidad
4	DEP.NACIMIENTO	Departamento de Nacimiento.
5	TIPCOLEGIO	Tipo de colegio
6	DATOSLABOR	Datos laborales (No trabaja, Trabaja, Practicante)
7	DEP.COLEGIO	Departamento del Colegio
8	VIVECONPADR	Vive con sus padres (SI, NO)
9	DEP.RESIDENCIA	Departamento de Residencia
10	EDADINGRESO	Edad de Ingreso a la Universidad
11	SITUACIÓN	Situación del alumno

III. METODOLOGÍA

Se usó una Red Neuronal de clasificación, para predecir la situación de un alumno al estudiar su carrera en la FIIS, ver Tabla II. La arquitectura de la red neuronal es de una red neuronal de 8 capas ocultas (ver Fig. 2).

TABLA II
SITUACIÓN DE UN ALUMNO

Suspensión Voluntaria
Bachiller
Egresado
Titulado
Separado Definitivamente.
Suspendido (01 año)
Exceso Licencia
Alumno Regular
Inactivo
Currículo Completa
Ingresante

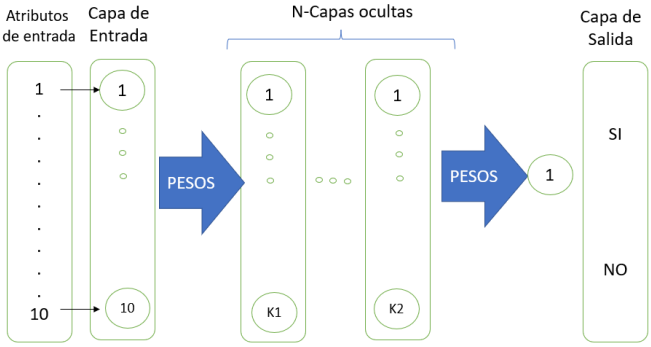

Fig. 2 Arquitectura de la red neuronal para hallar clasificar a un alumno

La preparación de los datos, entrenamiento y validación del modelo se realizó en la plataforma Colaboratory de Google, mediante los siguientes pasos:

Paso 1. Cargar todos los datos de los alumnos ingresantes desde el 2010 hasta el 2022 a la plataforma de Colaboratory de Google en la nube.

Paso 2. Se eliminaron las filas que no son datos y se renombraron los nombres de las columnas.

Paso 3. Se remplazaron los valores nulos mediante el valor más frecuente de cada columna.

Paso 4. Se crea una base de datos considerando solo las variables de la Tabla I.

Paso 5. Se realiza la codificación de las variables no numéricas: ESPECIA, SEXO, NACIONALIDAD, DEP.NACIMIENTO, TIPCOLEGIO, DATOSLABOR, DEP.COLEGIO, VIVECONPADR, DEP.RESIDENCIA, SITUACIÓN.

Paso 6. Se analiza los datos a partir del Botplox ver fig. 1 el caso de una de las variables "Tipo de Colegio", para identificar los valores extremos de cada una de las variables. Identificando los límites superior e inferior del Botplox. Ver Fig. 3.

```
import numpy as np
# Calcular el primer cuartil, tercer cuartil y el rango intercuartil (IQR)
q1 = np.percentile(datos_tot["Vive_con_padres_int"], 25)
q3 = np.percentile(datos_tot["Vive_con_padres_int"], 75)
iqr = q3 - q1

# Calcular los límites superior(ls) e inferior(li)
ls = q3 + 1.5 * iqr
li = q1 - 1.5 * iqr
```

Fig. 3 Cálculo de los límites superior e inferior del Botplox.

.

Paso 7. Se configura una definición de modelo de red neuronal en TensorFlow para el problema de clasificación con 11 clases, ver Fig. 4. El modelo consta de varias capas de neuronas, que se especifican utilizando la clase "tf. keras. layers". A continuación, describimos cada una de las capas:

Capa Dense con 64 neuronas y activación relu: Esta capa es la primera capa oculta del modelo y tiene 64 neuronas con una función de activación ReLU. Además, esta capa especifica la forma de entrada para la red, que es un vector de longitud 9.

Capa Dense con 128 neuronas y activación relu: Esta es otra capa oculta de la red, que tiene 128 neuronas con activación ReLU.

Capa Dense con 256 neuronas y activación relu: Esta capa es otra capa oculta de la red, que tiene 256 neuronas con activación ReLU.

Capa BatchNormalization: Esta capa normaliza los datos de entrada a lo largo de la dimensión de características. Esto ayuda a estabilizar la salida de cada capa y acelera la convergencia del modelo.

Capa Dropout con tasa de abandono de 0.2: Esta capa desactiva aleatoriamente algunas de las neuronas en la capa anterior con el objetivo de reducir el sobreajuste.

Capa Dense con 128 neuronas y activación relu: Esta es otra capa oculta de la red, que tiene 128 neuronas con activación ReLU.

Capa Dense con 256 neuronas y activación relu: Esta es otra capa oculta de la red, que tiene 256 neuronas con activación ReLU.

Capa Dense con 128 neuronas y activación relu: Esta es otra capa oculta de la red, que tiene 128 neuronas con activación ReLU.

Capa Dense con 11 neuronas y activación softmax: Esta es la capa de salida de la red, que tiene 11 neuronas correspondientes a las 11 clases de salida posibles. La activación softmax se utiliza para obtener una distribución de probabilidad sobre las clases de salida.

En resumen, el modelo usado es una red neuronal con varias capas ocultas y una capa de salida con activación softmax para el problema de clasificación con 11 clases. También incluye regularización mediante la capa Dropout para reducir el sobreajuste y BatchNormalization para estabilizar la salida de cada capa.

```
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu',
                          input_shape=(10,)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(11, activation='softmax')
    ])
```

Fig. 4 Configuración de la red neuronal con 10 nodos de entrada.

Paso 8. Se configura el proceso de entrenamiento model.compile() del modelo en TensorFlow. A través de esta configuración, especificamos el optimizador que se utilizará para ajustar los pesos del modelo, la función de pérdida que se utilizará para calcular la discrepancia entre las predicciones del modelo y las etiquetas de entrenamiento y las métricas que se utilizarán para evaluar el rendimiento del modelo durante el entrenamiento y la evaluación.

Se configuró con los siguientes argumentos:

optimizer='adam': Especifica el optimizador que se utilizó para ajustar los pesos del modelo. En este caso, el optimizador seleccionado es Adam, que es un algoritmo de optimización de gradiente estocástico que se utiliza comúnmente en el entrenamiento de redes neuronales.

loss='sparse_categorical_crossentropy': Especifica la función de pérdida que se utilizó para calcular la discrepancia entre las predicciones del modelo y las etiquetas de entrenamiento. En este caso, se utilizó la función de pérdida sparse_categorical_crossentropy, que se utiliza comúnmente en problemas de clasificación multiclase en los que las etiquetas de salida son valores enteros.

metrics=['accuracy']: Especifica las métricas que se utilizó para evaluar el rendimiento del modelo durante el entrenamiento y la evaluación. En este caso, se utiliza la métrica accuracy, que es la fracción de predicciones correctas del modelo para el conjunto de datos de entrenamiento y evaluación.

En resumen, esta configuración se utilizó para compilar un modelo de red neuronal en TensorFlow para el problema de clasificación con 11 clases. Utilizando el optimizador Adam para ajustar los pesos del modelo, la función de pérdida sparse_categorical_crossentropy para calcular la discrepancia entre las predicciones del modelo y las etiquetas de entrenamiento y la métrica accuracy para evaluar el rendimiento del modelo, ver Fig. 5

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Fig. 5 Configuración del proceso de entrenamiento.

IV. EXPERIMENTO

Se usó la función model.fit() para entrenar el modelo de red neuronal, que se usa en TensorFlow para ajustar los pesos de un modelo de red neuronal a un conjunto de datos de entrenamiento específico ver Fig. 6. Con los siguientes argumentos:

cat_data_dep: El primer argumento es un conjunto de datos de entrenamiento que se utiliza para entrenar el modelo. En este caso, cat_data_dep representa los datos de entrada (variables predictoras) del modelo.

train_labels: El segundo argumento representa las etiquetas de entrenamiento (o variables de respuesta) correspondientes a los datos de entrada en cat_data_dep.

validation_split=0.2: La opción validation_split es una forma de especificar qué proporción del conjunto de datos de entrenamiento se utilizará para la validación del modelo durante el entrenamiento. En este caso, el valor 0.2 indica que el 20% de los datos se utilizará para la validación y el 80% para entrenar.

epochs=225: El número de épocas (o iteraciones) que se utilizarán para entrenar el modelo. En este caso, el modelo se entrenará durante 225 épocas.

La función model.fit() devuelve un objeto history que contiene información sobre el rendimiento del modelo durante el entrenamiento, como la pérdida (loss) y la precisión (accuracy) del modelo en cada época. Esta información se puede utilizar para evaluar la calidad del modelo y realizar ajustes en el proceso de entrenamiento si es necesario.

Esta configuración se usó para entrenar el modelo de red neuronal utilizando los datos de entrenamiento cat_data_dep y train_labels. Durante el entrenamiento, el 20% de los datos se utilizarán para la validación y el modelo se entrenó durante 225 épocas. El objeto history devuelto por la función model.fit() contenía la información sobre el rendimiento del modelo durante el entrenamiento.

```
history=model.fit(cat_data_dep,
                  train_labels, validation_split=0.2,
                  epochs=225)
```

Fig. 6 Configuración del entrenamiento

Se usó la función model.evaluate() en TensorFlow para evaluar el rendimiento del modelo entrenado anteriormente ver Fig. 7, se usó los siguientes argumentos:

cat_data_dep: El primer argumento es un conjunto de datos de prueba que se usó para evaluar el modelo. En este caso, cat_data_dep representa los datos de entrada (variables predictoras) del modelo.

train_labels: El segundo argumento representa las etiquetas de prueba (o variables de respuesta) correspondientes a los datos de entrada en cat_data_dep.

verbose=2: Esta opción controla el nivel de verbosidad de la salida del modelo. En este caso, se establece en 2 para que la función model.evaluate() muestre información detallada sobre la evaluación del modelo, como la pérdida y la precisión del modelo.

La función model.evaluate() devuelve dos valores: el valor de pérdida (test_loss) y la precisión (test_acc) del modelo en el conjunto de datos de prueba.

Se usó esta configuración para evaluar el rendimiento del modelo de red neuronal entrenado anteriormente en el conjunto de datos de prueba cat_data_dep y train_labels. La función model.evaluate() devuelve el valor de pérdida (test_loss) y la precisión (test_acc) del modelo en el conjunto de datos de prueba. El argumento verbose=2 se utiliza para obtener una salida detallada de la evaluación del modelo.

```
test_loss, test_acc = model.evaluate(cat_data_dep,
                                     train_labels, verbose=2)
```

Fig. 7 Configuración de la evaluación

Los resultados de la evaluación del modelo de aprendizaje automático, ver Fig. 8, Fig. 9 y Fig. 10, que ha sido entrenado en un conjunto de datos utilizando la función de pérdida (loss) y la métrica de precisión (accuracy) como medidas de rendimiento. La explicación detallada de los resultados de la evaluación es el siguiente:

"83/83": Esto se refiere a que el modelo ha sido entrenado en 83 iteraciones o "épocas" en el conjunto de datos.

"- 0s": El tiempo que tomó entrenar el modelo no se ha registrado.

"loss: 1.1759": La función de pérdida se utiliza para medir la diferencia entre la salida predicha del modelo y la salida real del conjunto de datos de entrenamiento. En este caso, la pérdida media del modelo durante el entrenamiento fue de 1.1759. Un valor bajo de pérdida indica que el modelo está aprendiendo correctamente.

"accuracy: 0.6630": La precisión (accuracy) es una medida de qué tan bien el modelo está haciendo las predicciones correctas en el conjunto de datos de entrenamiento. En este caso, la precisión media del modelo durante el entrenamiento fue de 0.6630 o 66.30%. Un valor alto de precisión indica que el modelo está haciendo predicciones precisas.

"180ms/epoch": Esto indica cuánto tiempo tomó cada época de entrenamiento. En este caso, cada época tardó 180 milisegundos.

"2ms/step": Esto indica cuánto tiempo tomó cada paso individual en cada época de entrenamiento. En este caso, cada paso tardó 2 milisegundos.

En la Fig. 9 se observa una tendencia creciente de la predicción, hacia el 100%.

En la Fig. 10 por lo contrario a la precisión se tiene una tendencia hacia el 0.

Con estos resultados podemos concluir que el modelo está realizando predicciones precisas de 66.30% con una pérdida relativamente baja de 1.1759.

```
83/83 - 0s - loss: 1.1759 - accuracy: 0.6630 - 180ms/epoch - 2ms/step
```

Fig. 8 Resultado del entrenamiento de la red neuronal de clasificación

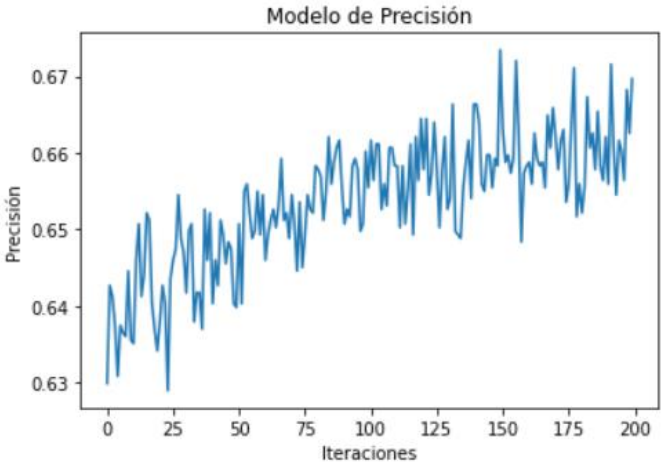

Fig. 9 Comportamiento del valor de la precisión en el entrenamiento

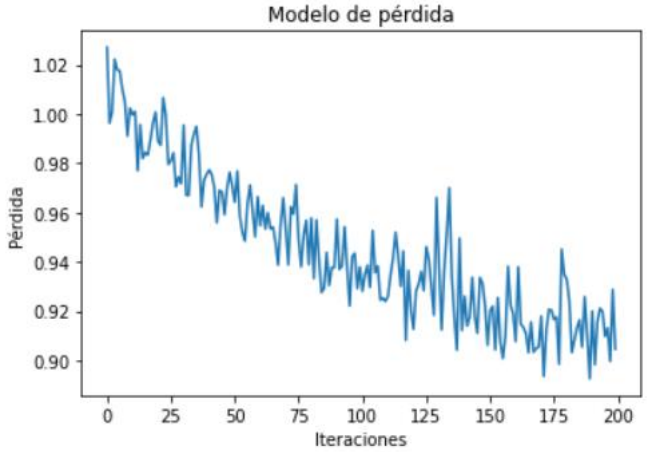

Fig. 10 Comportamiento del valor de la pérdida en el entrenamiento

V. ANÁLISIS DE RESULTADOS

El modelo hallado es una red neuronal de 8 capas internas, además de la capa de entrada y de salidas, se ha considerado 225 iteraciones de entrenamiento, obteniéndose como resultado una precisión de 66.30% aproximadamente ver figura 8; así como el comportamiento de la precisión con una tendencia de crecimiento hacia 1, de acuerdo con la figura 10 y el de la pérdida de 1.1759 ver figura 8, con una tendencia a disminuir ver figura 11. Este modelo permitirá clasificar la situación de un alumno, que permitirá trabajar con los alumnos del tipo de situación considerados de abandono o deserción como el de “Suspensión voluntaria”.

VI. CONCLUSIONES Y RECOMENDACIONES

A. *Conclusiones*

- Se creó un modelo de una red neuronal, que permite clasificar a un alumno, de acuerdo con las clases existentes de la situación de los alumnos.
- El modelo creado tiene una mediana precisión y baja pérdida.
- Este modelo se puede usar para identificar a los alumnos que pueden abandonar los estudios, considerando la situación "Suspensión voluntaria".

B. *Recomendaciones*

- Considerar otros atributos de los alumnos, relacionados a la deserción para poder mejorar la precisión del modelo de red neuronal.
- Usar este estudio como base para futuras investigaciones con respecto a la deserción universitaria.
- Usar el modelo con los alumnos que dejan de estudiar en el futuro, para contrastar la precisión del modelo.

REFERENCIAS

[1] UNESCO. Inteligencia artificial en la educación (s.f.). https://es.unesco.org/themes/tic-educacion/inteligencia-artificial

[2] P. Tocto, G. Huamaní y L. Zuloaga. Construcción de un modelo basado en redes neuronales para determinar la duración de los estudios de ingeniería en una universidad pública en el Perú. 19th LACCEI International Multi-conference for engineering, education and technology,2021.

[3] P. Tocto, G. Huamaní, E Villacorta. Application of artificial intelligence in the management of a public university in Peru: A case of supervised machine learning using neural networks to classify if an engineering student would graduate in 5 years. 20 LACCEI International Multi-conference for engineering, education and technology. 2022.

[4] Orazem y Gunnarson. Child Labor school attendance and performance: A review. Ames: Iowa State University. WP 4001. 2004

[5] L. Rojas. Estudio sobre la repitencia y deserción en la educación superior chilena IESALC – UNESCO April 2005

[6] L. Alcazar. Asistencia y deserción en estudio sobre la oferta y la demanda de educación secundaria en zonas rurales. 2009

[7] OCDE. Estudio de bienestar y políticas de juventud en el Perú. Proyecto OCDE-UE Inclusión juvenil, París. 2017

[8] M. Ardeleanu and Stanescu, (2016) D, "A new research concerning some influence factors in the orientation of high school graduates towards higher technical education," 2016 International Symposium on Fundamentals of Electrical Engineering (ISFEE), pp. 1-5, Doi: 10.1109/ISFEE.2016.7803241.

[9] RRPP. En rpp.pe/campanas/valor-compartido/el-problema-de-la-desercion-universitaria-por-que-los-estudiantes-abandonan-su-carrera-noticia-1371610 27.05.21

[10] F. Alayo. Unos 174000 estudiantes peruanos dejaron la universidad en el 2020. El comercio. Perú 28.09.2020. https://elcomercio.pe/lima/sucesos/unos-174000-estudiantes-peruanos-dejaron-la-universidad-en-lo-que-va-del-2020-noticia/

[11] A. Tovar Y G. Huamán. El otro rostro de beca 18. En OJOpúblico https://ojo-publico.com/2330/beca-18-el-43-de-indigenas-no-logra-concluir-estudios 14.12.20

[12] J. Cotler. Educación superior e inclusión social. Un estudio cualitativo de los becarios del PROGRAMA BECA 18. PRPNABEC 2015.

[13] D. Pasic y D. Kukac, Machine learning model for detecting high school students as candidates for drop-out from a study program. 43rd International Convention on. 1140-1144 Sep 2020; Croatian Society MIPRO 2020

[14] J.D. García. Y A. Skarita. Predicting Academic Performance Based on Students' Family Environment: Evidence for Colombia Using Classification Trees. Psychology, Society & Education 2018

[15] EF. De La Hoz, En. DE La Hoz y T. Fontalvo. Metodología de Aprendizaje Automático para la Clasificación y Predicción de Usuarios en Ambientes Virtuales de Educación. Información Tecnológica. Vol. 30(1), 247-254. 2019

[16] L.Contreras, H.Fuentes y J. Rodríguez. (2020) Predicción del rendimiento académico como indicador de éxito/fracaso de los estudiantes de ingeniería, mediante aprendizaje automático. Formación universitaria. Vol. 13 Issue 5, p 233-246 DOI 10.4067/S0718-50062020000500233

[17] Mk. Sharma Mk y M. Yaday. Predicting Students' Drop-Out Rate Using Machine Learning Models: A Comparative Study. Third International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT) 2022

Revista Iberoamericana sobre Calidad, Eficacia y Cambio en Educación (2013) - Volumen 11, Número 2