UNIVERSIDAD DE HUÁNUCO

ESCUELA DE POSGRADO

PROGRAMA ACADÉMICO DE MAESTRÍA EN INGENIERÍA DE SISTEMAS E INFORMÁTICA, CON MENCIÓN EN GERENCIA DE SISTEMAS Y TECNOLOGÍAS DE INFORMACIÓN

TESIS

"ESTUDIO COMPARATIVO DEL NIVEL DE EFICACIA EN MODELOS ALGORÍTMICOS AL ESTIMAR LA DESERCIÓN DE LOS ESTUDIANTES DEL NIVEL PREGRADO EN LA UNIVERSIDAD DE HUÁNUCO – 2019"

PARA OPTAR EL GRADO ACADÉMICO DE MAESTRO EN INGENIERÍA DE SISTEMAS E INFORMÁTICA, CON MENCIÓN EN GERENCIA DE SISTEMAS Y TECNOLOGÍAS DE INFORMACIÓN

AUTOR: Alvarado Echigoyen, Jhon Frank

ASESOR: Bernardo Tello, Alcides

HUÁNUCO – PERÚ

2022

TIPO DEL TRABAJO DE INVESTIGACIÓN:

- Tesis (X)
- Trabajo de Suficiencia Profesional( )
- Trabajo de Investigación ( )
- Trabajo Académico ( )

LÍNEAS DE INVESTIGACIÓN: Protección del medio ambiente y equilibrio del ecosistema (agua, conflictos ambientales).

AÑO DE LA LÍNEA DE INVESTIGACIÓN: (2018 - 2019)

CAMPO DE CONOCIMIENTO OCDE:

Área: Ingeniería, Tecnología
Sub área: Ingeniería eléctrica, Ingeniería electrónica
Disciplina: Ingeniería de sistemas y comunicaciones

DATOS DEL PROGRAMA:

Nombre del Grado/Título a recibir: Maestro en ingeniería de sistemas e informática, con mención en gerencia de sistemas y tecnologías de información.
Código del Programa: P25
Tipo de Financiamiento:

- Propio (X)
- UDH ( )
- Fondos Concursables ( )

DATOS DEL AUTOR:

Documento Nacional de Identidad (DNI): 70018616

DATOS DEL ASESOR:

Documento Nacional de Identidad (DNI): 22505727
Grado/Título: Doctor en ciencias de la educación
Código ORCID: 0000-0002-0946-0236

DATOS DE LOS JURADOS:

N°	APELLIDOS Y NOMBRES	GRADO	DNI	Código ORCID
1	Jacha Rojas, Johnny Prudencio	Maestro en ingeniería de sistemas e informática con mención en: gerencia de sistemas y tecnologías de información	40895876	0000-0001-7920-1304
2	Camara Llanos, Frank Erick	Maestro en ciencias de la salud con mención en: salud pública y docencia universitaria	44287920	0000-0001-9180-7405
3	Suarez Paucar, Carlos Enrique	Maestro en ciencias con mención en ingeniería de sistemas	41836635	0000-0001-5123-2088

UNIVERSIDAD DE HUÁNUCO

Escuela de Posgrado

ACTA DE SUSTENTACIÓN DEL GRADO DE MAESTRO EN INGENIERÍA DE SISTEMAS E INFORMÁTICA

En la ciudad de Huánuco, siendo las 18:00 horas del día 30 del mes de noviembre del año 2021, en cumplimiento de lo señalado en el Reglamento de Grados y Títulos de la Universidad de Huánuco, se reunieron el sustentante y el Jurado Calificador mediante la plataforma virtual Google meet integrado por los docentes:

- MG. JOHNNY PRUDENCIO JACHA ROJAS
- MG. FRANK ERICK CAMARA LLANOS
- MG. CARLOS ENRIQUE SUÁREZ PAUCAR

Nombrados mediante resolución N° 568-2021-D-EPG-UDH; para evaluar la tesis intitulada **"ESTUDIO COMPARATIVO DEL NIVEL DE EFICACIA EN MODELOS ALGORÍTMICOS AL ESTIMAR LA DESERCIÓN DE LOS ESTUDIANTES DEL NIVEL PREGRADO EN LA UNIVERSIDAD DE HUÁNUCO - 2019";** Presentado por el Bach. **ALVARADO ECHIGOYEN, Jhon Frank** para optar el grado de maestro en ingeniería de sistemas e informática, con mención en Gerencia de Sistemas y Tecnologías de Información.

Dicho acto de sustentación se desarrolla en dos etapas: exposición y absolución de preguntas procediéndose luego a la evaluación por parte de los miembros de jurado.

Habiéndose absuelto las objeciones que le fueron formuladas por los miembros del jurado y de conformidad con las respectivas disposiciones reglamentarias procedieron a deliberar y calificar, declarándolo **Aprobado** por **Unanimidad** con calificativo cuantitativo de **17 y** cualitativo de **Muy Bueno.**

Siendo las **19:41** horas del día martes 30 del mes de noviembre del año dos mil veintiuno, los miembros del jurado calificador firman la presente acta en señal de conformidad.

Presidente
Mg Johnny Prudencio Jacha Rojas

Secretario
Mg. Frank Erick Camara Llanos

Vocal
Mg. Carlos Enrique Suárez Paucar

DEDICATORIA

A Dios

Por haberme permitido llegar a este punto.

A mis padres Atilio y Bertha

Por ser apoyo incondicional en toda mi vida.

A mi novia Leydy Anabel

Por ser parte importante en mi vida y ser la motivación para conseguir mis metas.

A mi hermano Rick Hans

Por acompañarme en todo este proceso de crecimiento.

AGRADECIMIENTOS

Al Dr. Alcides Bernardo Tello por su asesoría, tiempo y dedicación durante la elaboración de esta tesis.

A la Universidad de Huánuco por ser mi casa de estudios y facilitarme la adquisición de conocimientos en mi formación académica.

A mis compañeros de trabajo y amigos que directa o indirectamente contribuyeron a poder lograr este objetivo.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente estudio de investigación tuvo como objetivo comparar el nivel de eficacia en modelos algorítmicos al estimar la deserción de los estudiantes del nivel pregrado en la Universidad de Huánuco. Se definió como un tipo de investigación aplicada, con un enfoque cuantitativo, con alcance o nivel descriptivo y con un diseño pre experimental. Se recopiló un total de 127 332 casos de estudio, donde cada caso de estudio era un conjunto de datos de cada alumno matriculado durante los semestres del 2010-0 al 2018-2 compuesto por 17 atributos y uno de ellos era el indicador de deserción; se seleccionó como muestra la cantidad de 14 800 casos y cuya composición necesaria es que haya igual número de casos de deserción como de no deserción. Dentro del desarrollo se aplicaron técnicas propias de data science, data mining y machine learning; los modelos algorítmicos que se compararon fueron: K-nearest neighbors, Support vector machines, Multi-layer perceptron y Random forest; con ayuda de software desarrollado por el investigador, en el lenguaje Python, se automatizaron ciertas tareas para lograr obtener las métricas de desempeño, de las cuales se eligió como medida de estudio a la precisión. Para la etapa de entrenamiento se utilizó un dataset que fue tomado de la población total con un número similar al número de casos de la muestra para que el aprendizaje de los modelos algorítmicos sea consistente. En la etapa de evaluación de procedió a procesar los casos correspondientes a la muestra donde se obtuvo que la precisión de los modelos rondaba el 75%. Al aplicar la prueba estadística se llegó a comprobar que el nivel de eficacia en modelos algorítmicos presenta diferencias al estimar la deserción de los estudiantes del nivel pregrado en la Universidad de Huánuco, por lo cual se acepta la hipótesis planteada. Considerando el nivel de eficacia basado en la precisión se concluye que el mejor modelo es Random forest, y el peor modelo es K-nearest neighbors.

Palabras clave: Deserción universitaria, Machine learning, Data science, Modelos algorítmicos, Python.

ABSTRACT

The objective of this research study was to compare the level of effectiveness in algorithmic models when estimating the dropout rate of undergraduate students at the University of Huánuco. It was defined as a type of applied research, with a quantitative approach, with a descriptive scope or level and with a pre-experimental design. A total of 127,332 case studies were collected, where each case study was a data set of each student enrolled during the semesters from 2010-0 to 2018-2 composed of 17 attributes and one of them was the dropout indicator; the number of 14,800 cases was selected as a sample, the necessary composition of which is that there be the same number of cases of desertion as non-desertion. Within the development, own techniques of data science, data mining and machine learning were applied; the algorithmic models that were compared were: K-nearest neighbors, Support vector machines, Multi-layer perceptron and Random forest; with the help of software developed by the researcher, in the Python language, certain tasks were automated to obtain performance metrics, from which precision was chosen as the study measure. For the training stage, a dataset was used that was taken from the total population with a number similar to the number of cases in the sample so that the learning of the algorithmic models is consistent. In the evaluation stage, the cases corresponding to the sample were processed, where it was obtained that the precision of the models was around 75%. When applying the statistical test, it was found that the level of efficiency in algorithmic models presents differences when estimating the dropout rate of undergraduate students at the University of Huánuco, so the hypothesis is accepted. Considering the level of efficiency based on precision, it is concluded that the best model is Random forest, and the worst model is K-nearest neighbors.

Keywords: University dropout, Machine learning, Data science, Algorithmic models, Python.

INTRODUCCIÓN

Los modelos algorítmicos se han convertido en una herramienta para el procesado de datos muy importante en los últimos años y su participación en el mundo real es aún más notoria. Cuando cierto problema tiene la característica de tener mucha información disponible y cuyo objetivo es predecir, clasificar o agrupar dicha información es muy probable que la aplicación de los modelos algorítmicos dé la solución correcta. El problema de la deserción es un fenómeno presente en todas las instituciones de educación superior, y en el caso particular de la Universidad de Huánuco es un tema de actual preocupación, actualmente existe mucha información de los estudiantes relacionada a la deserción y se connota como un problema de clasificación; los modelos algorítmicos se presentan entonces como una solución viable.

La multitud de modelos algorítmicos representan las diversas soluciones que existen al problema de la deserción, se hace necesario poder comparar el nivel eficacia de dichos modelos para elegir a los mejores. De esta manera surge la siguiente pregunta: ¿Es diferente el nivel de eficacia en modelos algorítmicos al estimar la deserción de los estudiantes del nivel pregrado en la Universidad de Huánuco?, el cual es el inicio del presente trabajo de investigación.

Cabe recalcar que se hace necesario el manejo de ciertos conceptos y técnicas relacionadas con áreas tales como el data mining, machine learning y data science; estos proveen de herramientas muy útiles para estructurar los datos recopilados, algoritmos de aprendizaje automático y las tareas necesarias para solucionar problemas de clasificación como el estudiado en el presente trabajo de investigación. Dentro de los datos recopilados la deserción solo tiene dos valores posibles (Es deserción y no es deserción), esto conlleva a denominar más específicamente al problema estudiado como de clasificación binaria.

CAPÍTULO I

1. PLANTEAMIENTO DEL PROBLEMA

1.1. DESCRIPCIÓN DEL PROBLEMA

La cantidad de información que actualmente se genera en los sistemas digitales es enorme; los usuarios que antes eran solo consumidores de información se han convertido en generadores de información muy valiosa para diversos fines. Conforme avanza la tecnología, se ha encontrado la manera de cómo utilizar dicha información para la solución de diversos problemas del mundo real; la respuesta más óptima a esto es la aplicación de la inteligencia artificial como pilar fundamental del aprendizaje automático, rediciendo costes en cuanto a labor humana se refiere. La presencia de modelos algorítmicos en la vida cotidiana se ha hecho muy notoria en los últimos años, aunque en forma de servicios (recomendaciones de productos, predicción de valores, clasificación de correo no deseado, asistentes personales autónomos, etc.); el dilema llega cuando existe la necesidad de averiguar qué modelo algoritmo brinda la mejor solución o nivel de eficacia. Alrededor del mundo existen muchos artículos y estudios que buscan encontrar modelos algorítmicos que satisfagan cierto objetivo planteado, llevándonos hacia un escenario donde se hace necesario comparar el resultado de cada uno obteniendo así la mejor solución para el problema planteado.

Cada problema tiene una naturaleza propia, ya sea por el entorno o el periodo de tiempo donde se manifiesta; en el ámbito educativo es aún más determinante, debido a que la educación varia por el idioma, plan de estudio, nivel de estudio, región geográfica, etc. En este sentido la aplicación de modelos algorítmicos a los problemas educativos tiene la necesidad de ser contextualizadas dentro del entorno en donde se presentan; por lo tanto, no es viable aplicar el mismo modelo algorítmico en dos realidades diferentes. Se

hace necesario que dentro del ámbito de la Universidad de Huánuco se realicen estudios de este tipo, para que así se genere un precedente dentro del contexto local y regional acerca del uso de modelos algorítmicos como solución de problemas de nuestra realidad.

La labor fundamental de toda institución dedicada al rubro educativo es formar a sus estudiantes con conocimientos y valores relevantes, a fin de que cuando se cumpla el periodo de estudio determinado pueda ser completada su formación académica. Esto es el funcionamiento correcto y esperado del proceso educativo; pero existe un problema frente a esto, la deserción. La Universidad de Huánuco, como una de las instituciones más representativas de la ciudad de Huánuco, tiene a su cargo una gran cantidad de estudiantes; esto conlleva que dentro del avance académico estudiantil se presenten diversas situaciones, entre estas se observa la presencia de casos de deserción. En la Tabla 12 se puede observar un índice considerable de la deserción dentro de esta universidad, a esto se suma una desatención a las causas y mecanismos de control para este fenómeno. La deserción como tal, representa un problema muy grave en cuanto a la misión de esta institución “Formamos profesionales de alta calidad académica humanística, científica y tecnológica, con sensibilidad para contribuir al desarrollo de la región y el país (…)” (Universidad de Huánuco, 2018, pág. 14); en tal sentido se da a entender que cuando un estudiante interrumpe sus estudios, ya sea por cualquier motivo, no solo está declinando a su objetivo universitario sino también la universidad está fallando en cuanto a su misión como institución educativa. Dado el impacto de la deserción en el proceso formativo de los estudiantes se hace necesario un análisis, con ayuda de técnicas modernas, que nos permitan el manejo de indicadores o estimaciones que faciliten el monitoreo y control de los estudiantes matriculados en el presente periodo y a futuro.

Una de las consideraciones de la presente investigación es analizar no solo la situación actual sino todos los datos históricos que se posea, planteando esto se ve la inutilidad de cualquier tipo de instrumento de recolección de datos que requiera el contacto directo con los estudiantes, dado que solo se obtendría una imagen de la situación actual; en este sentido se dispone del acceso a la base de datos central donde se almacena todos los datos actuales e históricos. A la fecha, la información correspondiente sobre el fenómeno de la deserción y los análisis pertinentes es casi nula, salvo por algunos reportes emitidos; siendo así que la información presente en la universidad es mayormente correspondiente al ámbito académico (cursos, alumnos, docentes, matriculas, pagos, notas, etc.) por lo que se requiere un análisis y tratamiento especial, para que a partir de los datos actuales se construya y elabore información valiosa para el estudio de la deserción. El procesamiento de los datos actuales no solo se centra en la obtención de estos, sino a una correcta adaptación ya sea de formatos, categorías o incluso asumir datos faltantes con la finalidad de obtener una colección de datos consistente para el propósito definido.

La Universidad de Huánuco es un ejemplo muy claro del crecimiento exponencial de la información, considerando que a la fecha se cuenta con 15 783 estudiantes matriculados y haciendo una retrospección a periodos anteriores se puede prever la presencia de grandes volúmenes de datos. Con esta premisa, y entendiendo el ámbito tecnológico e informático de esta investigación nos lleva a abocarnos en el campo del data science, el cual hace uso de la data mining y machine learning para el análisis de grandes volúmenes de datos. Aunque existe mucha información respecto a la aplicación de modelos algorítmicos, se requiere realizar una adaptación del método provisto por estas áreas ajustando ciertos parámetros que permitan un nivel de eficacia aceptable para poder predecir nuevos

casos de deserción dentro de esta universidad y la selección del modelo más óptimo.

1.2. FORMULACIÓN DEL PROBLEMA

1.2.1 Problema general

¿Es diferente el nivel de eficacia en modelos algorítmicos al estimar la deserción de los estudiantes del nivel pregrado en la Universidad de Huánuco?

1.2.2 Problemas específicos

Problema Especifico 1 (PE1)

¿Cómo transformar y estructurar la información actual en un conjunto de instancias para su posterior tratamiento?

Problema Especifico 2 (PE2)

¿Cómo adaptar y entrenar cuatro algoritmos de clasificación?

Problema Especifico 3 (PE3)

¿Cuál es la eficacia de cuatro modelos algorítmicos al estimar la deserción de los estudiantes del nivel pregrado en la Universidad de Huánuco?

1.3. OBJETIVO GENERAL

Comparar el nivel de eficacia en modelos algorítmicos al estimar la deserción de los estudiantes del nivel pregrado en la Universidad de Huánuco.

1.4. OBJETIVOS ESPECÍFICOS

Objetivo Especifico 1 (OE1)

Transformar y estructurar la información actual en un conjunto de instancias para su posterior tratamiento.

Objetivo Específico 2 (OE2)

Realizar la adaptación y entrenamiento de cuatro algoritmos de clasificación.

Objetivo Específico 3 (OE3)

Determinar la eficacia de cuatro modelos algorítmicos al estimar la deserción de los estudiantes del nivel pregrado en la Universidad de Huánuco.

1.5. TRASCENDENCIA Y JUSTIFICACIÓN DE LA INVESTIGACIÓN

Las investigaciones con datos académicos son abundantes; sin embargo, las técnicas y los métodos de análisis modernos abren las puertas hacia procesos más automatizados y mejores resultados. Campos como el data science, data mining y machine learning han tenido un notorio auge dentro del campo del análisis de datos, en ese sentido es menester que siendo esta una investigación con un sentido tecnológico e informático se haga uso de estos para dar una nueva perspectiva del análisis de información referente a la deserción. En la actualidad los estudios en estos campos son muy reducidos en cuanto a artículos o investigaciones en español, lo cual representa la oportunidad de establecer una guía y modelo de trabajo para futuras investigaciones en multitud de rubros y problemas diversos.

Se hace claro que la finalidad de obtener nueva información y las conclusiones serán de mucha ayuda para la institución donde se aplica esta investigación. Una mejor comprensión de la deserción puede ayudar a mejorar los servicios que la universidad brinda; y acompañado del uso de tecnologías modernas, como los modelos algorítmicos, permitirá abrir el camino para nuevas investigaciones.

CAPÍTULO II

2. MARCO TEÓRICO

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

2.1.1. Antecedentes a nivel internacional:

En México, Hernández *et al.* (2016) realizaron un estudio titulado: **"Comparative Study of Algorithms to Predict the Desertion in the Students at the ITSM-Mexico"**. El objetivo fue comparar la evaluación de cuatro algoritmos para predecir la deserción en un instituto superior, haciendo uso de las herramientas Microsoft SQL Server Analysis Services. La muestra estaba compuesta por 134 estudiantes del programa educativo Ingeniería en Tecnologías de la Información y Comunicaciones del Instituto Tecnológico Superior de Misantla – México. Los resultados en términos de precisión fueron: regresión logística (100%), árboles de decisión (0%), clustering (100%) y redes neuronales (100%). En esta investigación se llegó a las siguientes conclusiones:

- Es viable predecir que alumnos tienen altas posibilidades de desertar sus estudios de nivel superior.
- La tarea de identificar alumnos desertores no es tan fácil de realizarlo de manera manual por la cantidad de información que involucra, es necesario un sistema automatizado.
- El algoritmo de regresión logística es más óptimo para realizar la predicción y aún más para definir el perfil de un alumno con alto riesgo de deserción

En Chile, Vásquez (2016) realizó un estudio titulado: **"Modelo predictivo para estimar la deserción de estudiantes en una institución de educación superior"**. El objetivo fue la implementación de modelos predictivos para una detección

temprana de las deserciones en el programa de Ingeniería en Información y Control de Gestión de la Universidad de Chile, se realizó con datos semestrales y los modelos también se aplicaron semestralmente. La muestra lo conforman un total de 928 estudiantes (cada uno con 45 atributos), los cuales son ingresantes a la carrera de Ingeniería de la Información y Control de Gestión de la Universidad de Chile bajo la modalidad de PSU (Prueba de Selección Universitaria) entre los años 2007 al 2014. Los resultados en términos de precisión fueron: redes neuronales (95,43%) y máquinas de soporte vectorial (90,96%). En esta investigación se llegó a las siguientes conclusiones:

- El modelo más óptimo fue el algoritmo de redes neuronales.
- Los predictores que logra identificar corresponden al potencial académico (rendimiento preuniversitario), antecedentes familiares y contexto educacional los cuales impactan en la decisión del estudiante acerca de la continuidad de sus estudios.

2.1.2. Antecedentes a nivel nacional:

En Lima – Perú, Galvez & Flores (2015) realizaron un estudio titulado: **"Modelo predictivo de deserción universitaria de la carrera de Ingeniería Informática en la Universidad Ricardo Palma"**. El objetivo fue la implementación de un modelo para predecir la deserción universitaria, se utilizó la metodología CRIPS-DM (Para el tratamiento de la información) y RUP para el desarrollo de un aplicativo de software que automatiza la carga de información desde una base de datos. La muestra estaba integrada por estudiantes, de los últimos 4 años, de la carrera de Ingeniería Informática de la Universidad Ricardo Palma. Los resultados en términos de precisión fueron: árboles de decisión (84%), red neuronal (72%), clustering (59%), bayes naives (69%), reglas

asociativas (60%) y regresión (70%). En esta investigación se llegó a las siguientes conclusiones:

- Los factores con mayor influencia dentro de la deserción universitaria son el nivel socioeconómico C y aquellos que han desaprobado un aproximado de 4 cursos o más.
- El modelo más óptimo fue el algoritmo de árboles de decisión.

En Lima – Perú, Daza (2016) realizó un estudio titulado: **"Un modelo basado en árboles de decisión para predecir la deserción estudiantil en la Educación Superior Privada"**. El objetivo fue dar solución a la problemática de la deserción en la educación superior mediante modelos basados en árboles de decisión que permitan predecir con alta precisión la deserción de los alumnos; la recopilación de información se realizó mediante la minería de datos usando la metodología CRIPS-DM y la herramienta SPSS Clementine 12.0. La muestra consiste en 1761 datos de los estudiantes de la escuela profesional de Ingeniería de Sistemas de la Universidad Privada César Vallejo, con 27 atributos cada uno, desde el semestre 2009-I al 2013-II. Los resultados en términos de precisión fueron: C5.0 (89%), CRT (84%) y CHAID (90%). En esta investigación se llegó a las siguientes conclusiones:

- El algoritmo C5.0 de árboles de decisión dan buenos resultados para predecir la deserción de un alumno.
- Se comprobó que las variables que más aportan a este tipo de investigación son: código de universidad, rendimiento académico de la universidad, rendimiento académico del colegio, edad, sexo y la deserción. Y las variables que influyen de manera importante son: currícula, ciclo, cantidad de créditos aprobados y cantidad de créditos desaprobados.

2.1.3. Antecedentes a nivel local

No se encontraron trabajos similares.

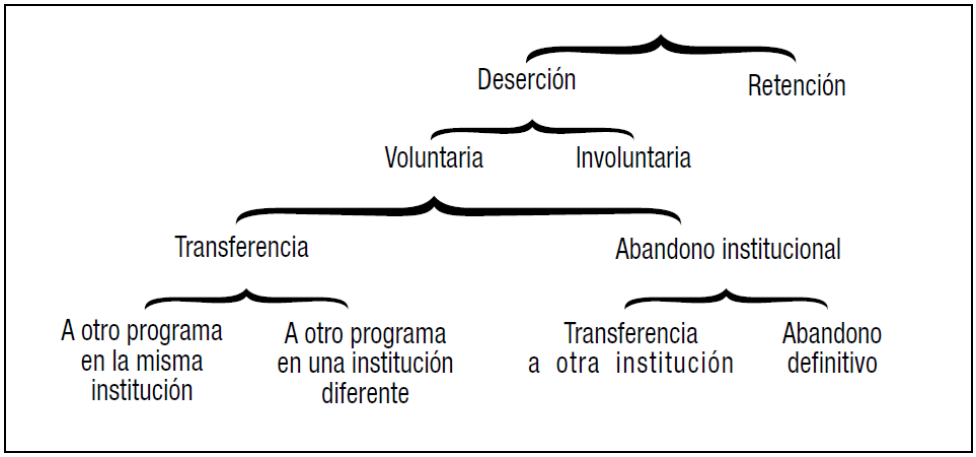
2.2. BASES TEÓRICAS

2.2.1 Deserción universitaria

El termino deserción se puede conceptualizar como un abandono o interrupción de alguna actividad, tarea o función asignada; esta no solo se presenta en el ambiente educativo, sino que forma parte del día a día de cualquier proceso. La deserción como tal, es visto como un problema y generalmente se busca minimizar estos casos ya sea creando mecanismos de control o indicadores que ayuden a identificar cuando un determinado caso esta propenso a presentar deserción.

En el ámbito de la deserción universitaria y de acuerdo con Himmel (2002): “La deserción se refiere al abandono prematuro de un programa de estudios antes de alcanzar el título o grado (…)” (pág. 94). Entendiendo que esto ocurre dentro de un ambiente de educación universitaria y siendo está una de las autoras más referidas en el ambito de los estudios sobre la deserción, plantea una tipología que reflejan una realidad muy común en muchas instituciones educativas.

Figura 1. Tipos de deserción en la educación superior



Fuente: Himmel, E. (2018). Modelo de análisis de la deserción estudiantil en la educación superior.

En un claro análisis de la Figura 1, podemos apreciar la complejidad y ciertos aspectos asociados a la deserción; los diferentes puntos de vista o áreas de interés que puedan aplicarse al estudio de este fenómeno brindarán como resultado un entendimiento más claro de lo que conlleva a un estudiante a la deserción universitaria.

Retención universitaria

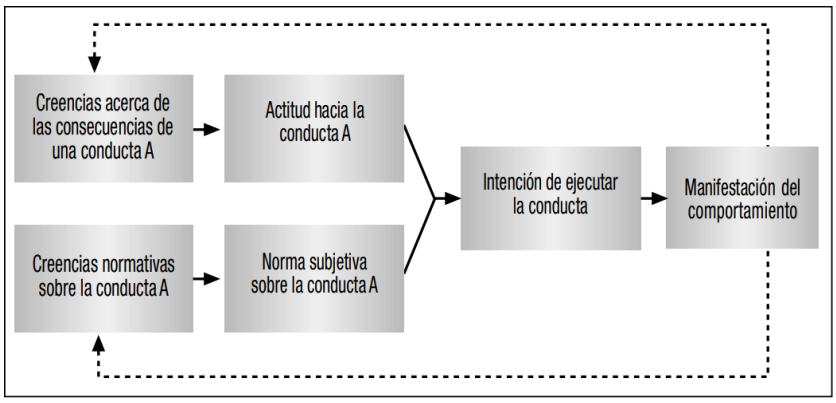
Si bien es cierto que la deserción universitaria es uno de los problemas que más atañen a la formación profesional, existe también el concepto de retención universitaria que permite entender los factores o características que influyen en un estudiante para continuar sus estudios superiores. En ese sentido la deserción se muestra como un concepto contrario a la retención universitaria y el estudio de estos dos fenómenos sirve para poder entender cuál es el punto de quiebre donde la retención se convierte en deserción, entendiendo que el éxito de uno será el fracaso del otro.

A lo largo de la historia se han planteado diferentes enfoques o modelos que tratan de explicar la retención universitaria y aunque dichos modelos han ido mejorando conforme el avance de los años, cabe recalcar los siguientes:

a) En la Figura 2 se muestra unos de los primeros modelos planteados, los autores formulan que las creencias de una persona tienen influencia sobre sus actitudes e intenciones, en consecuencia, esto se ve reflejado en el comportamiento de la misma. Esto no es un proceso que se realice una sola vez, sino que conforme el estudiante va adquiriendo nuevas experiencias dichas creencias se ven reforzadas o debilitadas, sumado a esto las creencias normativas (valores considerados correctos dentro de un ámbito definido), van creando dentro del individuo las intenciones de continuar su formación profesional o

interrumpirlo. Entonces cuando nos referimos a retención, se trata de un caso de reforzamiento de la intención conductual; mientras que la deserción viene a ser un caso de debilitamiento.

Figura 2. Modelo de Fishbein y Ajzen (1975)

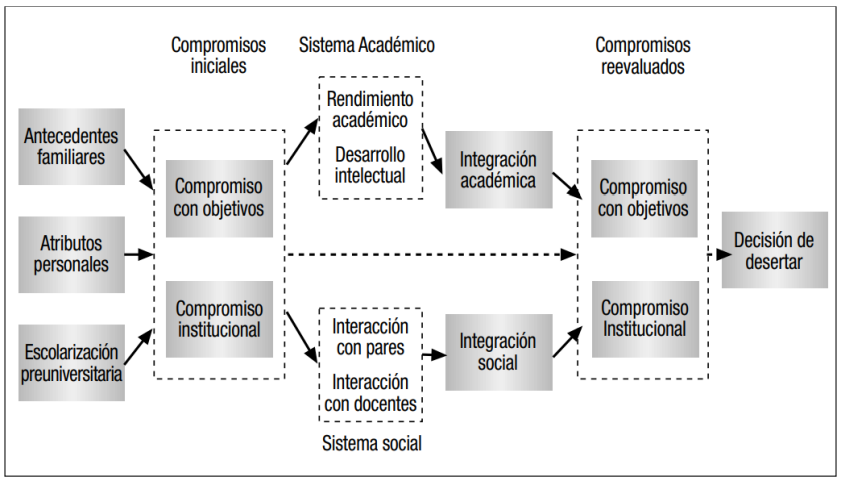


Fuente: Himmel, E. (2018). Modelo de análisis de la deserción estudiantil en la educación superior.

b) Otro de los modelos que tratan de explicar la retención y deserción se muestra en la Figura 3, donde al autor enriquece su punto de vista con un concepto conocido como la teoría del intercambio. Se plantea que un individuo trata de evitar aquellas conductas que requieran de algún tipo de costo y procuran encontrar cierto beneficio en las relaciones, interacciones y estados emocionales. El autor propone que el estudiante va construyendo su integración social y académica siguiendo este principio; si un individuo percibe que los beneficios por continuar su formación profesional son mayores que los costos personales a los cuales está sometido decidirá continuar, en caso contrario si otras actividades son vistas como mayor recompensa que los estudios universitarios se tendrá más predilección para desertar. Aquí no se evalúan solo aspectos académicos sino también la experiencia social, siendo así que la retención universitaria se puede entender como un compromiso estudiantil

consolidado tanto en objetivos personales y la integración social.

Figura 3. Modelo de Tinto (1975, 1982)



Fuente: Himmel, E. (2018). Modelo de análisis de la deserción estudiantil en la educación superior.

La deserción como problema educativo y social

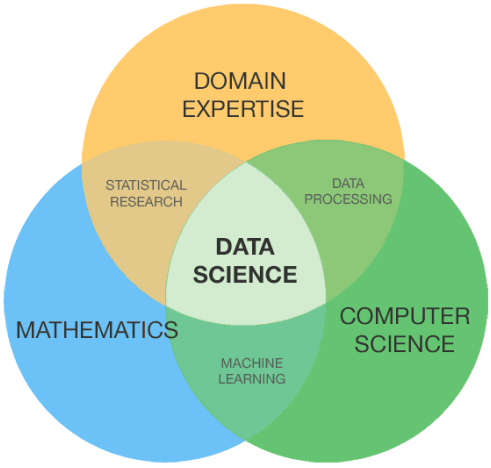
Muchas veces dentro del entendimiento común de las personas, la deserción se ubica solo como un problema educativo pero un entendimiento más amplio de este fenómeno nos llevará a entender las consecuencias sociales que implica. A opinión de Páramo & Correa (1999): “El primer objetivo de los sistemas educativos, debe ser disminuir la vulnerabilidad social de los niños y jóvenes procedentes de medios marginados y desfavorecidos, a fin de romper el círculo vicioso de la pobreza y la exclusión.” (pág. 70). Queda claro no solo la labor educativa de las instituciones de este rubro, sino la responsabilidad social que conlleva el éxito de la formación académica de sus estudiantes.

2.2.2. Data science

De acuerdo a Stanton (2013), data science se refiere a un área emergente de trabajo concerniente con la recopilación, preparación, análisis, visualización, gestión y conservación de grandes colecciones de información. Como su mismo nombre lo

indica es la ciencia aplicada a los datos, esto es un concepto nuevo que abre camino a nuevos estudios y técnicas.

Figura 4. Data science

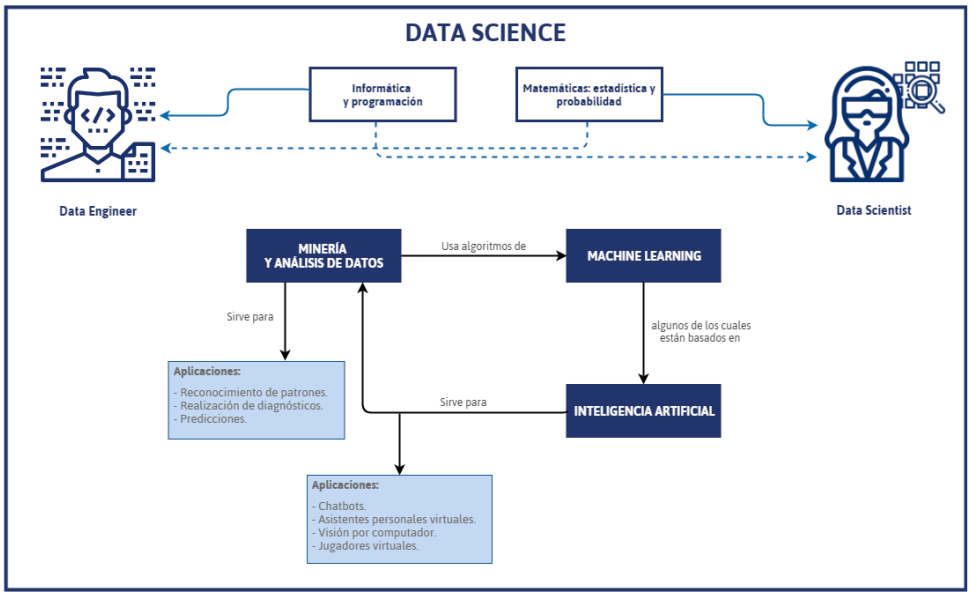


Fuente: Palmer, S. (2015). Data Science for the C-Suite. Recuperado de: https://www.shellypalmer.com/data-science/

Con base en la Figura 4 se puede entender a esta nueva área como la intersección de otras áreas, podemos concluir a partir de aquí que justamente es por eso que lleva el título de emergente. Si bien es cierto que ya desde años anteriores se mencionaba dicho termino, es ahora que ha cobrado notoriedad con el auge de la tecnología y la inteligencia artificial como parte del día a día de la población mundial en todos sus niveles. Su nacimiento fue como resultado de la necesidad de interrelacionar las ciencias computacionales, matemáticas y experiencia de campo; estos a su vez nutren de multitud de herramientas para su desarrollo y avance. Como se observa en el mundo actual la tecnología está logrando avances muy notorios y relacionados al día a día de las personas, las cuales generan información y avances muy valiosos que puede ser usada para distintos fines y más cuando sean aplicados a la investigación.

Esta área se relaciona con otras disciplinas detalladas en la Figura 5, donde también se aprecia la diferencia entre data enginner y data scientist.

Figura 5. Data science y otras disciplinas



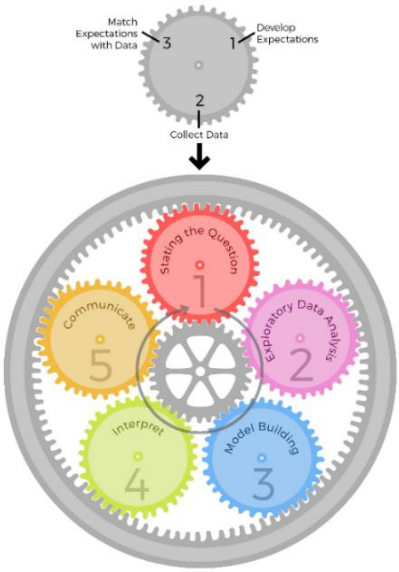
Fuente: Lafuente, A. (2018). Data science: qué es (y qué no es). Recuperado de: https://aukera.es/blog/data-science-que-es-y-que-no-es/

Podemos concluir que el data science es la aplicación de conocimientos de informática y programación, apoyados en técnicas matemáticas de estadística y probabilidad para el análisis de datos y su posterior presentación al público para que sean útiles y fácil de interpretar. Su aplicación se puede dar en multitud de problemas de diferentes áreas o disciplinas y se caracteriza por trabajar con datos no estructurados; si bien es cierto que los distintos aplicativos de software tratan de que la información obtenida este almacenada en estructuras de fácil consulta y recuperación, la realidad es que al momento de acudir a los centros de datos es muy frecuente encontrar datos no estructurados o distribuidos, es ahí donde entra a tallar el data science.

Según Peng & Matsui (2018), define el análisis de datos como un proceso altamente iterativo y no lineal, esto debido que la información de una etapa sirve de retroalimentación para otra

estableciéndose una serie de ciclos de flujo de información muy fuertemente ligado a todo el proceso del análisis de datos. Como el data science tiene como fin el analizar aquellos datos de interés, se plantea esta percepción en la Figura 6.

Figura 6. Epiciclos y etapas del análisis



Fuente: Peng, R., & Matsui, E. (2018). The Art of Data Science.

También definieron una serie de etapas que constituyen las actividades macro dentro del data science:

a) Definición y formulación de preguntas.
b) Exploración de datos.
c) Construcción de modelos estadísticos formales.
d) Interpretación de resultados.
e) Comunicación de resultados.

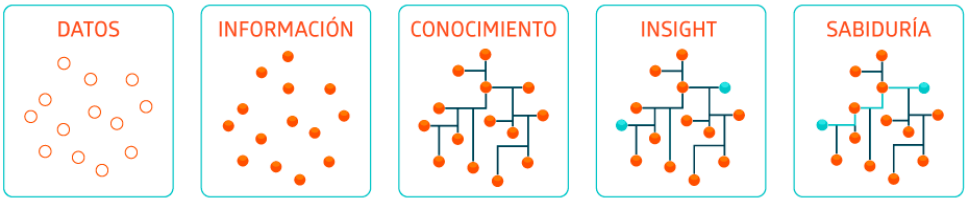
Dentro de la percepción presentada se define el termino epiciclo como una serie de tres pasos que se realizan en cualquier momento de las cinco etapas antes mencionadas y que ayudan a dar ese flujo de información no lineal. Los epiciclos del análisis son:

a) Definición de expectativas.
b) Recopilación de datos.
c) Revisión expectativas y datos.

2.2.3. Data mining

Zaki & Meira (2014) enfatizan que el data mining está comprendido por algoritmos que permiten obtener conocimientos a partir de datos masivos. Se caracteriza por ser un campo interdisciplinario relacionado a sistemas de base de datos, estadística, aprendizaje automático y reconocimiento de patrones; resaltando que los puntos de vista algebraico, geométrico y probabilístico tienen un rol importante durante un proceso de data mining. La importancia matemática en este ámbito radica que el objeto esencial de análisis es un conjunto de datos que representa n puntos en un espacio d-dimensional, el cual se traslada a planos matemáticos para su tratamiento y aplicación de fórmulas.

Figura 7. Creación del conocimiento



Fuente: Lages, G. (2018). ¿Qué es Data Science y cómo utilizar el análisis de datos para ampliar tu negocio?. Recuperado de: https://blog.hotmart.com/es/que-es-data-science/

El data mining tiene como objetivo la construcción de estructuras más complejas a partir de datos recopilados de una determinada área de estudio, dicho proceso permite ir escalando en complejidad como se muestra en la Figura 7; siendo muy usada para los sistemas de apoyo a la toma de decisiones, sistemas basados en indicadores y otros que ayuden a la gestión de recursos dentro de una entidad.

Matriz de datos

Según Zaki & Meira (2014), es la representación o abstracción de la información o datos recopilados en una matriz de dimensiones $n \times d$, con n filas y d columnas. Cada fila corresponde a la medición de ciertos atributos estudiados de una entidad dada; también

llamados instancias, ejemplos, registros, transacciones, objetos, puntos, vector de características entre otros. Las columnas representan a las mediciones o datos de interés; también llamados atributos, propiedades, características, dimensiones, variables, etiquetas. Una matriz de datos se podría visualizar de la siguiente manera:

$$D = \left(\begin{array}{c|cccc} & X_1 & X_2 & \cdots & X_d \\ \hline x_1 & x_{11} & x_{12} & \cdots & x_{1d} \\ x_2 & x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_n & x_{n1} & x_{n2} & \cdots & x_{nd} \end{array}\right)$$

Atributos

Zaki & Meira (2014) mencionan que los atributos están clasificados en:

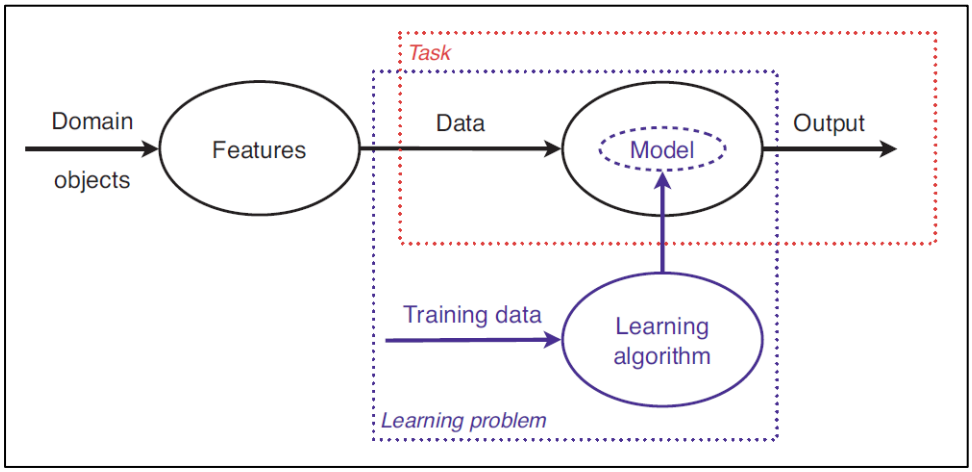
a) Atributos numéricos: Son aquellos que adoptan un valor dentro del conjunto de los números reales. Por ejemplo: Edad (1; 5; 6), Temperatura (34.50; 67.89; 2.03), etc.
b) Atributos categóricos: Son los que toman su valor dentro de un conjunto definido de etiquetas o categorías. Por ejemplo: Color (azul, verde, rojo), Posición de llegada (primero, segundo, tercero), etc.

2.2.4. Machine learning

Surge como un campo de estudio cuyo objetivo es construir programas informáticos que mejoren automáticamente con la experiencia (Mitchell, 1997). Si bien es cierto que el ser humano es apto para el desarrollo de diversas tareas, ya sean simples o complejas; los avances de la tecnología han logrado que se reemplace a las personas en tareas rutinarias y hasta cierto grado de complejidad, esto se logra identificando el proceso actual y dotando a los programas informáticos de ciertas instrucciones capaces de reproducir el actuar humano para dichas tareas. Conforme la complejidad de las tareas va en aumento, resulta casi

imposible y poco óptimo poder establecer qué instrucciones o comandos puedan ser usados por los programas informáticos para ejecutar la tarea en cuestión; es aquí donde los algoritmos de machine learning cobran protagonismo brindando mayores beneficios para problemas complejos y cuya característica fundamental es que a partir de la experiencia se podrá ofrecer mejores resultados, en otras palabras están programados para aprender.

Figura 8. Visión general de un caso de machine learning



Fuente: Flach, P. (2012). Machine Learning. The art and Science of Algorithms that Make Sense Data.

Según Flach (2012), un caso de machine learning puede estar guiado por el esquema de la Figura 8. Por un lado, están los objetos del dominio o casos de estudio y a partir de estos se obtienen datos que sirven para llevar a cabo una tarea específica; en ese sentido al aplicar machine learning aparecen nuevos elementos tales como las características o atributos que son datos relevantes de los objetos de dominio, parte de ellos serán usados para entrenar un algoritmo de aprendizaje el cual dará lugar a un modelo que al interactuar con la información inicial deberá realizar la tarea en cuestión. Los modelos son medidos para probar su rendimiento, es así como no para todos los casos se aplican el mismo modelo o algoritmo de aprendizaje dado que dependiendo de cuan optimo sea un modelo

u otro se optará llevarlo a funcionar en un ambiente real o de producción.

Métodos

El machine learning adopta ciertos métodos para ser aplicados en las tareas o problemas del mundo real, y estas pueden clasificarse en:

a) Regresión: Es usado para predecir un valor, tiene mayor utilidad cuando el conjunto de datos no tiene mucha dimensionalidad. (Ejemplo: Predecir la cantidad de postulantes del próximo año.)
b) Clasificación: Se centra en asignar categorías a un conjunto de datos, conociendo la totalidad de categorías existentes para el problema. (Ejemplo: Clasificar si el perfil económico de un grupo de clientes es Confiable, Dudoso o Peligroso)
c) Agrupación: Se aplica para encontrar grupos a partir de un conjunto de datos no categorizados, pero que cuentan con información válida para este fin. (Ejemplo: Búsqueda de patrones de comportamiento en los pacientes de una clínica psiquiátrica.)

Tipos

a) Supervisado: Es el caso donde del conjunto de datos están etiquetados o tiene un valor de salida definido. Dentro de este se ubican los métodos de regresión y clasificación.
b) No supervisado: Se produce cuando dentro del conjunto de datos solo se puede identificar las características o atributos sin conocer a que categoría, clase, etiqueta o valor de salida definido. Dentro de este se ubica el método de agrupación.

Etapas

De acuerdo con Contreras (2016), el proceso de machine learning tiene las siguientes etapas:

a) Seleccionar el problema a resolver.
b) Seleccionar características.
c) Seleccionar algoritmos de machine learning.
d) Preparar datos.
e) Analizar datos.
f) Analizar rendimiento de modelos.
g) Publicar el modelo.
h) Monitorear y ajustar.

Algoritmos de aprendizaje

Este es el núcleo del machine learning, ya que justamente le da esa capacidad de "aprender" a las computadoras. Se podría pensar que este término solo es usado para referirse a los algoritmos que imitan el pensar humano o la estructura del cerebro; nada más alejado de la realidad ya que podemos encontrar múltiples técnicas y mucha teoría matemática que ayudan a conseguir este mismo objetivo. Algunas áreas donde se apoya el machine learning para su desarrollo son la estadística y la probabilidad. El planteamiento general para el método de clasificación del tipo supervisado es que dado un conjunto de datos $D = \{d_1, d_2, d_3, \dots, d_m\}$ $(d = \mathbb{R}^n)$ y su correspondiente conjunto de clases $C = \{c_1, c_2, c_3, \dots, c_m\}$ se busca encontrar una función de correspondencia $f: D \to C$ para el tratamiento de nuevos casos.

Como algoritmos, estos han tenido un desarrollo más que amplio a lo largo de toda la historia y cuyas implementaciones en los diversos lenguajes de programación y servicios han hecho que su uso aumente a tal punto que, hoy en día, es casi algo indispensable la interacción que tenemos con ellos.

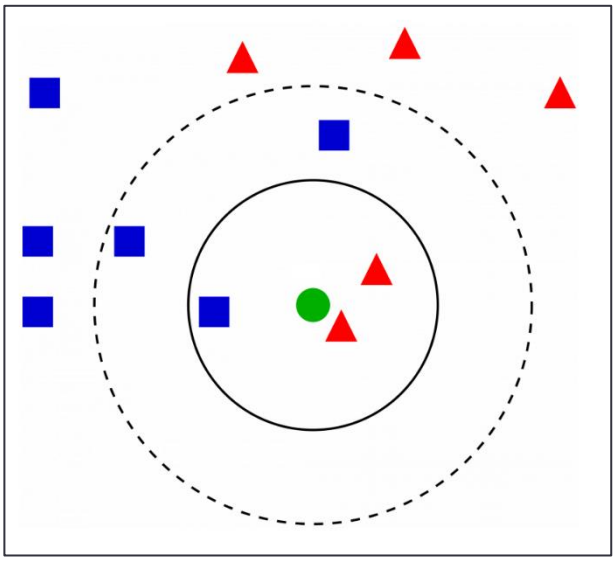
Algunos de los algoritmos usados para la clasificación son:

a) K-nearest neighbors (k-vecinos más cercanos)

Este algoritmo pertenece al método de clasificación del tipo supervisado, se basa en la proximidad entre diversos puntos para la clasificación.

Como primer paso se define el valor de k, el cual representa el total de elementos cercanos a buscar. Sea $\rho(x, x')$ una función para medir la distancia entre dos puntos dentro de un plano n-dimensional, esta función puede usar diferentes implementaciones: Euclidiana, Manhattan, Minkowsky, etc. Se tiene X como el nuevo caso a clasificar y se reordena ascendentemente a todos los datos de acuerdo a la función $\rho(X, x_i)$ para todo $i \leq m$ dentro de un nuevo conjunto $\alpha = \{\alpha_1, \alpha_2, \alpha_3, .., \alpha_m\}$; se agrupa de acuerdo al conjunto de clases a todos los datos que cumplan $\alpha_i \leq \alpha_k$ y se asigna a X la clase de la cual existan mayor cantidad de casos.

Figura 9. Algoritmo k-nearest neighbors



Fuente: Srivastava, T. (2018). Introduction to k-Nearest Neighbors: A powerful Machine Learning Algorithm (with implementation in Python). Recuperado de: https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

Ejemplo: En la Figura 9 el circulo verde es el nuevo caso a clasificar, en este caso $k = 3$ y los elementos más próximos son dos triángulos y un cuadrado, por lo tanto, se asigna la clase de los triángulos. Por otro lado si el caso fuera que $k = 5$ se le asignaría la clase de los cuadrados.

b) Support vector machines (máquinas de soporte vectorial)

Este algoritmo pertenece al método de clasificación del tipo supervisado, se basa en encontrar un hiperplano o un conjunto de estos que permitan separar las diversas clases de interés.

El objetivo es definir el siguiente hiperplano:

$$H(x): (\omega_1 x_1 + \cdots + \omega_n x_n) + b = \omega x + b = 0$$

Este hiperplano tiene la característica de ubicarse en las regiones fronterizas de cada clase, exactamente a la misma distancia de los puntos más próximos, esta separación está determinada por los vectores de soporte asegurándose que este a la máxima distancia posible. Así mismo para un caso de clasificación binaria se tiene los hiperplanos que contienen a los puntos más próximos.

$$H_{+1}: \omega x + b = 1 \qquad\qquad H_{-1}: \omega x + b = -1$$

Sea un punto $P \in H_{+1}$, entonces la distancia entre H_{+1} y H esta puede expresarse por:

$$d(P, H) = \frac{|H_{+1}|}{\|\omega\|} = \frac{|\omega x + b|}{\|\omega\|} = \frac{1}{\|\omega\|} = margen$$

A partir de esto, llegamos a la conclusión que el objetivo es minimizar la función $f(\omega) = \|\omega\|$, esto es un problema de optimización cuadrática el cual se puede reescribir de la siguiente forma:

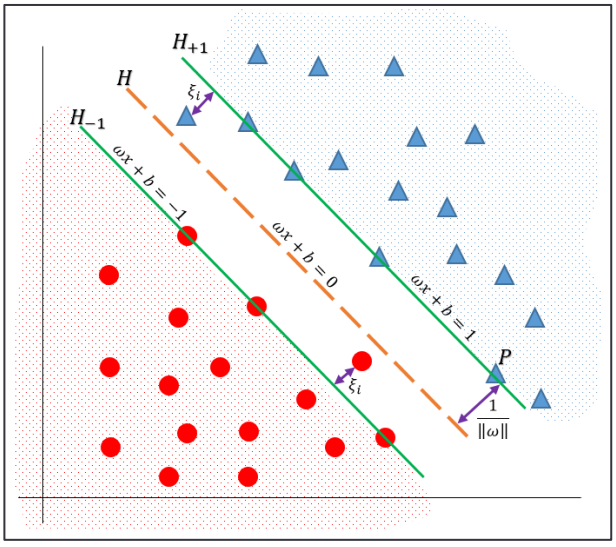
$$\min_{\omega, b, \xi} \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m} \xi_i$$

$$s.a. \quad y_i(\omega x + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

La última función $y_i(\omega x + b) \geq 1 - \xi_i$ no es más que la generalización de los hiperplanos H_{-1} y H_{+1}; ξ_i son los errores de clasificación por el hiperplano; C es un parámetro que modifica la maximización del margen contra el grado de penalización o error, un gran valor expresa menor error y un valor pequeño permite mayor error. El problema de optimización se resuelve con los multiplicadores de Lagrange.

Figura 10. Algoritmo support vector machines



Elaboración propia

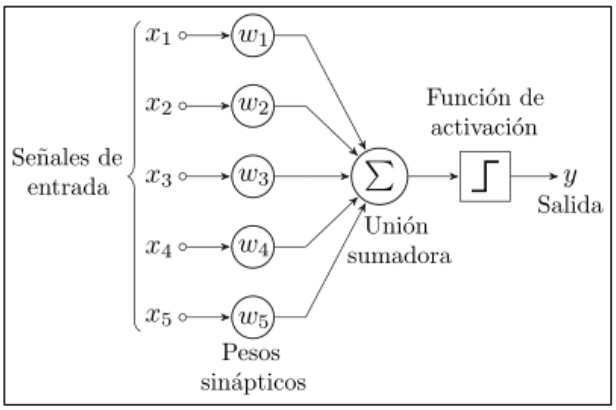
Extendiendo sus aplicaciones, se pueden emplear conjunto de datos con mayor dimensionalidad y cuya separación no sea lineal, para esto se aplica el concepto de kernel que ayuda a transformar la disposición del conjunto de datos de manera que la obtención del hiperplano sea más factible. Algunos de los tipos de kernel más usados son: gaussiano, exponencial, polinomial y sigmoidal.

c) Multi-layer perceptron (perceptrón multicapa)

Este algoritmo pertenece al método de clasificación del tipo supervisado, se basa en el uso de redes neuronales dispuestas por capas.

El perceptrón o neurona artificial es un discriminador lineal que usa una tasa de corrección para las ponderaciones de los datos de entrada y que mediante iteraciones se llega a un resultado óptimo convergente. Puede ser representado con la función $y = \delta(\omega x + b)$, donde δ es la función de activación que normaliza el valor de salida en 0 o 1.

Figura 11. Perceptrón



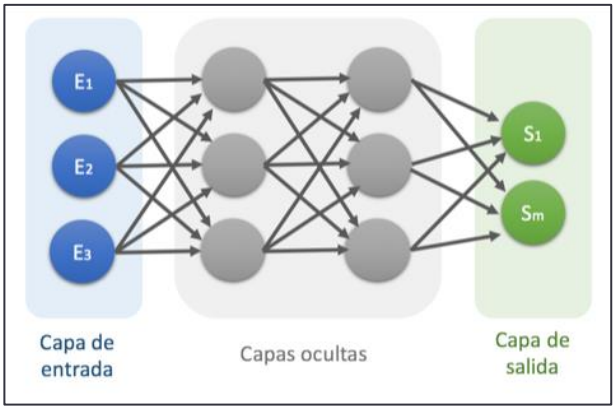
Fuente: Wikipedia (2019). Perceptrón. Recuperado de: https://es.wikipedia.org/wiki/Perceptr%C3%B3n

La actualización de los ponderaciones o pesos se da por $\omega_i' = \omega_i + \alpha(z - y)x_i$, donde α es un valor numérico que expresa la tasa de aprendizaje, i es el número de iteración y z es la salida esperada.

El perceptrón multicapa es un caso especial de redes neuronales que han sido agrupadas de la siguiente manera: la capa de entrada, las capas ocultas y la capa de salida. Su funcionamiento básico se da en tres etapas y refleja el mismo proceso que un perceptrón; se realiza los cálculos de las sumatorias y la salida de las funciones de activación de las capas iniciales que a su vez se convierten en entradas para las capas posteriores; una vez que los valores se hayan propagado hasta la capa de salida, se realiza la comparación con el valor esperado y se obtiene

una tasa de error o perdida; por último, el error obtenido es propagado hacia las capas anteriores corrigiendo el valor de los pesos de cada perceptrón y concluyendo la iteración.

Figura 12. Algoritmo multi-layer perceptron



Fuente: Calvo, D. (2018). Perceptrón Multicapa – Red Neuronal. Recuperado de: http://www.diegocalvo.es/perceptron-multicapa/

d) Random forest (Bosques aleatorios)

Este algoritmo pertenece al método de clasificación del tipo supervisado, se basa en el uso de árboles de decisión en paralelo para obtener una clasificación más aproximada.

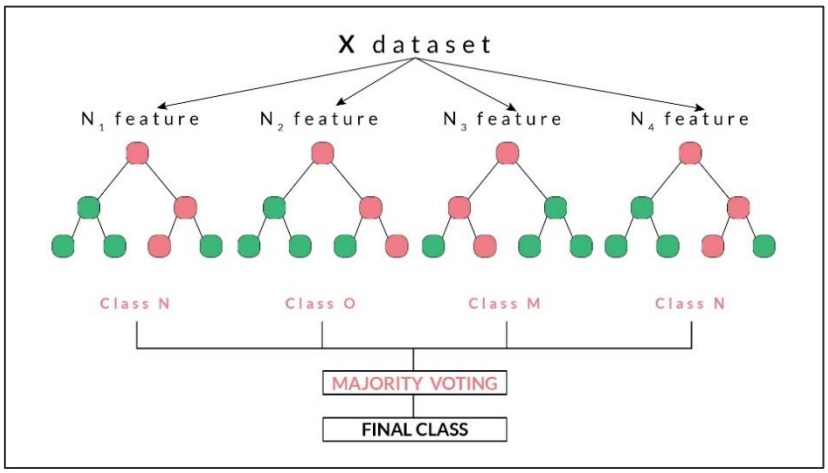
Un árbol de decisión es una estructura de condicionales o reglas compuesto por nodos y ramas que se establece como la guía de un conjunto de datos para poder generalizar la clasificación de estos. Toda la estructura de un árbol de decisión se construye a partir del conjunto de datos, eso puede traer algunas desventajas ya que la calidad de estos determina directamente en la eficacia del algoritmo. Otro aspecto a considerar es que al estar compuesto por condicionales no hay lugar a evaluaciones aproximadas, siendo que las fronteras de evaluación pasan de un estado a otro en un espacio muy corto.

Su implementación está determinada por varias técnicas, pero una de las más conocidas es ID3 (Induction Decision Trees). Para la construcción de la estructura de condicionales o reglas de decisión, se procede a analizar el conjunto de datos al que se tiene acceso de manera que se identifique que atributo será usado como raíz, para ello se calcula la entropía (La medida de la incertidumbre) $H(S) = \sum_{x \in X} -p(x) \log_2 p(x)$, donde X es el conjunto de clases, S es el conjunto de datos actual y $p(x)$ es la proporción de elementos que pertenecen a la clase x sobre el total de elementos del conjunto de datos S. Luego se calcula la ganancia de cada uno de los atributos $IG(S, A) = H(S) - \sum_{t \in T} p(t)H(t) = H(S) - H(S|A)$, donde $H(S)$ es la entropía del conjunto de datos S, t es el subconjunto de S según los valores del atributo A, $p(t)$ es la proporción de elementos del subconjunto de datos t sobre el total de elementos del conjunto de datos S y $H(t)$ es la entropía del subconjunto de datos t. Se escoge como raíz al atributo que tenga mayor ganancia y el conjunto inicial de datos S queda dividido subconjuntos T, al cual se aplica este mismo procedimiento a cada uno de manera que dentro de ese nuevo subconjunto se encuentre el atributo raíz y realizar otra división. Todo esto se realiza recursivamente, teniendo en cuenta las siguientes consideraciones: Si todos los elementos pertenecen a una clase se retorna la misma clase, si no se tienen más atributos se retorna el valor de la clase mayoritaria o en caso contrario, se busca el mejor atributo volviendo a aplicar todo el proceso.

Los bosques aleatorios tienen como característica fundamental el de dividir el número de atributos en subconjuntos de manera aleatoria, y cada subconjunto

servirá para crear un árbol de decisión; al final se hace una sumatoria o ponderación de los valores de salida de cada árbol de decisión y se da como salida el valor de la clase mayoritaria. Esto brinda ciertos beneficios como reducir la profundidad de los árboles de decisión y reducir la dimensionalidad de los datos; pero también la distribución de estos favorece a que, habiendo muchos árboles de decisión, el resultado es aún mejor debido al trabajo en conjunto de todos. Claro está que no es la solución perfecta para todos los casos, pero si gozan de una buena aceptación y con un correcto proceso de entrenamiento se puede llegar a resultados óptimos.

Figura 13. Algoritmo random forest



Fuente: Tahsildar, S. (2019). Random Forest Algorithm In Trading Using Python. Recuperado de: https://blog.quantinsti.com/random-forest-algorithm-in-python/

Cada algoritmo tiene sus puntos positivos y negativos, como profesional es muy importante un correcto conocimiento y manejo de estos para lograr mejores resultados.

Métricas de desempeño

Dentro del usó de los modelos algorítmicos, existe una etapa de evaluación para poder medir el desempeño del mismo. De acuerdo a Japkowicz & Shah (2011), es necesario considerar la distribución de clases para entender la efectividad de las métricas,

siendo así que existen datos donde las clases son desbalanceadas o existen más de dos clases a clasificar. En el caso de los problemas de clasificación binaria se asume que la presencia de la característica en estudio se denomina casos positivos y la ausencia casos negativos, siguiendo esta denominación se plantea los siguientes indicadores:

- (TP) Verdaderos positivos: Son aquellos valores positivos que han sido pronosticados correctamente.
- (TN) Verdaderos negativos: Son aquellos valores negativos que han sido pronosticados correctamente.
- (FP) Falsos positivos: Son aquellos valores positivos que han sido pronosticados incorrectamente.
- (FN) Falsos negativos: Son aquellos valores negativos que han sido pronosticados incorrectamente.

Estos indicadores dan lugar a las siguientes métricas que sirven para determinar qué tan optimo es un modelo determinado o comparar el desempeño de varios modelos:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$f1 - score = 2 * \frac{precision * recall}{precision + recall}$$

$$AUC = área\ bajo\ la\ curva\ ROC$$

2.3. DEFINICIONES CONCEPTUALES

2.3.1. Estimación: Es la atribución de un valor numérico a una persona, cosa o suceso mediante ciertos criterios que facilitan su comprensión; de esta manera se le da una representación que no solo es categórica sino establecer su grado de pertenencia, causalidad, gravedad u otros (Peng & Matsui, 2018).

2.3.2. Clasificación: Se denomina así al proceso de etiquetar por clases a un conjunto de objetos o instancias realizada por alguna técnica o algoritmo. Estás se basan al procesamiento de los atributos del conjunto datos o dataset, los cuales mediante procesos matemáticos logran determinar una relación entre los atributos y la clase, estableciendo así una función de correspondencia (Zaki & Meira, 2014).

2.3.3. Datos: Característica o atributo obtenida a partir de la observación, medición o alguna percepción. Es una forma de interpretar la realidad traduciéndola a medidas o categorías que conocemos y facilitarán su manejo para diferentes propósitos (Leek, 2015).

2.3.4. Información: Es el conjunto de datos relacionados y que tiene un significado concreto; bajo este punto influye mucho la interpretación siendo que los mismos datos pueden brindar diferentes tipos de información (Leek, 2015).

2.3.5. Dataset: Es el conjunto de objetos o instancias que serán usadas para entrenar un algoritmo y realizar su validación, surgen como resultado de la recolección de información de base de datos, mediciones, compilaciones, etc. Requieren un tratamiento debido a que los datos encontrados en el mundo real no son lo suficientemente normalizados para el uso en algoritmos (Witten & Frank, 2005).

2.3.6. Algoritmo: Conjunto de pasos definidos que permiten obtener una salida o realizar algún proceso. Estos pueden estar expresado ya sea en lenguajes de programación, lenguaje natural, pseudocódigo, etc. Algunos de ellos reciben la denominación de modelos debido a tener una estructura definida, haciendo posible su aplicación en un determinado problema modificando ciertos parámetros o adaptándolo a la realidad estudiada (Flach, 2012).

2.3.7. Python: Lenguaje de programación interpretado cuya alta aceptación ha repercutido en el alcance de diferentes áreas. Posee muchas librerías orientadas para el trabajo de machine learning y el análisis de datos, así como gráficos y reportes; siendo una herramienta muy usada para el data science (Pilgrim, 2009).

2.3.8. SQL: Lenguaje de consulta de base de datos relaciones que sirve para acceder a información almacenada y gestionar el conjunto de objetos de la misma. Dado la gran presencia de las bases de datos relacionales en el mundo, su distribución y uso es también considerable convirtiéndose en una de las formas principales de obtención de datos (Oppel & Sheldon, 2009).

2.3.9. Programa académico: Es la denominación de los diversos currículos de estudio ofrecidos por las instituciones de educación con la finalidad de lograr una formación profesional. Estos están determinados por actividades académicas que el estudiante deberá de aprobar satisfactoriamente junto con otros aspectos necesarios para cada profesión (Universidad de Huánuco, 2018).

2.3.10. Indicadores: Es una medida de la realidad que puede ser usada para el análisis o evaluación. En su forma más simple solo expresa un valor o categoría el cual es usado con un fin concreto, así mismo su grado de complejidad puede aumentar hasta reflejar una razón de cambio, avances, criterios u otros (Japkowicz & Shah, 2011).

2.3.11. Inteligencia artificial: Es un área de las ciencias de la computación avocada al desarrollo de tecnología para que las máquinas realicen funciones cognitivas como si se tratase de un ser humano. Este es un campo muy amplio y en constante desarrollo por sus importantes beneficios en cuanto a su aplicación en el mundo real, siendo así también una de las disciplinas más nuevas en toda la historia (Mitchell, 1997).

2.3.12. Validación de modelos: Es el proceso mediante el cual se hace un control interno del funcionamiento del modelo algoritmo estudiado, que haciendo uso de métricas de desempeño logran obtener indicadores permiten ponderar la eficacia del modelo. Existen varias técnicas para aplicarse a este proceso, lo importante es garantizar que el conjunto de datos de entrenamiento y el de validación sean diferentes (David & Shwartz, 2014).

2.4. SISTEMA DE HIPÓTESIS

2.4.1 Hipótesis general

El nivel de eficacia en modelos algorítmicos presenta diferencias al estimar la deserción de los estudiantes del nivel pregrado en la Universidad de Huánuco.

2.4.2 Sistema de Variables

Variable Interviniente

Deserción de los estudiantes

Variable Independiente

Modelos algorítmicos

Variable Dependiente

Nivel de eficacia

2.5. OPERACIONALIZACIÓN DE VARIABLES

Tabla 1. Operacionalización de variables

VARIABLE	DEFINICIÓN CONCEPTUAL	DIMENSIONES	INDICADORES	INSTRUMENTO
INTERVINIENTE DESERCIÓN DE LOS ESTUDIANTES (Z)	La deserción como problema educativo es la interrupción del proceso formativo de un estudiante por diversos factores.	Personal	Edad.	Consultas SQL sobre la base de datos central de la UDH
			Sexo.	
		Académica	Modalidad de ingreso.	
			Programa académico.	
			Ciclo actual.	
			Número de cursos desaprobados.	
			Créditos matriculados.	
			Notas parciales.	
			Comparativa de promedios.	
		Económica	Retraso en el pago de pensiones.	
		Geográfica	Sede.	
		Temporal	Tipo de semestre.	
INDEPENDIENTE MODELOS ALGORÍTMICOS (X)	Estructura de pasos definidos que permiten obtener una salida o realizar algún proceso, haciendo posible su aplicación en un determinado problema modificando ciertos parámetros o adaptándolo a la realidad estudiada.	Funcional	Precision.	Matriz de confusión
			F1 score.	
			Recall.	
			AUC.	Curva ROC
		Operativa	Tiempo de ejecución.	Depuración de código
DEPENDIENTE NIVEL DE EFICACIA (Y)	Medida del cumplimiento de objetivos luego de aplicar alguna acción o realizar algún proceso.	Eficacia	Precisión del Modelo KNN	Validación cruzada
			Precisión del Modelo SVM	
			Precisión del Modelo MLP	
			Precisión del Modelo RNF	

Elaboración propia.

CAPÍTULO III

3. MARCO METODOLÓGICO

3.1. TIPO DE INVESTIGACIÓN

Tam, Vera, & Oliveros (2008) afirman que la investigación aplicada es el uso de conocimientos adquiridos, con o sin mayor refinamiento, para crear nueva tecnología a partir de estos y debe estar orientado hacia un propósito definido; cuya información generada a partir de este tipo de investigación debe ser aplicable en otros lugares, facilitando así su difusión.

En este sentido, ya que la presente investigación cumple con las condiciones anteriormente mencionadas y cuyo ámbito es el tecnológico, se puede afirmar que es una investigación aplicada.

3.1.1. Enfoque

Dado que se recolectarán y analizarán datos numéricos, o en su defecto se dará una ponderación numérica a aquellos valores categóricos, para el desarrollo de esta investigación se define un enfoque cuantitativo (Hernández, Fernández, & Baptista, 2014).

3.1.2. Alcance o nivel

El presente trabajo de investigación se abocará a recopilar la información necesaria que ayude a la explicación de la deserción universitaria. Dicha información será analizada e interpretada para darle el tratamiento respectivo y determinar las propiedades importantes del objeto en estudio, con la finalidad de tener cierto grado de predicción aplicando modelos algorítmicos. Asimismo, se medirá el nivel eficacia de estos para poder obtener el modelo más óptimo. En este sentido se optó por un alcance o nivel descriptivo (Del Cid, Méndez, & Sandoval, 2011).

3.1.3. Diseño

Considerando la mínima manipulación de la variable independiente y que no se posee grupo de control, el presente

trabajo de investigación posee un diseño pre experimental (Salas, 2013); con un solo grupo de estudio y una sola medición:

$$RG \quad \rightarrow \quad X \quad \rightarrow \quad O$$

Donde:

$\boldsymbol{RG}$: Grupo de estudio de selección aleatoria (dataset)

$\boldsymbol{X}$: Aplicación de modelos algorítmicos

$\boldsymbol{O}$: Observación

3.2. POBLACIÓN Y MUESTRA

La población está conformada por 127 332 casos de estudio, cada uno de ellos representa a un conjunto de atributos de un determinado estudiante al finalizar el semestre, dentro del periodo 2010-0 al 2018-2, y un campo adicional que representa si es un casó de deserción o no. Estos datos fueron obtenidos mediante consultas SQL a la base de datos central.

Para determinar la cantidad de casos que conformarán la muestra se utilizó la siguiente formula:

$$n = \frac{Z_{\alpha}^{2} N p q}{e^{2}(N-1) + Z_{\alpha}^{2} p q}$$

Donde:

$\boldsymbol{N}$: Tamaño de la población (127 332)

$\boldsymbol{\alpha}$: Nivel de confianza (99%)

$\boldsymbol{Z_{\alpha}}$: Constante del nivel de confianza (2,58)

$\boldsymbol{e}$: Error muestral (1%)

$\boldsymbol{p}$: Proporción de casos de deserción (50%)

$\boldsymbol{q}$: Proporción de casos de no deserción (50%)

$$\therefore n = 14\,717{,}67 \cong 14\,800$$

Por lo tanto, la muestra estría conformada por 14 800 casos y cuya composición necesaria es que haya igual número de casos de deserción como de no deserción.

3.3. TÉCNICAS E INSTRUMENTOS DE RECOLECCIÓN DE DATOS

Para la obtención de datos se tendrá acceso la base de datos central de la universidad, al cual solo se accede mediante consultas SQL. El objetivo es poder considerar todos los aspectos o atributos relacionados a un alumno para que a partir de ahí poder construir un dataset adecuado para su posterior procesamiento y análisis.

3.4. TÉCNICAS PARA EL PROCESAMIENTO Y ANÁLISIS DE INFORMACIÓN

En cuanto al desarrollo de la investigación se aplicarán técnicas contenidas en el área del data sience. El cual comprende desde la preparación del dataset, la aplicación de los modelos algorítmicos de clasificación, evaluación y su posterior análisis.

Se tomará como herramienta de trabajo el software Anaconda Navigator y se delimitará al uso de cuatro modelos algorítmicos de clasificación:

- KNN: K-nearest neighbors
- SVM: Support vector machines
- MLP: Multi-layer perceptron
- RNF: Random forest

La selección de estos modelos se debe al mejor manejo de estos por parte del investigador y referencias de uso en estudios diversos.

CAPÍTULO IV

4. RESULTADOS

4.1. DESCRIPCIÓN DE LA REALIDAD OBSERVADA

Todos los datos a considerar dentro del presente trabajo de investigación se encuentran en estructuras de datos conocidas como tablas relacionales, cada uno definido con un tipo de datos (Cadena, entero, decimal, fecha, booleano, etc.). Se desarrolló un script en lenguaje SQL con la finalidad de obtener toda la información relevante y verídica que se dispone en la institución (ver ANEXO 05). Se observó que existían ciertas inconsistencias en algunos datos por lo que se optó por excluir todos aquellos que no hayan sido validados o ingresados sin verificación previa; y realizar una nueva normalización de los datos cualitativos acorde a los lineamientos de esta investigación. En la Tabla 2. se detalla los atributos de los casos de estudio que fueron recopilados.

Tabla 2. Lista de atributos recopilados

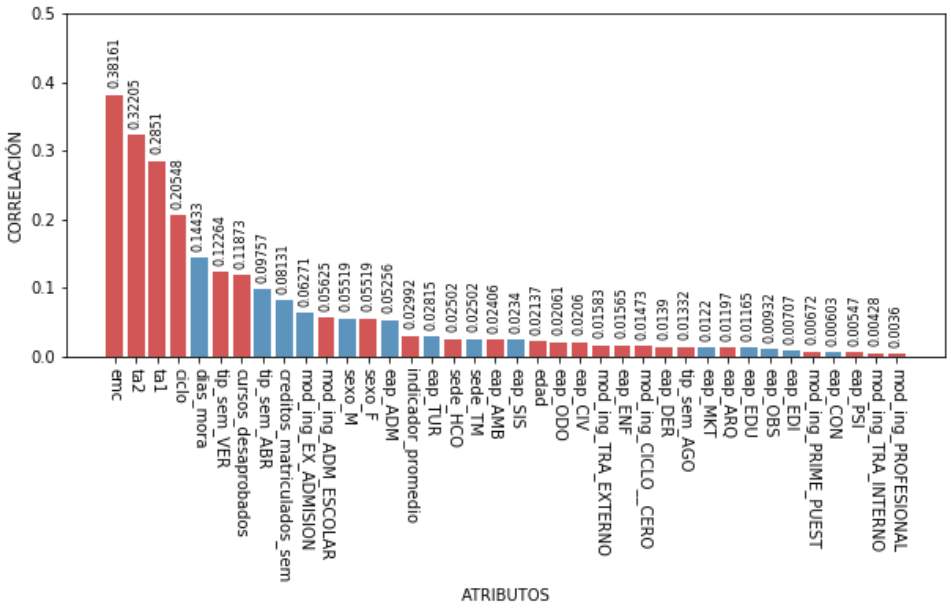
Nº	Nombre	Tipo	Dimensión
1	codigo	String	Personal
2	modalidad_ingreso	String	Académica
3	edad	Integer	Personal
4	eap	String	Académica
5	sexo	String	Personal
6	sede	String	Geográfica
7	semestre	String	Temporal
8	tipo_semestre	String	Temporal
9	ciclo	Integer	Académica
10	ta1	Integer	Académica
11	ta2	Integer	Académica
12	emc	Integer	Académica
13	creditos_matriculados_sem	Integer	Académica
14	cursos_desaprobados	Integer	Académica
15	indicador_promedio	Double	Académica
16	dias_mora	Integer	Económica
17	desercion	Boolean	Atributo de estudio

Elaboración propia.

Dicha totalidad de atributos se sometieron a una transformación para completar cierta información faltante, codificación de atributos categóricos y quitar aquellos considerados para el control de errores pero que una vez concluido dichos procesos ya no eran necesarios, cabe recalcar que para estos fines se utilizó técnicas basadas en data science (ver ANEXO 08).

Con ayuda del software se calculó la correlación que existía entre los diferentes atributos obtenidos con la deserción, esto se muestra en la Figura 14 donde la correlación positiva y negativa están representados por los colores azul y rojo respectivamente. Cabe recalcar que siendo este un coeficiente de correlación se puede asumir que aquellos valores cercanos a cero representan información que no tiene relevancia para el estudio de la deserción, esto genera la duda sobre que atributos pueden ser descartados debido a que hasta este punto se tenían 38 atributos documentados y una disminución de estos ayudaría a mejorar la eficacia de los modelos.

Figura 14. Correlación de atributos con la deserción



Elaboración propia.

Un análisis más exhaustivo de la relación de atributos y su relevancia se logró mediante la aplicación de ciertas técnicas de selección de características (métodos estadísticos y algoritmos de data science), estos realizan una ponderación de los atributos en relación con la deserción estableciendo cuales son útiles para un problema de clasificación. Las técnicas usadas fueron:

- A: Pearson Correlation
- B: Chi-Squared
- C: Recursive Feature Elimination
- D: Lasso
- E: Tree-based
- F: LightGBM

Tabla 3. Lista de atributos seleccionados

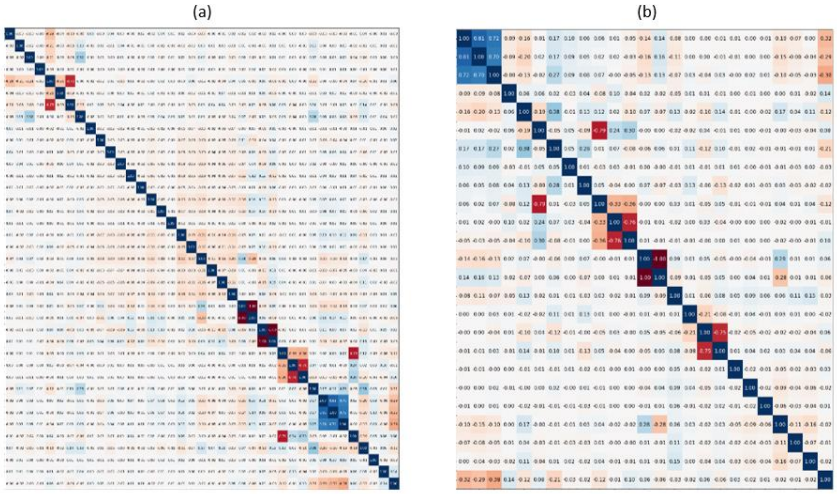
Nº	Nombre	A	B	C	D	E	F	Total
1	ta2	1	1	1	1	1	1	6
2	ta1	1	1	1	1	1	1	6
3	emc	1	1	1	1	1	1	6
4	dias_mora	1	1	1	1	1	1	6
5	cursos_desaprobados	1	1	1	1	1	1	6
6	creditos_matriculados_sem	1	1	1	1	1	1	6
7	ciclo	1	1	1	1	1	1	6
8	indicador_promedio	1	0	1	1	1	1	5
9	edad	1	0	1	1	1	1	5
10	tip_sem_VER	1	1	1	1	0	0	4
11	tip_sem_AGO	1	1	1	1	0	0	4
12	tip_sem_ABR	1	1	1	1	0	0	4
13	sexo_M	1	1	1	1	0	0	4
14	sexo_F	1	1	1	1	0	0	4
15	sede_HCO	1	1	1	1	0	0	4
16	mod_ing_TRA_EXTERNO	1	1	1	1	0	0	4
17	mod_ing_EX_ADMISION	1	1	1	1	0	0	4
18	mod_ing_ADM_ESCOLAR	1	1	1	1	0	0	4
19	eap_TUR	1	1	1	1	0	0	4
20	eap_ODO	1	1	1	1	0	0	4
21	eap_MKT	1	1	1	1	0	0	4
22	eap_CIV	1	1	1	1	0	0	4
23	eap_ARQ	1	1	1	1	0	0	4
24	eap_AMB	1	1	1	1	0	0	4

Elaboración propia.

Cada uno de estos establecieron un valor de 1 si se consideraba que eran útiles y 0 si podían ser descartados. Todo esto se trabajó de manera conjunta para poder aprovechar todas las técnicas disponibles, siendo así que se obtuviera la suma de cada uno de estos como un puntaje total que se muestra en la Tabla 3. Se seleccionó aquellos atributos que tuvieran un puntaje de 4 o más, en otras palabras, que hayan sido considerados como mínimo por 4 técnicas como atributos relevantes (ver ANEXO 09).

Los atributos que se muestran en la Tabla 3 son aquellos que se van a ser considerados para el presente estudio, en dicha conjetura cabe la necesidad de verificar que si la elección de estos conlleva a una mejora en cuanto a la información que brindan. Un mapa de calor corresponde a la representación de la intensidad de cierto factor sobre un conjunto de datos, en este caso aprovechando la obtención de una matriz de correlación se procedió a verificar un antes y un después de los datos.

Figura 15. Comparación de mapa de calor



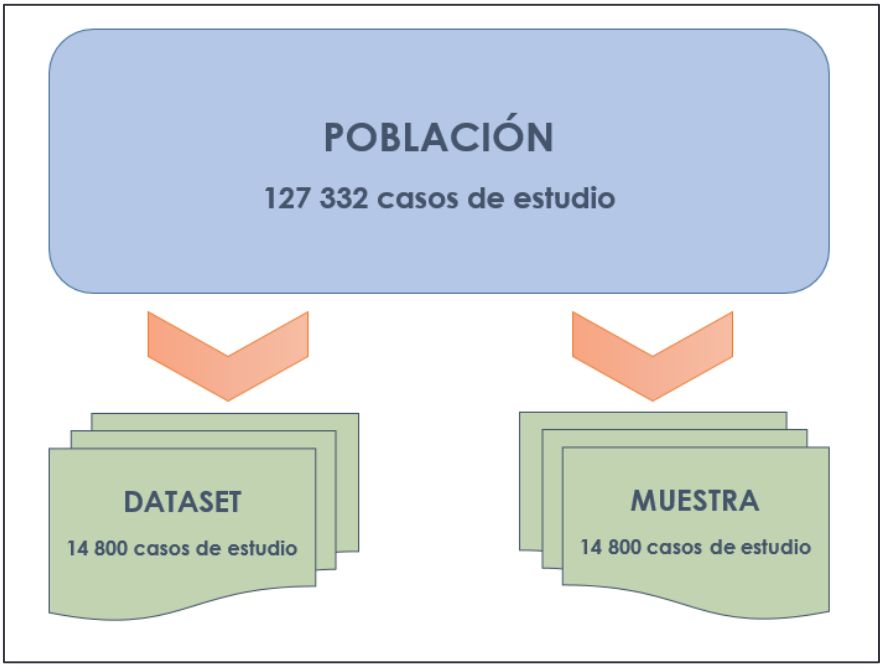
Elaboración propia.

El la Figura 15 se muestra al lado izquierdo o lado a, la correlación que existe antes de aplicar la selección de atributos. En el lado izquierdo o lado b, se muestra la correlación que existe luego

de aplicar la selección de atributos; se puede verificar la reducción de dimensionalidad (Por el tamaño de cada cuadro dentro del gráfico) y la presencia de mayor correlación (Por la coloración roja o azul).

En el análisis de la composición de los datos en la población se observó que las clases estaban desbalanceadas respecto al atributo en estudio (109 888 no deserción y 17 444 deserción), lo cual también determinó que se trataba de un problema de clasificación binaria. Debido a la naturaleza del presente trabajo de investigación que utiliza dos conjuntos de datos (Entrenamiento y validación), se procedió a extraer dichos conjuntos manera aleatoria cuya característica fundamental es que las clases estén balanceadas; cada conjunto poseía un total de 14 800 instancias, al conjunto de entrenamiento se le llamo dataset y al conjunto de validación se le llamo muestra (Ver ANEXO 08).

Figura 16. Selección de muestra y dataset



Elaboración propia.

Con esto se logra cumplir el Objetivo Especifico 1 del presente trabajo de investigación.

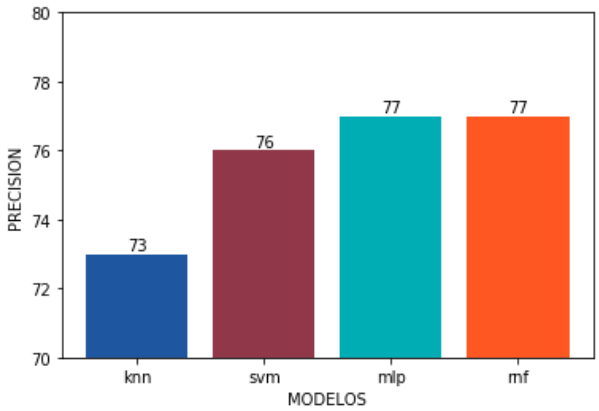
4.2. PROCESAMIENTO DE DATOS

En esta parte se llevó a cabo la ejecución de los modelos algorítmicos, para esto se desarrolló una clase “Model” que permitió un trabajo más ordenado y sistematizado (Ver ANEXO 06). Dicha clase no solo permitió definir los formatos de entrada y salida, sino que también permitió definir una especie de plantilla para la ejecución del entrenamiento de los modelos algorítmicos; uno de los aspectos fundamentales en este tipo de problemas es que los modelos puedan ser exportados luego de entrenamiento para que puedan ser almacenados en algún espacio de memoria para su uso posterior, la clase desarrollada tiene esta funcionalidad implementada para facilitar dicho proceso.

Cabe recalcar que el entrenamiento no consiste en una ejecución única, sino que es necesario varias ejecuciones iterativas hasta llegar a conseguir adaptar correctamente los modelos algorítmicos al problema en estudio y lograr un resultado óptimo (ver ANEXO 10). Se utilizó la técnica de búsqueda aleatoria de parámetros combinado con validación cruzada, que permitió conseguir mejores valores. Se detalla a continuación las métricas de desempeño obtenidas por los cuatro modelos algorítmicos:

Precisión

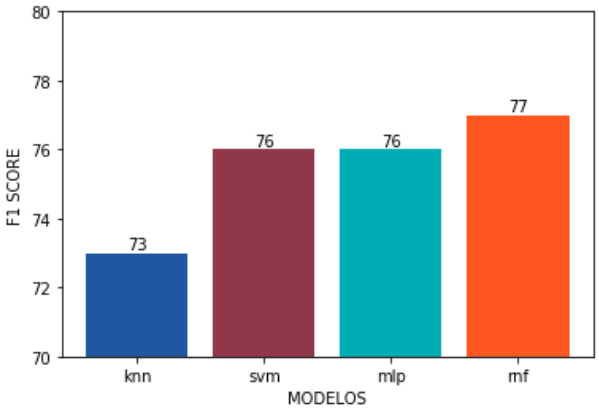
Figura 17. Precision en el entrenamiento de los modelos



Elaboración propia.

F1 score

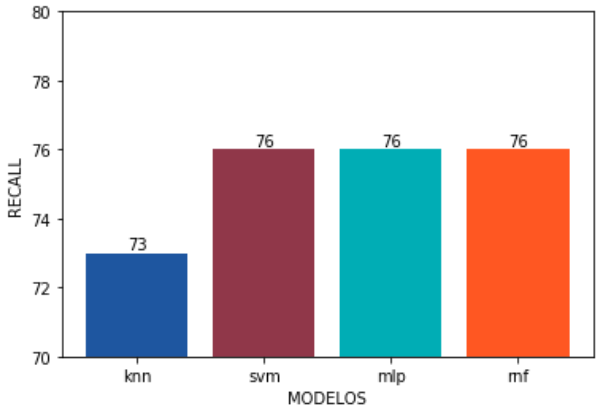
Figura 18. F1 score en el entrenamiento de los modelos



Elaboración propia.

Recall

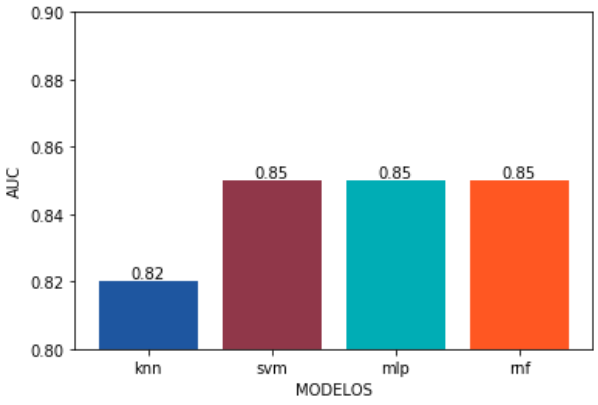
Figura 19. Recall en el entrenamiento de los modelos



Elaboración propia.

AUC

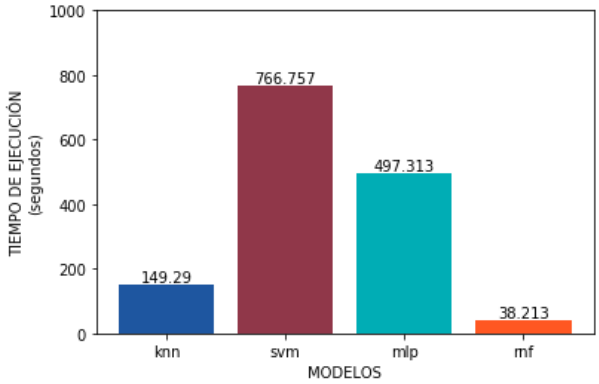
Figura 20. AUC en el entrenamiento de los modelos



Elaboración propia.

Tiempo de ejecución

Figura 21. Tiempo de ejecución en el entrenamiento de los modelos



Elaboración propia.

Las métricas de desempeño pueden dar indicios de la eficacia de los modelos algorítmicos, en la Tabla 4 se hace una consolidado de todas métricas obtenidas durante el entrenamiento de los modelos.

Tabla 4. Consolidado de métricas de desempeño (Entrenamiento)

Modelo	Precision	F1 score	Recall	AUC	T. de ejecución
knn	73	73	73	0.82	149.29 seg
svm	76	76	76	0.85	766.757 seg
mlp	77	76	76	0.85	497.313 seg
rnf	77	77	76	0.85	38.213 seg

Elaboración propia.

Los modelos algorítmicos fueron exportados utilizando el módulo pickle (protocolos binarios para serializar), el cual es ideal para el almacenamiento de objetos.

Tabla 5. Información de modelos algorítmicos exportados

	Modelos			
	knn	svm	mlp	rnf
Tiempo (seg.)	0.024	0.000	0.000	0.047
Tamaño (KB)	4789	1340	55	14 840

Elaboración propia.

Con esto se logra cumplir el Objetivo Especifico 2 del presente trabajo de investigación.

4.3. CONTRASTACIÓN Y PRUEBA DE HIPÓTESIS

Con los modelos algorítmicos entrenados y aptos para poder estimar la deserción, se procedió a elegir una métrica para poder medir la eficacia de los modelos. Según Japkowicz & Shah (2011), la evaluación de los algoritmos de aprendizaje tienen como uno de sus componentes principales a las métricas de desempeño; entendiendo que la precisión pertenece a dichas métricas y es una medición que ayuda a cuantificar la tasa de acierto de los casos positivos (En el presente estudio estos son los estudiantes propensos a la deserción), se establece a este como la métrica elegida para realizar la comparación del nivel de eficacia entre los modelos algorítmicos estudiados.

Los datos se organizaron mediante 10 divisiones estratificadas (Folds) del conjunto de datos llamado muestra y para ejecutar la estimación de la deserción se importaron los modelos algorítmicos con la clase “Model”; los datos obtenidos luego de la estimación consistían en las predicciones realizadas por los modelos para posteriormente obtener las métricas necesarias para la presente investigación (Ver ANEXO 11). Los resultados obtenidos luego de la ejecución se pueden observar en la Tabla 6.

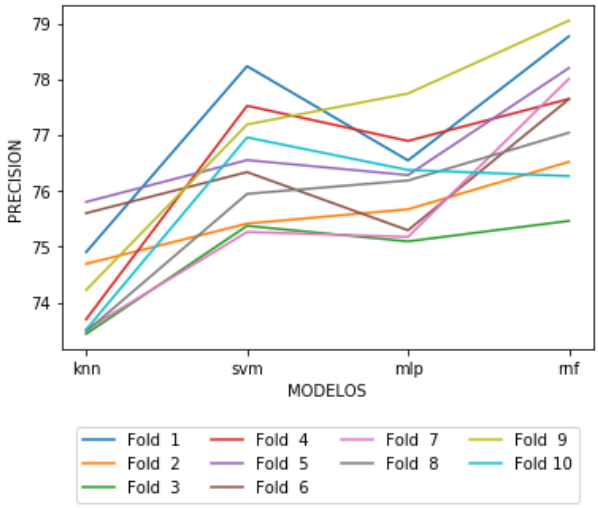
Tabla 6. Precision de los modelos algoritmicos

	Modelos			
	knn	**svm**	**mlp**	**rnf**
Fold 1	74.90362846	78.24031739	76.54932052	78.77662127
Fold 2	74.69322439	75.41585840	75.67228883	76.52446161
Fold 3	73.43586992	75.37535766	75.09697810	75.46427418
Fold 4	73.69845466	77.52976190	76.89903842	77.65275674
Fold 5	75.80424976	76.55513766	76.28553014	78.20783486
Fold 6	75.60267857	76.33955753	75.29624708	77.64887679
Fold 7	73.51155767	75.26578765	75.17744818	78.01394603
Fold 8	73.47790003	75.94938578	76.18886932	77.04678363
Fold 9	74.22353742	77.19402566	77.75000000	79.05887732
Fold 10	73.51318692	76.96177958	76.37798684	76.26849777

Elaboración propia.

En la Figura 22 se muestra la comparativa de la presicion, el orden de los modelos fue de manera aleatoria y no representa ningún tipo se secuencialidad; las lineas muestran las diversas tendencias del nivel de eficacia de los modelos algorítmicos frente al problema de estimar la deserción en la Universidad de Huánuco.

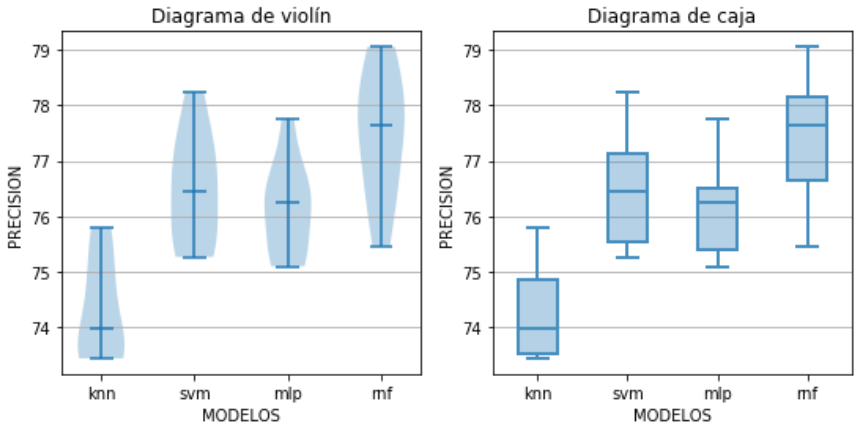
Figura 22. Precision de los modelos



Elaboración propia.

Como información adicional tenemos a dos diagramas en la Figura 23 que muestran cómo se distribuyen los resultados obtenidos anteriormente.

Figura 23. Diagramas de comparación de precision de los modelos



Elaboración propia.

Con esto se logra cumplir el Objetivo Especifico 3 del presente trabajo de investigación.

Según Japkowicz & Shah (2011), la prueba ANOVA de medidas repetidas permite comparar el rendimiento de varios clasificadores en varios conjuntos de datos para determinar si existe diferencia significativa, y que como complemento la prueba post hoc HSD de Tukey realiza la comparación del rendimiento de pares de clasificadores. La formulación anterior se adecua correctamente a la presente investigación dado que se tienen cuatro modelos algorítmicos o clasificadores, y diez conjuntos de datos; en este sentido la prueba ANOVA de medidas repetidas tiene dos supuestos (Normalidad y esfericidad) que fueron comprobados y se detalla en la Tabla 7 y Tabla 8, donde se demostró que se cumplen dichas condiciones (Ver ANEXO 12).

Tabla 7. Prueba de Shapiro–Wilk (Normalidad)

	Conjunto de datos			
	knn	**svm**	**mlp**	**rnf**
Valor de prueba (Prob.)	0.0539	0.6149	0.5745	0.8993
Valor crítico (Prob.)	0.05	0.05	0.05	0.05
Resultado	*Normalidad aceptada*			

Nota: H_o= Los datos provienen de una población normalmente distribuida. Elaboración propia.

Tabla 8. Prueba de Bartlett (Esfericidad)

	Conjunto de datos
	knn + svm + mlp + rnf
Valor de prueba (Prob.)	0.8279
Valor crítico (Prob.)	0.05
Resultado	*Esfericidad aceptada*

Nota: H_o= Todas las varianzas poblacionales son iguales. Elaboración propia.

Dichas consideraciones muestran que se hace viable la ejecución de la prueba ANOVA de medidas repetidas con un nivel de confianza de 1%.

$$H_o: \mu_{knn} = \mu_{svm} = \mu_{mlp} = \mu_{rnf}$$

$$H_1: \mu_{knn} \neq \mu_{svm} \neq \mu_{mlp} \neq \mu_{rnf}$$

$$\alpha = 0.01$$

Tabla 9. Prueba ANOVA de medidas repetidas

	SC	GL	MC	F	P
Inter Grupos	53.0281	3	17.6760	32.4059	0.0000
Intra Grupos	34.6213	36			
Error	14.7273	27	0.5455		
Sujetos	19.8939	9			
TOTAL	87.6494	39			

Elaboración propia.

En la Tabla 9 se muestra un valor de probabilidad de aproximadamente 0.0000, por lo que se concluye que existe diferencias en el nivel de eficacia en modelos algorítmicos y se comprueba la hipótesis planteada en el presente trabajo de investigación.

Comprobada la existencia de diferencias entre los niveles de eficacia de algunos modelos algoritmos se procedió a realizar la prueba post hoc HSD de Tukey con un nivel de confianza de 1% (Ver ANEXO 12).

Tabla 10. Prueba HSD de Tukey

Grupo 1	Grupo 2	Diferencia	P	Resultado
knn	mlp	1.8429	0.001	Rechazar
knn	rnf	3.1799	0.001	Rechazar
knn	svm	2.1963	0.001	Rechazar
mlp	rnf	1.3369	0.0214	Aceptar
mlp	svm	0.3533	0.835	Aceptar
rnf	svm	-0.9836	0.131	Aceptar

Nota: H_o= Las medias de Grupo 1 y Grupo 2 son iguales. Elaboración propia.

En la Tabla 10 se muestran los resultados de la comparación por pares, donde se confirma que el modelo knn presenta diferencia con todos los demás modelos; y que los modelos svm, mlp y rnf no presentan diferencias.

Con esto se logra cumplir el Objetivo General del presente trabajo de investigación y se comprueba la hipótesis formulada.

CAPÍTULO V

5. DISCUSIÓN

5.1 CONTRASTACIÓN DE RESULTADOS

El presente trabajo de investigación se desarrolla dentro del ámbito del data science, esto conlleva a que se analice cierto fenómeno para poder extraer información muy valiosa y que permita reconocer aspectos que muchas veces son obviados por no ser considerados importantes o relevantes; dentro de todo el proceso se puso a evaluación cuatro modelos algorítmicos. Los resultados obtenidos en este trabajo de investigación, muestran que existe diferencia en el nivel de eficacia de los modelos algorítmicos aplicados y por lo tanto se acepta la hipótesis formulada.

A aplicar técnicas de análisis se pudo cuantificar el grado de correlación que se tenía entre la información recopilada y la deserción, siendo los atributos más importantes los siguientes:

- Académico: ta2, ta1, emc, cursos_desaprobados, creditos_matriculados_sem, ciclo, indicador_promedio, mod_ing_TRA_EXTERNO, mod_ing_EX_ADMISION, mod_ing_ADM_ESCOLAR, eap_TUR, eap_ODO,eap_MKT, eap_CIV, eap_ARQ, eap_AMB
- Economico: dias_mora
- Personal: edad, sexo_M, sexo_F
- Temporal: tip_sem_VER, tip_sem_AGO, tip_sem_ABR
- Geográfica: sede_HCO

Esto se podría definir como cierto perfil estudiantil determinante para identificar la deserción, siendo así que los otros atributos se les puede considerar no significativos. Se puede observar que los atributos seleccionados son de diferentes

dimensiones y no solo de la académica, esto refuerza la idea de que la deserción universitaria no solo está relacionada con el ambiente educativo o académico, sino que es una circunstancia en cual intervienen varios factores (Himmel, 2002), y por lo tanto se considera también un fenómeno social (Páramo & Correa, 1999).

En la Tabla 4. Consolidado de métricas de desempeño (Entrenamiento) se muestran las métricas de desempeño durante el proceso de entrenamiento de los modelos algorítmicos, dichas métricas no se consideran como determinantes al momento de realizar la comparación de los modelos, sino que aportan valiosa información en cuanto al nivel de eficacia de cada uno de los modelos en esta etapa (Japkowicz & Shah, 2011). El presente trabajo de investigación pudo haber considerado cualquiera de las métricas para realizar la comparación, pero por la naturaleza del problema en estudio se optó por trabajar con solo una de ellas para facilitar el proceso de evaluación. Durante toda la etapa de entrenamiento y evaluación de los modelos se hizo uso de software implementadas por el investigador que facilitó mucho en cuanto a la ejecución de procesos repetitivos y a los reportes. Respecto al proceso de evaluación de los modelos la métrica de precisión ya mostraba ciertas diferencias como se puede notar en la Figura 22. Precision de los modelos y Figura 23. Diagramas de comparación de precision de los modelos, y cuya prueba estadística posterior confirmó la presencia de diferencia en por lo menos uno de ellos. En la Tabla 10. Prueba HSD de Tukey se puede verificar un análisis más detallado por cada par de modelos, donde se observa las diferencias existentes entre los valores de precisión obtenidos; siguiendo está prueba se puede determinar el siguiente orden del nivel de eficacia:

1. RNF: Random forest
2. SVM: Support vector machines
3. MLP: Multi-layer perceptron

4. KNN: K-nearest neighbors

Se considera que el modelo algorítmico RNF ha demostrado mayor eficacia que los otros modelos. Esto no conlleva a afirmar que los demás modelos sean más deficientes al momento de predecir la deserción de un alumno, sino más bien que los modelos RNF, SVM y MLP comparten cierto grado de eficacia aceptable (75%); la selección del mejor modelo dependerá de otros factores adicionales tales como tiempo de procesamiento, facilidad de implementación, u otros aspectos a considerar al momento de la implementación.

En comparación con el estudio realizado por Hernández *et al.* (2016), una de las principales diferencias es la cantidad de casos estudiados; si bien es cierto la cantidad de casos que se estudian no son tan determinantes, pero se siguiere el uso de una mayor cantidad de casos posibles a manera que el modelo logre un aprendizaje optimo. La poca cantidad de casos explicaría porque en dicho estudio los modelos lleguen a un rendimiento del 100%, ya que en otros casos como el presente trabajo de investigación se logran porcentajes menores, pero por la cantidad de casos se puede considerar que el proceso de aprendizaje ha sido satisfactorio.

Vásquez (2016) utiliza modelos basados en SVM y redes neuronales, esto puede entenderse como una desventaja frente a otros estudios donde se analiza un número mayor de modelos, pero también existe la posibilidad de realizar un estudió más a profundidad cuando solo son dos los modelos a evaluar. Como el presente trabajo de investigación se centra en realizar una comparación, la variedad de modelos elegidos resulta una ventaja al momento de obtener resultados dado que servirá para estudios futuros donde se puede profundizar a mayor detalle con aquellos que muestren mayor nivel de eficacia.

En el caso de los estudios de Galvez & Flores (2015) y Daza (2016), se puede observar que los modelos que obtuvieron mejores resultados son los que están basados en arboles de decisión. Los resultados de la presente investigación refuerzan está idea ya que el modelo algorítmico que obtuvo mayor nivel de eficacia es RNF.

CONCLUSIONES

A pesar de que el data science provea datos cuantitativos verídicos, en la presente investigación se aplicaron satisfactoriamente pruebas estadísticas a los resultados para determinar la presencia de diferencias en cuanto al nivel de eficacia de cuatro modelos algorítmicos.

Basado en la prueba estadística realizada sobre la precisión de los modelos, el mejor modelo es RNF (Random forest), y el peor modelo es KNN (K-nearest neighbors).

La aplicación de machine learning puede ayudar a transformar y estructurar la información que se disponga de un fenómeno en estudio, ya que provee una multitud de opciones de análisis y algoritmos.

Con la ayuda de un lenguaje de programación (Pyhton) se puede realizar la adaptación y el entrenamiento de modelos algorítmicos, a su vez se pueden desarrollar herramientas que faciliten ciertos procesos.

El nivel de eficacia puede ser medido con la precisión de los modelos, ya que es una métrica de desempeño muy adecuada para este tipo de casos. Para este caso en particular los modelos algorítmicos obtuvieron una precisión alrededor del 75%.

El presente estudio es una muestra de que los modelos algorítmicos no son una realidad alejada a nuestra realidad, sino muy por el contrario son nuevas tecnología que aplicadas correctamente nos brindarían muchas facilidad y nuevas oportunidades.

RECOMENDACIONES

Para estudios posteriores se debe entender que la deserción universitaria no es solo un problema académico, sino que debe abarcarse todas las áreas posibles de manera que se realice un diagnóstico correcto.

De acuerdo a la información recopilada, se recomienda que la institución ponga especial cuidado en los procesos de recolección de datos ya se encontró ciertas falencias en la información provista.

Con base a este estudio se puede realizar una futura implementación de dichos modelos en un software para realizar el seguimiento al desempeño del estudiante de acuerdo a la realidad de cada programa académico.

Se recomienda que la institución pueda extender la aplicación del machine learning a otros procesos que considere necesarios, ya que ha quedado demostrado que dicha aplicación permite reconocer patrones o criterios de correlación dentro de la información estudiada lo cual es una gran herramienta de análisis.

REFERENCIAS BIBLIOGRÁFICAS

Contreras, F. (2016). *Introducción a Machine Learning.* Obtenido de ZEMSANIA, S.L.: https://sunqu.net/white-papers-machine-learning/

David, S., & Shwartz, S. (2014). *Understanding machine learning.* New York: Cambridge University Press.

Daza, A. (2016). Un modelo basado en árboles de decisión para predecir la deserción estudiantil en la Educación Superior Privada. *UCV-SCIENTIA, VIII*(1), 59-73.

Del Cid, A., Méndez, R., & Sandoval, F. (2011). *INVESTIGACIÓN. Fundamentos y metodología* (Segunda ed.). México: Pearson Educación.

Flach, P. (2012). *Machine Learning. The art and Science of Algorithms that Make Sense Data.* New York: Cambridge University Press.

Galvez, M., & Flores, K. (2015). *Modelo predictivo de deserción universitaria de la carrera de Ingeniería Informática en la Universidad Ricardo Palma.* Tesis de pregrado, Universidad Ricardo Palma, Lima.

Hernández, A., Meléndez, R., Morales, L., Barrientos, A., Tecpanecatl, J., & Algredo, I. (2016). Comparative Study of Algorithms to Predict the Desertion in the Students at the ITSM-Mexico. *IEEE Latin America Transactions, XIV*(11), 4573-4578.

Hernández, R., Fernández, C., & Baptista, P. (2014). *Metodología de la investigación* (Sexta ed.). México D.F.: McGraw-Hill.

Himmel, E. (2002). Modelo de análisis de la deserción estudiantil en la educación superior. *Calidad en la Educación*(17), 91-108. doi:10.31619/caledu.n17.409

Japkowicz, N., & Shah, M. (2011). *Evaluating learning algorithms.* New York: Cambridge University Press.

Leek, J. (2015). *The Elements of Data Analytic Style.* Leanpub. Obtenido de http://leanpub.com/datastyle

Mitchell, T. (1997). *Machine Learning.* New York: McGraw-Hill, Inc.

Oppel, A., & Sheldon, R. (2009). *Fundamentos de SQL* (Tercera ed.). (C. Jiménez, Trad.) México: McGraw-Hil, Inc.

Páramo, G., & Correa, C. (1999). Deserción estudiantil universitaria. Conceptualización. *Revista Universidad EAFIT, XXXV*(114), 65-78.

Peng, R., & Matsui, E. (2018). *The Art of Data Science.* Leanpub. Obtenido de https://leanpub.com/artofdatascience

Pilgrim, M. (2009). *Inmersión en Python 3.* (J. González, Trad.) Obtenido de http://code.google.com/p/inmersionenpython3

Salas, E. (2013). Diseños preexperimentales en psicología y educación: una revisión conceptual. *Liberabit. Revista de Psicología*, 133-141.

Stanton, J. (2013). *Introduction to Data Science.* Syracuse University. Obtenido de https://archive.org/details/DataScienceBookV3

Tam, J., Vera, G., & Oliveros, R. (2008). Tipos, métodos y estrategias de investigación científica. *Pensamiento y acción*, 145-154.

Universidad de Huánuco. (2018). Plan estratégico de desarrollo institucional 2018 – 2023. Huánuco, Huánuco, Perú.

Vásquez, J. (2016). *Modelo predictivo para estimar la deserción de estudiantes en una institución de educación superior.* Tesis de maestría, Universidad de Chile, Santiago.

Witten, I., & Frank, E. (2005). *Data mining : practical machine learning tools and techniques* (Segunda ed.). San Francisco: Elsevier.

Zaki, M., & Meira, W. (2014). *Data mining and analysis: fundamental concepts and algorithms.* New York: Cambridge University Press.

ANEXOS

ANEXO 01

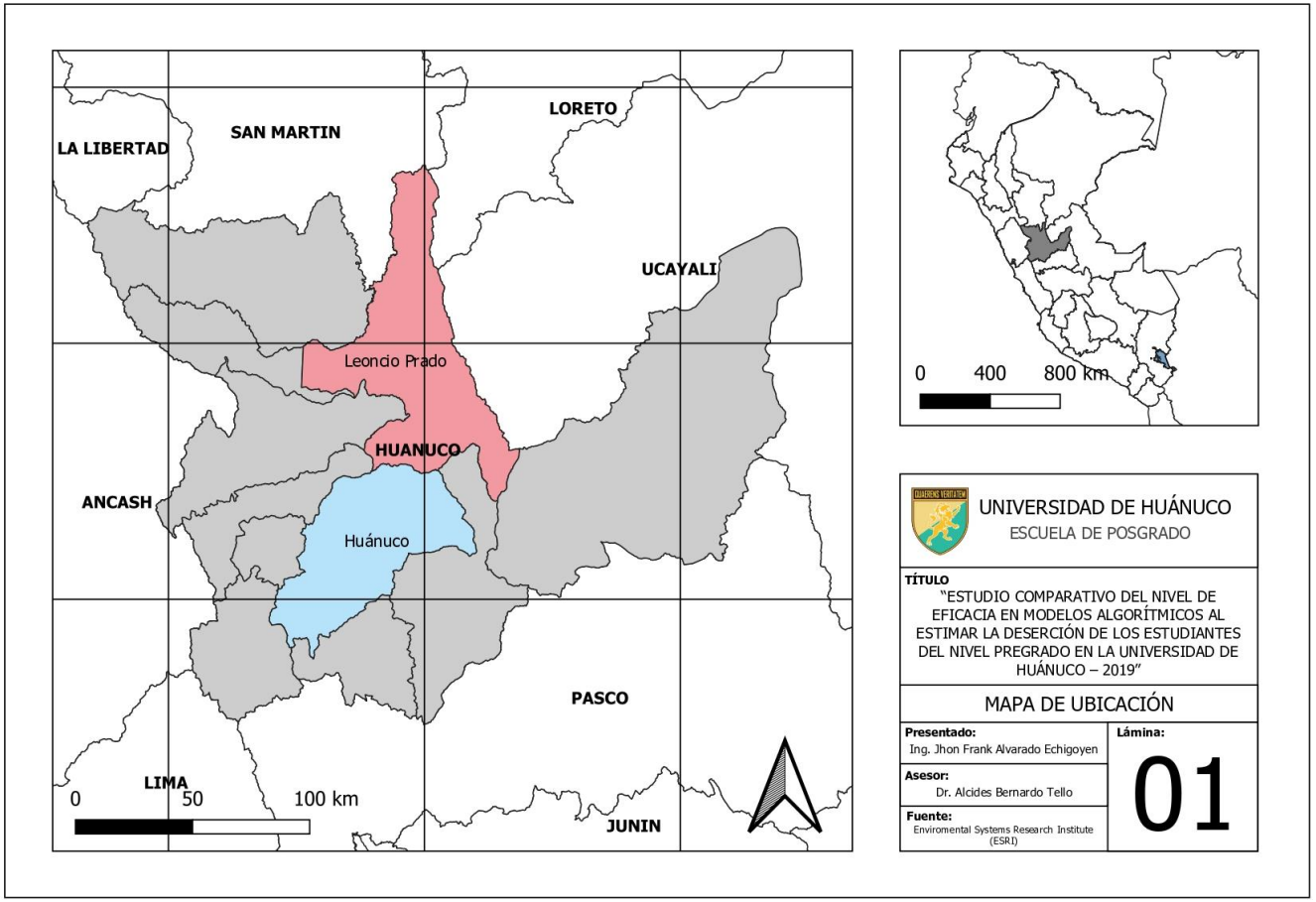
MATRIZ DE CONSISTENCIA

TÍTULO: "ESTUDIO COMPARATIVO DEL NIVEL DE EFICACIA EN MODELOS ALGORÍTMICOS AL ESTIMAR LA DESERCIÓN DE LOS ESTUDIANTES DEL NIVEL PREGRADO EN LA UNIVERSIDAD DE HUÁNUCO – 2019"

PROBLEMA	OBJETIVO	HIPÓTESIS	VARIABLES	DIMENSIONES	INDICADORES	METODOLOGÍA
GENERAL ¿Es diferente el nivel de eficacia en modelos algorítmicos al estimar la deserción de los estudiantes del nivel pregrado en la Universidad de Huánuco? **ESPECÍFICOS** (PE1) ¿Cómo transformar y estructurar la información actual en un conjunto de instancias para su posterior tratamiento? (PE2) ¿Cómo adaptar y entrenar cuatro algoritmos de clasificación? (PE3) ¿Cuál es la eficacia de cuatro modelos algorítmicos al estimar la deserción de los estudiantes del nivel pregrado en la Universidad de Huánuco?	**GENERAL** Comparar el nivel de eficacia en modelos algorítmicos al estimar la deserción de los estudiantes del nivel pregrado en la Universidad de Huánuco. **ESPECÍFICOS** (OE1) Transformar y estructurar la información actual en un conjunto de instancias para su posterior tratamiento. (OE2) Realizar la adaptación y entrenamiento de cuatro algoritmos de clasificación. (OE3) Determinar la eficacia de cuatro modelos algorítmicos al estimar la deserción de los estudiantes del nivel pregrado en la Universidad de Huánuco.	El nivel de eficacia en modelos algorítmicos presenta diferencias al estimar la deserción de los estudiantes del nivel pregrado en la Universidad de Huánuco.	**INTERVINIENTE** DESERCIÓN DE LOS ESTUDIANTES (Z)	Personal	Edad.	**TIPO DE INVESTIGACIÓN** Aplicada **ENFOQUE** Cuantitativo **ALCANCE O NIVEL** Descriptivo **DISEÑO** Pre experimental **POBLACIÓN** 127 332 casos de estudio **MUESTRA** 14 800 casos de estudio
					Sexo.	
				Académica	Modalidad de ingreso.	
					Programa académico.	
					Ciclo actual.	
					Número de cursos desaprobados.	
					Créditos matriculados.	
					Notas parciales.	
					Comparativa de promedios.	
				Económica	Retraso en el pago de pensiones.	
				Geográfica	Sede.	
				Temporal	Tipo de semestre.	
			INDEPENDIENTE MODELOS ALGORÍTMICOS (X)	Funcional	Precision.	
					F1 score.	
					Recall.	
					AUC.	
				Operativa	Tiempo de ejecución.	
			DEPENDIENTE NIVEL DE EFICACIA (Y)	Eficacia	Precisión del Modelo KNN	
					Precisión del Modelo SVM	
					Precisión del Modelo MLP	
					Precisión del Modelo RNF	

ANEXO 02

MAPA DE UBICACIÓN



ANEXO 03

PROGRAMAS ACADÉMICOS

Tabla 11. Programas académicos del nivel pregrado de la UDH

Nº	Descripción	Estado
1	Derecho y Ciencias Políticas	Activo
2	Obstetricia	Activo
3	Enfermería	Activo
4	Odontología	Activo
5	Psicología	Activo
6	Ingeniería de Sistemas e Informática	Activo
7	Ingeniería Civil	Activo
8	Arquitectura	Activo
9	Ingeniería Ambiental	Activo
10	Educación Básica: Inicial y Primaria	Activo
11	Contabilidad y Finanzas	Activo
12	Administración de Empresas	Activo
13	Marketing y Negocios Internacionales	Activo
14	Turismo, Hotelería y Gastronomía	Activo
15	Educación: Especialidad Idioma Extranjero Inglés	No activo

Elaboración propia.

ANEXO 04

CANTIDAD DE DESERCIÓN

Tabla 12. Deserción de los últimos 2 años en la UDH

Semestre	Nivel	Cantidad
2017-1	Pregrado	2928
	Posgrado	132
	Segunda Especialización	23
2017-2	Pregrado	2274
	Posgrado	148
	Segunda Especialización	43
2018-1	Pregrado	2027
	Posgrado	144
	Segunda Especialización	33
2018-2	Pregrado	1895
	Posgrado	152
	Segunda Especialización	24

Nota. Abril 2019. Elaboración propia.

ANEXO 05

SCRIPT SQL PARA RECOLECCIÓN DE INFORMACIÓN

```sql
--PyML
--Jhon Frank

--*TABLAS TEMPORALES DE LAS TABLAS REALES
-- #tmp_alumnos
-- #tmp_sede
-- #tmp_registro_electronico
-- #tmp_cursos
-- #tmp_matricula_semestre
-- #tmp_boletas
-- #tmp_detalle_boletas

--*TABLAS TEMPORALES PARA NORMALIZACIÓN DE DATOS
--*Escuelas
IF OBJECT_ID('tempdb..#escuela') IS NOT NULL
    DROP TABLE #escuela

SELECT 1  codesc, 'DER' desesc INTO #escuela UNION
SELECT 14 codesc, 'CIV' desesc UNION
SELECT 15 codesc, 'ARQ' desesc UNION
SELECT 2  codesc, 'OBS' desesc UNION
SELECT 31 codesc, 'EDI' desesc UNION
SELECT 34 codesc, 'ADM' desesc UNION
SELECT 35 codesc, 'CON' desesc UNION
SELECT 36 codesc, 'TUR' desesc UNION
SELECT 38 codesc, 'PSI' desesc UNION
SELECT 39 codesc, 'AMB' desesc UNION
SELECT 4  codesc, 'SIS' desesc UNION
SELECT 49 codesc, 'MKT' desesc UNION
SELECT 5  codesc, 'ENF' desesc UNION
SELECT 6  codesc, 'EDU' desesc UNION
SELECT 7  codesc, 'OOO' desesc

--*Forma ingreso
IF OBJECT_ID('tempdb..#forma_ingreso') IS NOT NULL
    DROP TABLE #forma_ingreso

SELECT 1    codigo_modalidad_ingreso, 'EX_ADMISTON' modalidad_ingreso INTO #forma_ingreso
UNION
SELECT 2    codigo_modalidad_ingreso, 'TRA_INTERNO' modalidad_ingreso UNION
SELECT 3    codigo_modalidad_ingreso, 'TRA_EXTERNO' modalidad_ingreso UNION
SELECT 4    codigo_modalidad_ingreso, 'PRIME_PUEST' modalidad_ingreso UNION
SELECT 5    codigo_modalidad_ingreso, 'DEPORTISTAS' modalidad_ingreso UNION
SELECT 6    codigo_modalidad_ingreso, 'PROFESIONAL' modalidad_ingreso UNION
SELECT 7    codigo_modalidad_ingreso, 'PRO_ESP_DIS' modalidad_ingreso UNION
SELECT 8    codigo_modalidad_ingreso, 'CICLO__CERO' modalidad_ingreso UNION
SELECT 9    codigo_modalidad_ingreso, 'SEGUNDA_OPC' modalidad_ingreso UNION
SELECT 10   codigo_modalidad_ingreso, 'CPU_CI_CERO' modalidad_ingreso UNION
SELECT 11   codigo_modalidad_ingreso, 'LARGA_DISTA' modalidad_ingreso UNION
SELECT 12   codigo_modalidad_ingreso, 'ADM_ESCOLAR' modalidad_ingreso UNION
SELECT 13   codigo_modalidad_ingreso, 'ESCOLARES_O' modalidad_ingreso

--*Sede
IF OBJECT_ID('tempdb..#sede') IS NOT NULL
    DROP TABLE #sede
```

1

```sql
SELECT 1    codigo_sede, 'HCO' sede INTO #sede UNION
SELECT 2    codigo_sede, 'TM' sede

--*Anotaciones
--sede del alumno no porque varía según el semestre(cambios de sede)
--la edad no porque conforme a los años la edad va cambiando mejor obtener el año de nacimiento nacimiento
--la información por alumnos se repite porque en combinación con el año de nacimiento se crean diferentes casos segun en semestre
-- p.e. Alumno 1: Semestre 2016: Edad 22 : Otros Datos
--      Alumno 1: Semestre 2017: Edad 23 : Otros Datos
--      Alumno 1: Semestre 2018: Edad 24 : Otros Datos

--*Alumnos
IF OBJECT_ID('tempdb..#alumno') IS NOT NULL
    DROP TABLE #alumno

select a.codigo,
       rtrim(a.alumno) alumno,
       fi.codigo_modalidad_ingreso,
       fi.modalidad_ingreso,
       case when year(a.fecha_nacimiento) between 1940 and 2005 then year(a.fecha_nacimiento) else null end anio_nacimiento,
       a.codigo_eap,
       e.desesc eap,
       a.sexo,
       case when resumen_certificado is null then 0 else 1 end resumen_certificado
  into #alumno
  from #tmp_alumnos a
inner join #forma_ingreso fi on case a.codigo_modalidad_ingreso when 13 then 12 when 10 then 8 when 5 then 1 else a.codigo_modalidad_ingreso end=fi.codigo_modalidad_ingreso --forma de ingreso parecidos(13,10), deportistas en duda-> Ex.Adm.
inner join #tmp_sede s on a.codigo_sede=s.codigo_sede
inner join #escuela e on a.codigo_eap=e.codesc
where semestre_ingreso>='2010-1' and codigo like '201[0-8]%'

--*Registro
--se eliminan valores que existen en registros de notas pero no en actas.
--se agregan ta1, ta2, emc para tener datos del semestre actual
IF OBJECT_ID('tempdb..#pre_registro') IS NOT NULL
    DROP TABLE #pre_registro

select s.codigo_sede,
       s.sede,
       r.semestre,
       r.codigo_alumno codigo,
       cast(avg(r.ta1) as numeric(5,2)) ta1,
       cast(avg(r.ta2) as numeric(5,2)) ta2,
       cast(avg(r.emc) as numeric(5,2)) emc,
       cast(avg(r.promedio) as numeric(5,2)) promedio,
       sum(c.creditos) creditos_matriculados_sem,
       sum(case when r.promedio<10.5 then 1 else 0 end) cursos_desaprobados,
       avg(c.ciclo_curso) ciclo,
       case right(r.semestre,1) when '0' then left(r.semestre,4)+'-1' when '1' then left(r.semestre,4)+'-2' when '2' then cast(cast(left(r.semestre,4) as int)+1 as varchar)+'-1' end semestre_siguiente,
       ROW_NUMBER() OVER(PARTITION BY r.codigo_alumno ORDER BY r.semestre) numero
```

2

```sql
       into #pre_registro
from #tmp_registro_electronico r
inner join #alumno a on r.codigo_alumno=a.codigo
inner join #sede s on r.codigo_sede=s.codigo_sede
inner join #tmp_cursos c on r.codigo_curso=c.codigo_curso
where r.semestre between '2010-0' and '2019-1' and exists(select ms.semestre from #tmp_matricula_semestre ms where ms.codigo_alumno=r.codigo_alumno and ms.semestre=r.semestre and ms.codigo_curso=r.codigo_curso)
group by s.codigo_sede,s.sede,r.semestre,r.codigo_alumno,a.resumen_certificado
order by s.codigo_sede,s.sede,r.codigo_alumno,r.semestre

--actualizando semestre siguiente para el caso de los que hacen verano
update pr1
set pr1.semestre_siguiente=pr2.semestre
from #pre_registro pr1
left join #pre_registro pr2 on pr1.numero+1=pr2.numero and pr1.codigo=pr2.codigo
where RIGHT(pr1.semestre, 1)='2' and LEFT(pr1.semestre_siguiente,4)+'-0'=pr2.semestre

--obteniendo desercion
IF OBJECT_ID('tempdb..#registro') IS NOT NULL
    DROP TABLE #registro

select pr1.*,
       case when pr2.codigo is null then 1 else 0 end desercion
into #registro
from #pre_registro pr1
left join #pre_registro pr2 on pr1.semestre_siguiente=pr2.semestre and pr1.codigo=pr2.codigo

--verificando aquellos que terminaron sus estudios:
-- si ya culminaron y por lo menos 9 ciclos o que sea de las siguientes modalidades
--6 PROFESIONAL
--3 TRA_EXTERNO
--ellos convalidan y terminan antes de los 10 ciclos
update r set r.desercion=0
from #registro r
inner join #alumno a on r.codigo=a.codigo
where desercion=1 and a.resumen_certificado=1 and (r.numero>=9 or a.codigo_modalidad_ingreso in (6,3))


--*CONSOLIDADO DE INFORMACIÓN
--eliminando aquellos valores que sean luego de la desercion: Si un alumno ya desertó toda información posterior es eliminado ya que no representa información relevante para este estudio
IF OBJECT_ID('tempdb..#r_borrar_posteriores') IS NOT NULL
    DROP TABLE #r_borrar_posteriores

select r2.* into #r_borrar_posteriores
from #registro r1
inner join #registro r2 on r1.semestre_siguiente<=r2.semestre and r1.codigo=r2.codigo
where r1.desercion=1
order by r1.codigo,r1.semestre,r2.semestre

-- el número de la consulta anterior debe ser menor(datos duplicados) a la cantidad de registros borrados:
```

3

```sql
delete r
from #registro r
inner join #r_borrar_posteriores rb on r.Sede=rb.Sede and r.codigo=rb.codigo and
r.semestre=rb.semestre

--actualizando ciclo
--en algunos semestres, el ciclo está con null completando con el ciclo anterior
update r1
set r1.ciclo=r2.ciclo
from #registro r1
inner join #registro r2 on r1.numero-1=r2.numero and r1.codigo=r2.codigo
where r1.ciclo is null

--*************************************************************************
--REGRESION LINEAL SIMPLE FORMULA
--autor: robvalk
--url: https://www.sqlteam.com/forums/topic.asp?TOPIC_ID=21948#51472
/*
    select 1.0 x, 9.0 y into #data
    union
    select 2.0 x, 6.0 y
    union
    select 3.0 x, 5.0 y
    union
    select 4.0 x, 5.0 y
    /**
    * m = (nSxy - SxSy)/(nSxx - SxSx)
    * b = Ay - (Ax * m)
    * N.B. S = Sum, A = Mean
    */
    DECLARE @n INT
    SELECT @n = COUNT(*) FROM #data
    SELECT (@n * SUM(X*Y) - SUM(X) * SUM(Y))/(@n * SUM(X*X) - SUM(X) * SUM(X)) AS M,
        AVG(Y) - AVG(X) *
        (@n * SUM(X*Y) - SUM(X) * SUM(Y))/(@n * SUM(X*X) - SUM(X) * SUM(X)) AS B
    FROM #data
*/
--*************************************************************************
--rev. 2 autor Jhon Frank
--@jhonfrank.ae
/*
    DECLARE @n INT
    SELECT @n = COUNT(*) FROM #data
    SELECT COALESCE( (COUNT(*) * SUM(X*Y) - SUM(X) * SUM(Y)) / NULLIF((COUNT(*) *
SUM(X*X) - SUM(X) * SUM(X)), 0) , 0) AS M,
            COALESCE( AVG(Y) - AVG(X) *
                    (COUNT(*) * SUM(X*Y) - SUM(X) * SUM(Y)) / NULLIF((COUNT(*) *
SUM(X*X) - SUM(X) * SUM(X)), 0) , 0) AS B
    FROM #data
*/
--*************************************************************************
--obteniendo algunos datos para realizar la regresion lineal simple y otros:
--#numero de cursos desaprobados anteriores al semestre actual
--#eje X (correlativo según el semestre - creciente)
--#eje Y (promedio aritmético del semestre - temporal creciente)
IF OBJECT_ID('tempdb..#historial_promedios') IS NOT NULL
    DROP TABLE #historial_promedios
```



```sql

select  r1.codigo_sede,
        r1.sede,
        r1.semestre,
        r1.codigo,
        r1.ciclo,
        r1.creditos_matriculados_sem,
        r1.numero,
        r1.ta1,
        r1.ta2,
        r1.emc,
        r1.desercion,
        ISNULL(r2.cursos_desaprobados,0) cursos_desaprobados,
        r2.numero X,
        r2.promedio Y
        into #historial_promedios
from #registro r1
left join #registro r2 on r1.numero>r2.numero and r1.codigo=r2.codigo
order by r1.codigo,r1.semestre,r2.semestre

--pre consolidado
--indicador_promedio=(nSxy - SxSy)/(nSxx - SxSx)
IF OBJECT_ID('tempdb..#pre_consolidado') IS NOT NULL
    DROP TABLE #pre_consolidado

select  hp.codigo_sede,
        hp.sede,
        hp.semestre,
        hp.codigo,
        hp.ciclo,
        hp.creditos_matriculados_sem,
        hp.numero semestres_estudiados,
        hp.ta1,
        hp.ta2,
        hp.emc,
        hp.desercion,
        SUM(hp.cursos_desaprobados) cursos_desaprobados,
        COALESCE( (COUNT(*) * SUM(X*Y) - SUM(X) * SUM(Y)) / NULLIF((COUNT(*) * SUM(X*X) -
SUM(X) * SUM(X)), 0) , 0) indicador_promedio
        into #pre_consolidado
from #historial_promedios hp
group by hp.codigo_sede,
        hp.sede,
        hp.semestre,
        hp.codigo,
        hp.ciclo,
        hp.creditos_matriculados_sem,
        hp.numero,
        hp.ta1,
        hp.ta2,
        hp.emc,
        hp.desercion
order by hp.codigo,hp.semestre

--obteniendo moras
IF OBJECT_ID('tempdb..#dias_mora') IS NOT NULL
    DROP TABLE #dias_mora
```



```sql

select codigo,semestre,round(avg(moras_dias),0) dias_mora
into #dias_mora
from (
    select b.codigo_alumno codigo,bdet.pago_semestre semestre,avg(
round(bdet.monto_detalle * 30 / ((b.monto_total-bdet.monto_detalle) * 0.02) , 0))
moras_dias--bdet.sempag,avg(bdet.candet)
    from #tmp_boletas b
    inner join #tmp_detalle_boletas bdet on b.numero=bdet.numero and b.serie=bdet.serie
and b.tipo_documento=bdet.tipo_documento
    INNER JOIN #alumno a ON b.codigo_alumno=a.codigo
    where  bdet.codigo_concepto in ('77080103') and b.monto_total-bdet.monto_detalle>0
    group by b.codigo_alumno,bdet.pago_semestre
) t
group by codigo,semestre

--consolidado
IF OBJECT_ID('tempdb..#consolidado') IS NOT NULL
    DROP TABLE #consolidado

select a.codigo,
        a.alumno,
        a.codigo_modalidad_ingreso,
        a.modalidad_ingreso,
        cast(left(pc.semestre,4) as int)-a.anio_nacimiento edad,
        a.codigo_eap,
        a.eap,
        a.sexo,
        pc.codigo_sede,
        pc.sede,
        pc.semestre,
        case RIGHT(pc.semestre,1) when '0' then 'VER' when '1' then 'ABR' when '2' then
'AGO' end tipo_semestre,
        pc.ciclo,
        pc.ta1,
        pc.ta2,
        pc.emc,
        pc.creditos_matriculados_sem,
        pc.cursos_desaprobados,
        pc.indicador_promedio,
        cast(isnull(dm.dias_mora,0) as int) dias_mora,
        pc.desercion
  into #consolidado
  from #pre_consolidado pc
inner join #alumno a on pc.codigo=a.codigo
left join #dias_mora dm on pc.semestre=dm.semestre and pc.codigo=dm.codigo

--consulta para csv
select codigo,
        --alumno,
        --codigo_modalidad_ingreso,
        modalidad_ingreso,
        edad,
        --codigo_eap,
        eap,
        sexo,
        --codigo_sede,
```

```
        sede,
        semestre,
        tipo_semestre,
        ciclo,
        ta1,
        ta2,
        emc,
        creditos_matriculados_sem,
        cursos_desaprobados,
        indicador_promedio,
        dias_mora,
        desercion
from #consolidado
where semestre<='2018-2'
```

ANEXO 06

CLASE PARA LA AUTOMATIZACIÓN DE EJECUCIÓN DE MODELOS EN PYTHON

```python
class Model:
    """
    Clase de ayuda para procesar datos con algoritmos de machine learning
    basado en sklearn enfocado de clasificación binaria.

    Attributes:
        X_train : Es el conjunto de features de entrenamiento.
        X_test : Es el conjunto de features de prueba.
        Y_train : Es el conjunto de targets de entrenamiento.
        Y_test : Es el conjunto de targets de entrenamiento.
        label : Son los nombres de los targets. Ejemplo: (nombreclase1, nombreclase2).
        sc : Usado para el preprocesamiento (normalización) de los datos.
        pca : Usado para el preprocesamiento (componentes) de los datos.
        clf : Es el mejor algoritmo de clasificación luego de la búsqueda de parametros.
    """
    X_train = []
    X_test = []
    Y_train = []
    Y_test = []
    label = []
    sc = {}
    clf = {}
    pca = {}

    def set_data(self, features, targets, target_names, n_components = 0):
        """
        Parameters:
            features(arr) : Es el conjunto de features.
            targets(arr) : Es el conjunto de targets.
            target_names(arr) : Son los nombres de los targets.
            n_components(int) : Para el análisis de componentes principales
        Returns:

        """
        import numpy as np
        from sklearn.model_selection import train_test_split
        from sklearn.utils import shuffle

        # Filas y columnas
        r, c = features.shape

        # Escogiendo datos aleatorios y dividiendo los datos de entrenamiento y prueba
        X, Y = shuffle(features, targets)
        self.label = target_names
        self.X_train, self.X_test, self.Y_train, self.Y_test = train_test_split(X, Y,
test_size = 0.20, stratify = Y)

        # Normalización de datos
        from sklearn.preprocessing import StandardScaler
        self.sc = StandardScaler()
        self.X_train = self.sc.fit_transform(self.X_train)
        self.X_test = self.sc.transform(self.X_test)

        # PCA
        from sklearn.decomposition import PCA
        n_components = c if n_components == 0 else n_components
```

1

```python
        self.pca = PCA(n_components = n_components)
        #self.X_train = self.pca.fit_transform(self.X_train)
        #self.X_test = self.pca.transform(self.X_test)

    def print_data(self):
        # Impresión de reporte
        nbr_train = self.Y_train.value_counts()
        nbr_test = self.Y_test.value_counts()
        nbr_total = self.Y_train.append(self.Y_test).value_counts()
        r, c = self.X_train.shape

        print('INFORMACIÓN DEL CONJUNTO DE DATOS:')
        print('**********************************')
        print(f'                \t  class_0    \t  class_1 \t    Total')
        print(f'  Entrenamiento \t {nbr_train[0]:8d} \t {nbr_train[1]:8d}\t
{nbr_train.sum():8d}")
        print(f"          Prueba \t {nbr_test[0]:8d} \t {nbr_test[1]:8d}\t
{nbr_test.sum():8d}")
        print(f"           Total \t {nbr_total[0]:8d} \t {nbr_total[1]:8d}\t
{nbr_total.sum():8d}")
        print('')
        print(f'       class_0: \t {self.label[0]}')
        print(f'       class_1: \t {self.label[1]}')
        print(f'      features: \t {c}')
        print()

    def find_best_params_(self, algorithm, params_dist):
        import time
        import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold

        print('Iniciar búsqueda de parametros...\n')

        # Inicialización de RandomizedSearchCV
        model = RandomizedSearchCV(
            algorithm,
            param_distributions=params_dist,
            n_iter=10,
            cv=StratifiedKFold(n_splits=4, shuffle=True),
            iid=False,
            refit=True,
            scoring='precision',
            return_train_score=True
            )

        start = time.time()
        model.fit(self.X_train, self.Y_train)
        end = time.time()


        print('RESULTADOS:')
        print('***********')

        print(f'Tiempo de ejecución: {self.diff_time_str(start, end)}')

        print(f'Modelo optimo: {model.best_estimator_}')
```

2

```python
        self.clf = model.best_estimator_

        print(f'Mejor resultado: {model.best_score_:0.3f}')

        print()

        # Resultados de la búsqueda de parámetros
        self.print_search_results(
            model.cv_results_['mean_test_score'],
            model.cv_results_['std_test_score'],
            model.cv_results_['params'],
            model.cv_results_['mean_fit_time']
        )
        # Obteniendo nuevos valores con predicción de probabilidad
        Y_pred_prob = model.predict_proba(self.X_test)

        # Asignando la clase correspondiente según la probalidad mayor
        Y_pred = np.array([ 1 if classes_[1] > classes_[0] else 0 for classes_ in
Y_pred_prob], dtype = np.int32)

        # Resultados de presicion, recall y f1-score
        self.print_report(self.Y_test, Y_pred)

        # Gráficos
        fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2, figsize=(8, 3))
        plt.subplots_adjust(right=0.5)

        # Resultados de la matriz de confusión
        self.plot_confusion_matrix(self.Y_test, Y_pred, normalize = True, ax = ax1, cmap
= 'Blues')
        # Resultados de  curva ROC y AUC
        self.plot_roc_curve(self.Y_test, Y_pred_prob[:, 1], ax = ax2)

        plt.tight_layout()
        plt.show()
        # Retornando una copia del objeto actual
        from copy import deepcopy
        return deepcopy(self)

    def _predict(self, X):
        import numpy as np

        X = self.sc.transform(X)
        #X = self.pca.transform(X)
        Y_prob = self.clf.predict_proba(X)
        Y = np.array([ 1 if classes_[1] > classes_[0] else 0 for classes_ in Y_prob],
dtype = np.int32)
        return (Y, Y_prob)

    def _export(self, filename):
        import time
        import pickle

        obj = (self.clf, self.sc, self.pca)

        start = time.time()
        with open(filename, 'wb') as fout:
```

3

```
            pickle.dump(obj,fout)
        end = time.time()

        print(f'Exportado correctamente {filename} ({self.diff_time_str(start,
end)})...')

    def _import(self, filename):
        import time
        import pickle

        start = time.time()
        with open(filename, 'rb') as fin:
            obj = pickle.load(fin)
        end = time.time()

        self.clf, self.sc, self.pca = obj
        print(f'Importado correctamente {filename} ({self.diff_time_str(start,
end)})...')
        return self

    def print_search_results(self, means, stds, params, seconds):
        print('ANALISIS DE PARAMETROS:')
        print('**********************')
        total = len(means)
        nbr_results_show = 3
        for i, (m, s, p, t) in enumerate(sorted(zip(means, stds, params, seconds),
reverse=True, key = lambda x: x[0])):
            if i < nbr_results_show:
                print("%d) %0.3f (+/-%0.3f) para %r (%0.3f seg)"%(i + 1, m, 2 * s, p, t))
        print(f'Mostrando {nbr_results_show} mejores resultados de {total}...')
        print()

    def print_report(self, y_test, y_pred):
        from sklearn.metrics import classification_report
        print('REPORTE DE CLASIFICACIÓN:')
        print('*************************')
        print(classification_report(y_test, y_pred, target_names = self.label))

    def plot_confusion_matrix(self, y_true, y_pred,
                              ax = None,
                              normalize = False,
                              title = None,
                              cmap = None):
        """
        https://scikit-
learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html
        """
        import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.metrics import classification_report, confusion_matrix
        from sklearn.utils.multiclass import unique_labels

        if ax is None:
            ax = plt.gca()

        if not title:
            if normalize:
```



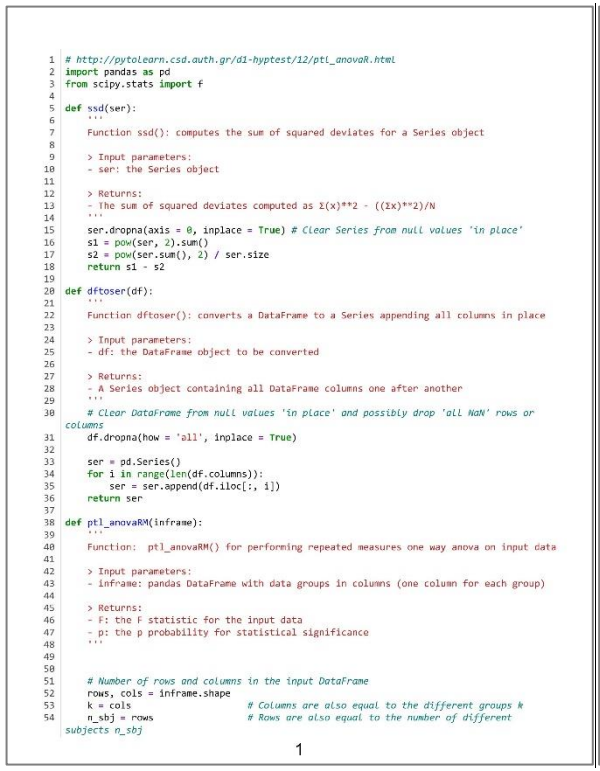
```
                title = 'MATRIZ DE CONFUSIÓN (NORMALIZADA)'
            else:
                title = 'MATRIZ DE CONFUSIÓN'

        # Compute confusion matrix
        cm = confusion_matrix(y_true, y_pred)
        # Only use the labels that appear in the data
        classes = self.label[unique_labels(y_true, y_pred)]
        if normalize:
            cm = cm.astype('float') / cm.sum(axis = 1)[:, np.newaxis]

        im = ax.imshow(cm, interpolation = 'nearest', cmap = cmap, aspect="auto")
        ax.figure.colorbar(im, ax = ax)
        # We want to show all ticks...
        ax.set(
            xticks = np.arange(cm.shape[1]),
            yticks = np.arange(cm.shape[0]),
            # ... and label them with the respective list entries
            xticklabels = classes, yticklabels = classes,
        )
        ax.set_title(title , {'fontweight' : 'bold'}, pad = 12)
        ax.set_xlabel('Valores de predicción', fontstyle = 'italic')
        ax.set_ylabel('Valores reales', fontstyle = 'italic')

        # Corrección del error matplotlib 3.1.1 heatmap: first and last y-axis
        ax.set_xticks(np.arange(cm.shape[1]+1)-.5, minor=True)
        ax.set_yticks(np.arange(cm.shape[0]+1)-.5, minor=True)

        # Loop over data dimensions and create text annotations.
        fmt = '.2f' if normalize else 'd'
        thresh = cm.max() / 2.
        for i in range(cm.shape[0]):
            for j in range(cm.shape[1]):
                ax.text(j, i, format(cm[i, j], fmt),
                        ha = "center", va = "center",
                        color = "white" if cm[i, j] > thresh else "black")
        return ax

    def plot_roc_curve(self, y_true, y_pred_score, ax = None):
        from sklearn.metrics import roc_curve, auc
        import matplotlib.pyplot as plt


        if ax is None:
            ax = plt.gca()

        fpr, tpr, threshold = roc_curve(y_true, y_pred_score)
        roc_auc = auc(fpr, tpr)
        ax.plot(fpr, tpr, color='steelblue', lw = 2, label='ROC (AUC: %0.2f)' % roc_auc)
        ax.plot([0, 1], [0, 1], color = 'navy', linestyle = '--')


        ax.set(
            xlim = [-0.02, 1.02],
            ylim = [-0.02, 1.02]
        )
        ax.set_xlabel('Falsos positivos', fontstyle = 'italic')
```



```
        ax.set_ylabel('Verdaderos positivos', fontstyle = 'italic')
        ax.set_title('CURVA   R O C', {'fontweight' : 'bold'}, pad = 12)
        ax.legend(loc = 'lower right')
        return ax

    def get_metrics(self, y_true, y_pred, y_pred_prob):
        """
        Returns:
            dict(
                precisiion,
                recall,
                f1_score,
                auc
            )
        """
        from sklearn.metrics import precision_score, recall_score, f1_score,
roc_auc_score
        precision = precision_score(y_true, y_pred, average = 'macro')
        recall = recall_score(y_true, y_pred, average = 'macro')
        f1_score = f1_score(y_true, y_pred, average = 'macro')
        auc = roc_auc_score(y_true, y_pred_prob[:, 1])
        return ( precision, recall, f1_score, auc)

    def diff_time_str(self, start, end):
        """
        https://www.w3resource.com/python-exercises/date-time-exercise/python-date-time-
exercise-37.php
        """
        diff = end - start
        minutes, seconds = divmod(diff, 60)
        hours, minutes = divmod(minutes, 60)
        days, hours = divmod(hours, 24)
        time_str = (
                    (f'{int(days)} día(s) ' if days > 0 else '') +
                    (f'{int(hours)} hora(s) ' if hours > 0 else '') +
                    (f'{int(minutes)} min ' if minutes > 0 else '') +
                    (f'{seconds:0.3f} seg' if seconds >= 0 else '')
                   )
        return time_str
```

ANEXO 07

IMPLEMENTACIÓN DE PRUEBA ANOVA DE MEDIDAS REPETIDAS EN PYTHON



```python
# http://pytolearn.csd.auth.gr/d1-hyptest/12/ptl_anovaR.html
import pandas as pd
from scipy.stats import f

def ssd(ser):
    '''
    Function ssd(): computes the sum of squared deviates for a Series object

    > Input parameters:
    - ser: the Series object

    > Returns:
    - The sum of squared deviates computed as Σ(x)**2 - ((Σx)**2)/N
    '''
    ser.dropna(axis = 0, inplace = True) # Clear Series from null values 'in place'
    s1 = pow(ser, 2).sum()
    s2 = pow(ser.sum(), 2) / ser.size
    return s1 - s2

def dftoser(df):
    '''
    Function dftoser(): converts a DataFrame to a Series appending all columns in place

    > Input parameters:
    - df: the DataFrame object to be converted

    > Returns:
    - A Series object containing all DataFrame columns one after another
    '''
    # Clear DataFrame from null values 'in place' and possibly drop 'all NaN' rows or columns
    df.dropna(how = 'all', inplace = True)

    ser = pd.Series()
    for i in range(len(df.columns)):
        ser = ser.append(df.iloc[:, i])
    return ser

def ptl_anovaRM(inframe):
    '''
    Function:  ptl_anovaRM() for performing repeated measures one way anova on input data

    > Input parameters:
    - inframe: pandas DataFrame with data groups in columns (one column for each group)

    > Returns:
    - F: the F statistic for the input data
    - p: the p probability for statistical significance
    '''


    # Number of rows and columns in the input DataFrame
    rows, cols = inframe.shape
    k = cols                    # Columns are also equal to the different groups k
    n_sbj = rows                # Rows are also equal to the number of different subjects n_sbj
```

1

```python

    # Convert dataframe to a series object
    allser = dftoser(inframe)
    n_t = allser.size           # n_t is the total number of data (measurements)


    # Compute ss_t (sum of squared deviates for the whole data set)
    ss_t = ssd(allser)


    # Compute ss_wg as sum of ss for all sample groups ( = columns in the 'inframe' DataFrame)
    ss_wg = 0
    for i in range(k):
        ss_wg += ssd(inframe.iloc[:, i])

    # OK Compute ss_bg by subtracking ss_wg from ss_t
    ss_bg = ss_t - ss_wg

    # Construct a new Series object: "subjectmeans" (contains the means for each subject data)
    subjmeans = pd.Series([0 for i in range(rows)])
    for i in range(rows):
        sm = inframe.iloc[i, :].mean(skipna = True)
        subjmeans.iloc[i] = sm

    # Compute ss_sb as the weighed (by groups) ssd for the subjmeans object
    ss_sb = k*ssd(subjmeans)
    ss_er = ss_wg - ss_sb

    # degrees of freedom
    df_t = n_t - 1
    df_bg = k - 1
    df_wg = n_t - k
    df_sb = n_sbj - 1
    df_er = df_wg - df_sb

    ms_bg = ss_bg / df_bg
    ms_er = ss_er / df_er

    F = ms_bg / ms_er
    p = f.sf(F, df_bg, df_er, loc = 0, scale = 1)

    # Printouts
    print('Trat.(Between groups):  SS_bg = {:8.4f},  df_bg = {:4d},  MS_bg = {:8.4f}'.format(ss_bg, df_bg, ms_bg))
    print('      (Within groups):  SS_wg = {:8.4f},  df_wg = {:4d}'.format(ss_wg, df_wg))
    print('     Residual(Error):  SS_er = {:8.4f},  df_er = {:4d},  MS_er = {:8.4f}'.format(ss_er, df_er, ms_er))
    print('     Sujetos(Subject):  SS_sb = {:8.4f},  df_sb = {:4d}'.format(ss_sb, df_sb))
    print('                TOTAL:  SS_t  = {:8.4f},   df_t = {:4d}'.format(ss_t, df_t))
    print()
    print('F = {:8.4f}, p = {:8.4f}'.format(F, p))

    return F, p
```

2

ANEXO 08

ETAPA 1 - DATOS

1. DATOS

Recolección de datos

Se realizó consultas SQL para obtener la información: csv/data.csv

Exploración de datos

In [1]:
```
import pandas as pd
data = pd.read_csv('csv/data.csv', sep=',', header = 0, encoding='latin1', dty
pe = {'codigo': str})
```

In [2]:
```
data.columns
```

Out[2]:
```
Index(['codigo', 'modalidad_ingreso', 'edad', 'eap', 'sexo', 'sede',
       'semestre', 'tipo_semestre', 'ciclo', 'ta1', 'ta2', 'emc',
       'creditos_matriculados_sem', 'cursos_desaprobados',
       'indicador_promedio', 'dias_mora', 'desercion'],
      dtype='object')
```

In [3]:
```
data.describe()
```

Out[3]:

	edad	ciclo	ta1	ta2	emc	creditos_mat
count	110155.000000	127332.000000	127332.000000	127332.000000	127332.000000	
mean	21.630539	4.052038	10.720994	10.845188	9.709383	
std	4.878329	2.500743	3.103882	3.221449	3.388315	
min	14.000000	1.000000	0.000000	0.000000	0.000000	
25%	19.000000	2.000000	9.170000	9.430000	8.000000	
50%	20.000000	3.500000	11.210000	11.500000	10.170000	
75%	23.000000	5.500000	12.830000	13.000000	12.000000	
max	73.000000	14.000000	20.000000	20.000000	20.000000	

In [4]:
```
data.desercion.value_counts()
```

Out[4]:
```
0    109888
1     17444
Name: desercion, dtype: int64
```

In [5]:
```
data.desercion.value_counts().sum()
```

Out[5]:
```
127332
```

Completando datos faltantes

In [6]:
```
data.columns[data.isnull().any()]
```

Out[6]:
```
Index(['edad'], dtype='object')
```

In [7]:
```
data['edad'].describe()
```

Out[7]:
```
count    110155.000000
mean         21.630539
std           4.878329
min          14.000000
25%          19.000000
50%          20.000000
75%          23.000000
max          73.000000
Name: edad, dtype: float64
```

Se va a reemplazar la columna edad siguiendo este orden:

1. Con aleatorios según distribución normal, agrupados por modalidad de ingreso, semestre y sexo.
2. Con la media de datos agrupados por modalidad de ingreso, semestre y sexo.
3. Con la media de todos los datos.
4. Con la media de todos los datos para aquellos cuya edad sea menor a 16.

In [8]:
```
import numpy as np

# Valor del RANDOM STATE
np.random.seed(1998)

#1
data['edad'] = data.groupby(['modalidad_ingreso', 'semestre', 'sexo'])['edad']
.transform(
    lambda x: x.fillna( x.mean(skipna = True) + x.std(skipna=True) * np.random
.randn() )
)
#2
data['edad'] = data.groupby(['modalidad_ingreso', 'semestre', 'sexo'])['edad']
.transform(
    lambda x: x.fillna( x.mean(skipna = True) )
)
#3
data['edad'] = data['edad'].transform(lambda x: x.fillna( x.mean(skipna=True)
))
#4
data['edad'] = np.where(data['edad'] < 16, data['edad'].mean(), data.edad)
```

In [9]:
```
data.columns[data.isnull().any()]
```

Out[9]:
```
Index([], dtype='object')
```

Codificación de variables categóricas

In [10]:
```
def createDummies(df, var_name, prefix = None):

    if prefix == None:
        prefix = var_name

    # Codificando
    dummy = pd.get_dummies(df[var_name], prefix = prefix)

    # Insertando en la misma posicion
    c_index = df.columns.get_loc(var_name)

    for column in dummy:
        df.insert(c_index, column,dummy[column])

    # Eliminando la columna inicial
    df = df.drop(var_name, axis = 1)

    return df
```

In [11]:
```
data = createDummies(data, 'modalidad_ingreso','mod_ing')
data = createDummies(data, 'eap')
data = createDummies(data, 'sexo')
data = createDummies(data, 'sede')
data = createDummies(data, 'tipo_semestre', 'tip_sem')
```

Nuevas columnas creadas:

In [12]:
```
data.columns
```

Out[12]:
```
Index(['codigo', 'mod_ing_TRA_INTERNO', 'mod_ing_TRA_EXTERNO',
       'mod_ing_PROFESIONAL', 'mod_ing_PRIME_PUEST', 'mod_ing_EX_ADMISION',
       'mod_ing_CICLO__CERO', 'mod_ing_ADM_ESCOLAR', 'edad', 'eap_TUR',
       'eap_SIS', 'eap_PST', 'eap_ODO', 'eap_OBS', 'eap_MKT', 'eap_ENF',
       'eap_EDU', 'eap_EDI', 'eap_DER', 'eap_CON', 'eap_CIV', 'eap_ARQ',
       'eap_AMB', 'eap_ADM', 'sexo_M', 'sexo_F', 'sede_TM', 'sede_HCO',
       'semestre', 'tip_sem_VER', 'tip_sem_AGO', 'tip_sem_ABR', 'ciclo', 'ta
1',
       'ta2', 'emc', 'creditos_matriculados_sem', 'cursos_desaprobados',
       'indicador_promedio', 'dias_mora', 'desercion'],
      dtype='object')
```

Eliminando datos no necesarios

In [13]:
```
data = data.drop(["semestre"], axis = 1)
data = data.drop(["codigo"], axis = 1)
```

Exportando

Los datos se van a exportar en tres conjuntos:

- **poblacion** : Representa al total de datos corregidos (127332 instancias).
- **dataset** : Conjunto de datos que servirá para obtener los modelos algorítmicos (14800 instancias).
- **muestra** : Conjunto de datos que servirá para validar la hipótesis (14800 instancias).

In [14]:
```
data.to_csv(r'csv/poblacion.csv', index = None, header=True)
```


In [15]:
```
import numpy as np
import pandas as pd
from sklearn.utils import resample

# Valor del RANDOM STATE
np.random.seed(1998)

# Número total de instancias para cada conjunto
nbr = 14800

# División por clases
df_majority = data[data.desercion == 1]
df_minority = data[data.desercion == 0]

# Obteniendo los subgrupos
df_majority_sampled = resample(df_majority,
                               replace = False,
                               n_samples = nbr)

df_minority_sampled = resample(df_minority,
                               replace = False,
                               n_samples = nbr)

# Combinando los subgrupos de clases obtenidas
dataset = pd.concat(
    [
        df_minority_sampled.iloc[:nbr//2, :],
        df_majority_sampled.iloc[:nbr//2, :]
    ]
)

muestra = pd.concat(
    [
        df_minority_sampled.iloc[nbr//2:, :],
        df_majority_sampled.iloc[nbr//2:, :]
    ]
)
```

In [16]:
```
dataset.to_csv(r'csv/dataset.csv', index = None, header=True)
```

In [17]:
```
muestra.to_csv(r'csv/muestra.csv', index = None, header=True)
```

ANEXO 09

ETAPA 2 - SELECCIÓN DE ATRIBUTOS

2. SELECCIÓN DE ATRIBUTOS

Importando datos

In [1]:
```
import numpy as np
import pandas as pd

poblacion = pd.read_csv('csv/poblacion.csv', sep = ',', header = 0, encoding =
'latin1')

X = poblacion.drop('desercion', 1)
Y = poblacion['desercion']
```

Matriz de correlación

In [2]:
```
corrmat = poblacion.corr()
```

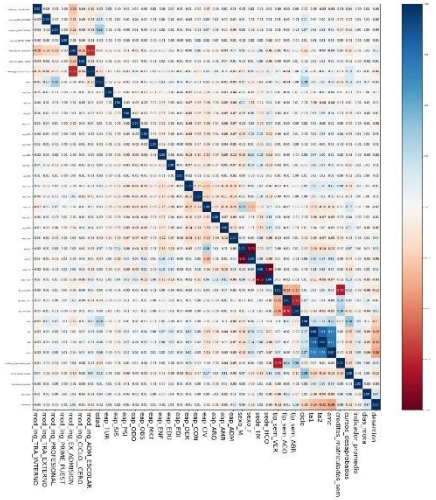
In [3]:
```
corrmat.values
```

Out[3]:
```
array([[ 1.        , -0.02976375, -0.02909876, ..., -0.00341424,
         0.00659988, -0.00427657],
       [-0.02976375,  1.        , -0.02158063, ...,  0.01118786,
         0.00820003, -0.0158287 ],
       [-0.02909876, -0.02158063,  1.        , ...,  0.00402859,
         0.00119131, -0.00360254],
       ...,
       [-0.00341424,  0.01118786,  0.00402859, ...,  1.        ,
        -0.02792218, -0.0299246 ],
       [ 0.00659988,  0.00820003,  0.00119131, ..., -0.02792218,
         1.        ,  0.14432563],
       [-0.00427657, -0.0158287 , -0.00360254, ..., -0.0299246 ,
         0.14432563,  1.        ]])
```

1

In [4]:
```
# Mapa de calor

%matplotlib inline
import matplotlib.pyplot as plt

# Tamaño de Letra
plt.rc('font', size = 9)          # controls default text sizes
plt.rc('axes', titlesize = 9)     # fontsize of the axes title
plt.rc('axes', labelsize = 10)    # fontsize of the x and y labels
plt.rc('xtick', labelsize = 20)    # fontsize of the tick labels
plt.rc('ytick', labelsize = 6)    # fontsize of the tick labels
plt.rc('legend', fontsize = 9)    # legend fontsize
plt.rc('figure', titlesize = 10)  # fontsize of the figure title

fig, ax = plt.subplots(figsize=(25, 24))
im = ax.imshow(corrmat, interpolation = 'nearest', cmap = 'RdBu', aspect = "au
to")
ax.figure.colorbar(im, ax = ax)

ax.set(
    xticks = np.arange(corrmat.shape[1]),
    yticks = np.arange(corrmat.shape[0]),
    xticklabels = corrmat.index,
    yticklabels = corrmat.index,
)

plt.setp(ax.get_xticklabels(), rotation=-90, ha = "right", rotation_mode = "de
fault")

# Corrección del error matplotlib 3.1.1 heatmap: first and last y-axis
ax.set_xticks(np.arange(corrmat.shape[1]+1)-.5, minor = True)
ax.set_yticks(np.arange(corrmat.shape[0]+1)-.5, minor = True)

# Formato a 2 decimales
fmt = '.2f'
thresh = corrmat.max().max() / 2.

for i in range(corrmat.shape[0]):
    for j in range(corrmat.shape[1]):
        ax.text(j, i, format(corrmat.iat[i, j], fmt), ha = "center", va = "cen
ter",
                color = "white" if abs(corrmat.iat[i, j]) > thresh else "blac
k")

plt.show()
```

2



3

In [5]:
```
# Lista de atributos que presentan mayor correlación con la columna desercion
corrmat['desercion'].iloc[(-np.abs(corrmat.iloc[38].values)).argsort()]
```

Out[5]:
```
desercion                   1.000000
emc                        -0.381613
ta2                        -0.322048
ta1                        -0.285103
ciclo                      -0.205484
dias_mora                   0.144326
tip_sem_VER                -0.122638
cursos_desaprobados        -0.118732
tip_sem_ABR                 0.097573
creditos_matriculados_sem   0.081313
mod_ing_EX_ADMISION         0.062712
mod_ing_ADM_ESCOLAR        -0.056245
sexo_M                      0.055187
sexo_F                     -0.055187
eap_ADM                     0.052561
indicador_promedio         -0.029925
eap_TUR                     0.028147
sede_HCO                   -0.025021
sede_TM                     0.025021
eap_AMB                    -0.024060
eap_SIS                     0.023404
edad                       -0.021373
eap_ODO                    -0.020608
eap_CIV                    -0.020595
mod_ing_TRA_EXTERNO        -0.015829
eap_ENF                    -0.015647
mod_ing_CICLO__CERO        -0.014735
eap_DER                    -0.013898
tip_sem_AGO                -0.013322
eap_MKT                     0.012197
eap_ARQ                    -0.011973
eap_EDU                     0.011654
eap_OBS                     0.009317
eap_EDI                     0.007073
mod_ing_PRIME_PUEST        -0.006723
eap_CON                     0.006030
eap_PSI                    -0.005469
mod_ing_TRA_INTERNO        -0.004277
mod_ing_PROFESIONAL        -0.003603
Name: desercion, dtype: float64
```


Selección

La selección de los atributos se hará de acuerdo a las siguientes técnicas:

1. Pearson Correlation
2. Chi-Squared
3. Recursive Feature Elimination
4. Lasso: SelectFromModel
5. Tree-based: SelectFromModel
6. LightGBM: SelectFromModel

In [6]:
```
"""
https://towardsdatascience.com/the-5-feature-selection-algorithms-every-data-scientist-need-to-know-3a6b566efd2
"""
# Número de atributos
num_feats=30
```

In [7]:
```
# 1) Pearson Correlation

def cor_selector(X, y,num_feats):
    cor_list = []
    feature_name = X.columns.tolist()
    # calculate the correlation with y for each feature
    for i in X.columns.tolist():
        cor = np.corrcoef(X[i], Y)[0, 1]
        cor_list.append(cor)
    # replace NaN with 0
    cor_list = [0 if np.isnan(i) else i for i in cor_list]
    # feature name
    cor_feature = X.iloc[:,np.argsort(np.abs(cor_list))[-num_feats:]].columns.tolist()
    # feature selection? 0 for not select, 1 for select
    cor_support = [True if i in cor_feature else False for i in feature_name]
    return cor_support, cor_feature
cor_support, cor_feature = cor_selector(X, Y,num_feats)
print(str(len(cor_feature)), 'selected features')
print(cor_feature)
```

```
30 selected features
['eap_ARQ', 'eap_MKT', 'tip_sem_AGO', 'eap_DER', 'mod_ing_CICLO__CERO', 'eap_ENF', 'mod_ing_TRA_EXTERNO', 'eap_CIV', 'eap_ODO', 'edad', 'eap_SIS', 'eap_AMB', 'sede_TM', 'sede_HCO', 'eap_TUR', 'indicador_promedio', 'eap_ADM', 'sexo_M', 'sexo_F', 'mod_ing_ADM_ESCOLAR', 'mod_ing_EX_ADMISION', 'creditos_matriculados_sem', 'tip_sem_ABR', 'cursos_desaprobados', 'tip_sem_VER', 'dias_mora', 'ciclo', 'ta1', 'ta2', 'emc']
```


In [8]:
```
# 2) Chi-Squared

# Valor del RANDOM STATE
np.random.seed(1998)

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.preprocessing import MinMaxScaler
X_norm = MinMaxScaler().fit_transform(X)
chi_selector = SelectKBest(chi2, k=num_feats)
chi_selector.fit(X_norm, Y)
chi_support = chi_selector.get_support()
chi_feature = X.loc[:,chi_support].columns.tolist()
print(str(len(chi_feature)), 'selected features')
print(chi_feature)
```

```
30 selected features
['mod_ing_TRA_EXTERNO', 'mod_ing_EX_ADMISION', 'mod_ing_CICLO__CERO', 'mod_ing_ADM_ESCOLAR', 'eap_TUR', 'eap_SIS', 'eap_ODO', 'eap_OBS', 'eap_MKT', 'eap_ENF', 'eap_EDU', 'eap_DER', 'eap_CIV', 'eap_ARQ', 'eap_AMB', 'eap_ADM', 'sexo_M', 'sexo_F', 'sede_TM', 'sede_HCO', 'tip_sem_VER', 'tip_sem_AGO', 'tip_sem_ABR', 'ciclo', 'ta1', 'ta2', 'emc', 'creditos_matriculados_sem', 'cursos_desaprobados', 'dias_mora']
```

In [9]:
```
# 3) Recursive Feature Elimination

# Valor del RANDOM STATE
np.random.seed(1998)

from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
rfe_selector = RFE(estimator=LogisticRegression(solver = 'liblinear'), n_features_to_select=num_feats, step=10, verbose=5)
rfe_selector.fit(X_norm, Y)
rfe_support = rfe_selector.get_support()
rfe_feature = X.loc[:,rfe_support].columns.tolist()
print(str(len(rfe_feature)), 'selected features')
print(rfe_feature)
```

```
Fitting estimator with 38 features.
30 selected features
['mod_ing_TRA_INTERNO', 'mod_ing_TRA_EXTERNO', 'mod_ing_PROFESIONAL', 'mod_ing_PRIME_PUEST', 'mod_ing_EX_ADMISION', 'mod_ing_ADM_ESCOLAR', 'edad', 'eap_TUR', 'eap_ODO', 'eap_MKT', 'eap_EDU', 'eap_EDI', 'eap_CIV', 'eap_ARQ', 'eap_AMB', 'sexo_M', 'sexo_F', 'sede_TM', 'sede_HCO', 'tip_sem_VER', 'tip_sem_AGO', 'tip_sem_ABR', 'ciclo', 'ta1', 'ta2', 'emc', 'creditos_matriculados_sem', 'cursos_desaprobados', 'indicador_promedio', 'dias_mora']
```

In [10]:
```
# 4) Lasso: SelectFromModel

# Valor del RANDOM STATE
np.random.seed(1998)

from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LogisticRegression
embeded_lr_selector = SelectFromModel(LogisticRegression(penalty="l1",solver =
'liblinear'), max_features=num_feats)#
embeded_lr_selector.fit(X_norm, Y)
embeded_lr_support = embeded_lr_selector.get_support()
embeded_lr_feature = X.loc[:,embeded_lr_support].columns.tolist()
print(str(len(embeded_lr_feature)), 'selected features')
print(embeded_lr_feature)
```

```
30 selected features
['mod_ing_TRA_INTERNO', 'mod_ing_TRA_EXTERNO', 'mod_ing_PROFESIONAL', 'mod_in
g_EX_ADMISION', 'mod_ing_ADM_ESCOLAR', 'edad', 'eap_TUR', 'eap_SIS', 'eap_OD
O', 'eap_OBS', 'eap_MKT', 'eap_EDU', 'eap_EDI', 'eap_CIV', 'eap_ARQ', 'eap_AM
B', 'sexo_M', 'sexo_F', 'sede_HCO', 'tip_sem_VER', 'tip_sem_AGO', 'tip_sem_AB
R', 'ciclo', 'ta1', 'ta2', 'emc', 'creditos_matriculados_sem', 'cursos_desapr
obados', 'indicador_promedio', 'dias_mora']
```

In [11]:
```
# 5) Tree-based: SelectFromModel

# Valor del RANDOM STATE
np.random.seed(1998)

from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier
embeded_rf_selector = SelectFromModel(RandomForestClassifier(n_estimators=100
), max_features=num_feats)
embeded_rf_selector.fit(X, Y)
embeded_rf_support = embeded_rf_selector.get_support()
embeded_rf_feature = X.loc[:,embeded_rf_support].columns.tolist()
print(str(len(embeded_rf_feature)), 'selected features')
print(embeded_rf_feature)
```

```
9 selected features
['edad', 'ciclo', 'ta1', 'ta2', 'emc', 'creditos_matriculados_sem', 'cursos_d
esaprobados', 'indicador_promedio', 'dias_mora']
```


In [12]:
```
# 6) LightGBM: SelectFromModel

# Valor del RANDOM STATE
np.random.seed(1998)

from sklearn.feature_selection import SelectFromModel
from lightgbm import LGBMClassifier
lgbc=LGBMClassifier(n_estimators=500, learning_rate=0.05, num_leaves=32, colsa
mple_bytree=0.2,reg_alpha=3, reg_lambda=1, min_split_gain=0.01, min_child_weig
ht=40)
embeded_lgb_selector = SelectFromModel(lgbc, max_features=num_feats)
embeded_lgb_selector.fit(X, Y)
embeded_lgb_support = embeded_lgb_selector.get_support()
embeded_lgb_feature = X.loc[:,embeded_lgb_support].columns.tolist()
print(str(len(embeded_lgb_feature)), 'selected features')
print(embeded_lgb_feature)
```

```
9 selected features
['edad', 'ciclo', 'ta1', 'ta2', 'emc', 'creditos_matriculados_sem', 'cursos_d
esaprobados', 'indicador_promedio', 'dias_mora']
```


In [13]:
```
# put all selection together
feature_name = X.columns.tolist()
feature_selection_df = pd.DataFrame(
    {
        'Feature':feature_name,
        'Pearson':cor_support,
        'Chi-2':chi_support,
        'RFE':rfe_support,
        'Log':embeded_lr_support,
        'RForest':embeded_rf_support,
        'LGBM':embeded_lgb_support
    }
)
# count the selected times for each feature
feature_selection_df['Total'] = np.sum(feature_selection_df, axis=1)
# display the top num_feats
feature_selection_df = feature_selection_df.sort_values(['Total','Feature'] ,
ascending=False)
feature_selection_df.index = range(1, len(feature_selection_df)+1)
feature_selection_df.head(num_feats)
```

Out[13]:

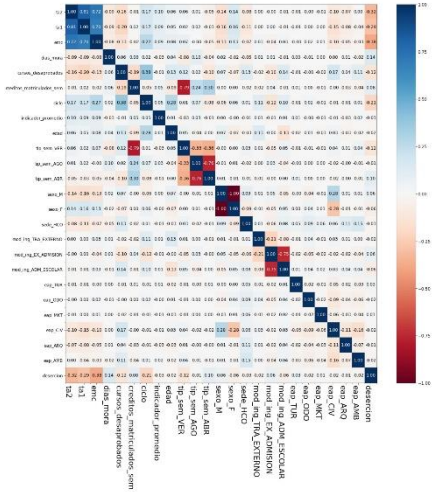
	Feature	Pearson	Chi-2	RFE	Log	RForest	LGBM	Total
1	ta2	True	True	True	True	True	True	6
2	ta1	True	True	True	True	True	True	6
3	emc	True	True	True	True	True	True	6
4	dias_mora	True	True	True	True	True	True	6
5	cursos_desaprobados	True	True	True	True	True	True	6
6	creditos_matriculados_sem	True	True	True	True	True	True	6
7	ciclo	True	True	True	True	True	True	6
8	indicador_promedio	True	False	True	True	True	True	5
9	edad	True	False	True	True	True	True	5
10	tip_sem_VER	True	True	True	True	False	False	4
11	tip_sem_AGO	True	True	True	True	False	False	4
12	tip_sem_ABR	True	True	True	True	False	False	4
13	sexo_M	True	True	True	True	False	False	4
14	sexo_F	True	True	True	True	False	False	4
15	sede_HCO	True	True	True	True	False	False	4
16	mod_ing_TRA_EXTERNO	True	True	True	True	False	False	4
17	mod_ing_EX_ADMISION	True	True	True	True	False	False	4
18	mod_ing_ADM_ESCOLAR	True	True	True	True	False	False	4
19	eap_TUR	True	True	True	True	False	False	4
20	eap_ODO	True	True	True	True	False	False	4
21	eap_MKT	True	True	True	True	False	False	4
22	eap_CIV	True	True	True	True	False	False	4
23	eap_ARQ	True	True	True	True	False	False	4
24	eap_AMB	True	True	True	True	False	False	4
25	sede_TM	True	True	True	False	False	False	3
26	eap_SIS	True	True	False	True	False	False	3
27	eap_EDU	False	True	True	True	False	False	3
28	mod_ing_TRA_INTERNO	False	False	True	True	False	False	2
29	mod_ing_PROFESIONAL	False	False	True	True	False	False	2
30	mod_ing_CICLO__CERO	True	True	False	False	False	False	2

In [14]:
```
# Seleccionando aquellas que tengan un puntaje mayor o igual a 4
best_features = feature_selection_df[feature_selection_df.Total >= 4][['Feature']]
# Agregar deserción para tener el lista completa de atributos a usar
best_features = best_features.append(pd.Series({'Feature': 'desercion'}), ignore_index=True)
# Lista de mejores atributos
best_features
```

Out[14]:

	Feature
0	ta2
1	ta1
2	emc
3	dias_mora
4	cursos_desaprobados
5	creditos_matriculados_sem
6	ciclo
7	indicador_promedio
8	edad
9	tip_sem_VER
10	tip_sem_AGO
11	tip_sem_ABR
12	sexo_M
13	sexo_F
14	sede_HCO
15	mod_ing_TRA_EXTERNO
16	mod_ing_EX_ADMISION
17	mod_ing_ADM_ESCOLAR
18	eap_TUR
19	eap_ODO
20	eap_MKT
21	eap_CIV
22	eap_ARQ
23	eap_AMB
24	desercion

In [15]:
```
# Seleccionando atributos importantes
corrmat_best_features = poblacion.loc[:, best_features.Feature].corr()

# Mapa de calor

%matplotlib inline
import matplotlib.pyplot as plt

# Tamaño de letra
plt.rc('font', size = 12)          # controls default text sizes
plt.rc('axes', titlesize = 12)     # fontsize of the axes title
plt.rc('axes', labelsize = 16)    # fontsize of the x and y labels
plt.rc('xtick', labelsize = 22)    # fontsize of the tick labels
plt.rc('ytick', labelsize = 12)    # fontsize of the tick labels
plt.rc('legend', fontsize = 12)    # legend fontsize
plt.rc('figure', titlesize = 16)  # fontsize of the figure title

fig, ax = plt.subplots(figsize = (20, 20))
im = ax.imshow(corrmat_best_features, interpolation = 'nearest', cmap = 'RdBu'
, aspect = "auto")
ax.figure.colorbar(im, ax = ax)

ax.set(
    xticks = np.arange(corrmat_best_features.shape[1]),
    yticks = np.arange(corrmat_best_features.shape[0]),
    xticklabels = corrmat_best_features.index,
    yticklabels = corrmat_best_features.index,
)

plt.setp(ax.get_xticklabels(), rotation = -90, ha = "right", rotation_mode =
"default")

# Corrección del error matplotlib 3.1.1 heatmap: first and last y-axis
ax.set_xticks(np.arange(corrmat_best_features.shape[1]+1)-.5, minor = True)
ax.set_yticks(np.arange(corrmat_best_features.shape[0]+1)-.5, minor = True)

# Formato a 2 decimales
fmt = '.2f'
thresh = corrmat_best_features.max().max() / 2.

for i in range(corrmat_best_features.shape[0]):
    for j in range(corrmat_best_features.shape[1]):
        ax.text(j, i, format(corrmat_best_features.iat[i, j], fmt), ha = "cent
er", va = "center",
                color = "white" if abs(corrmat_best_features.iat[i, j]) > thre
sh else "black")

plt.show()
```



Exportando

```
In [16]: best_features.to_csv(r'csv/best_features.csv', index = None, header=True)
```

ANEXO 10

ETAPA 3 - MODELOS

3. Modelos

Importando datos y ejecutando los modelos

In [1]:
```
%matplotlib inline
import warnings
import numpy as np
import pandas as pd
from scipy import stats
from sklearn.exceptions import ConvergenceWarning
from lib.Model import Model

# No mostrar los avisos de Error de Convergencia, para algoritmos que usen ite
raciones
warnings.filterwarnings("ignore", category = ConvergenceWarning)

# Importando datos
data = pd.read_csv('csv/dataset.csv', sep=',', header=0, encoding='latin1')

# Seleccionando atributos importantes
best_features = pd.read_csv('csv/best_features.csv', sep = ',', header = 0, en
coding = 'latin1')
data = data.loc[:, best_features.Feature]

# Dividiendo:    X(features)    Y(targets)    etiquetas(Label)
X = data.drop('desercion', 1)
Y = data['desercion']
label = np.array(['No Deserción', 'Deserción'])
```

In [2]:
```
# Valor del RANDOM STATE
np.random.seed(1998)

# Generando
model = Model()
model.set_data(X, Y, label)
model.print_data()
```

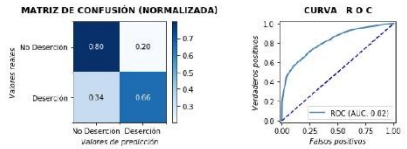
```
INFORMACIÓN DEL CONJUNTO DE DATOS:
**********************************
                        class_0        class_1          Total
  Entrenamiento            5920           5920          11840
         Prueba            1480           1480           2960
          Total            7400           7400          14800

       class_0:        No Deserción
       class_1:        Deserción
      features:        24
```

1

In [3]:
```
# 1) K-nearest neighbors
from sklearn.neighbors import KNeighborsClassifier

# Valor del RANDOM STATE
np.random.seed(1998)

params_dist = {
    'n_neighbors': stats.randint(10, 100),
    'weights': ['uniform', 'distance'],
    'metric': ['minkowski', 'euclidean', 'manhattan'],
    'algorithm': ['ball_tree', 'kd_tree', 'brute']
}

knn_model = model.find_best_params_(KNeighborsClassifier(), params_dist)
```

2

```
Iniciar búsqueda de parametros...

RESULTADOS:
***********
Tiempo de ejecución: 2 min 29.299 seg
Modelo optimo: KNeighborsClassifier(algorithm='kd_tree', leaf_size=30, metric
='euclidean',
                     metric_params=None, n_jobs=None, n_neighbors=44, p=2,
                     weights='uniform')
Mejor resultado: 0.769

ANALISIS DE PARAMETROS:
***********************
1) 0.769 (+/-0.018) para {'algorithm': 'kd_tree', 'metric': 'euclidean', 'n_n
eighbors': 44, 'weights': 'uniform'} (0.031 seg)
2) 0.766 (+/-0.020) para {'algorithm': 'ball_tree', 'metric': 'minkowski', 'n
_neighbors': 18, 'weights': 'distance'} (0.032 seg)
3) 0.764 (+/-0.014) para {'algorithm': 'brute', 'metric': 'euclidean', 'n_nei
ghbors': 26, 'weights': 'distance'} (0.006 seg)
Mostrando 3 mejores resultados de 10...

REPORTE DE CLASIFICACIÓN:
*************************
              precision    recall  f1-score   support

No Deserción       0.70      0.80      0.75      1480
   Deserción       0.77      0.66      0.71      1480

    accuracy                           0.73      2960
   macro avg       0.73      0.73      0.73      2960
weighted avg       0.73      0.73      0.73      2960
```

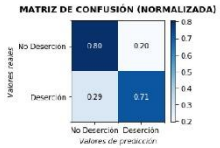


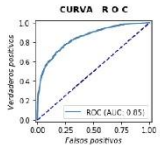
3

In [4]:
```
# 2) Support vector machines
from sklearn import svm

# Valor del RANDOM STATE
np.random.seed(1998)

params_dist = {
    'kernel': ['rbf'],
    'C': stats.uniform(1, 100),
    'gamma': stats.uniform(1e-5, 1e-2),
    #'degree': np.arange( 0, 100+0, 1 ).tolist(), #only kernel = poly
    #'coef0': np.arange( 0.0, 10.0+0.0, 0.1 ).tolist(), #only kernel = (poly, sigmoid)
    #'tol': stats.uniform(1e-30, 1e-2),
    #'shrinking': [False]
}

svm_model = model.find_best_params_(svm.SVC(decision_function_shape='ovr', probability=True), params_dist)
```



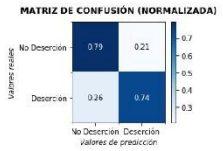
```
Iniciar búsqueda de parametros...

RESULTADOS:
**********
Tiempo de ejecución: 12 min 46.757 seg
Modelo optimo: SVC(C=37.51376470009997, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.007230940231938194,
    kernel='rbf', max_iter=-1, probability=True, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
Mejor resultado: 0.786

ANALISIS DE PARAMETROS:
**********************
1) 0.786 (+/-0.016) para {'C': 37.51376470009997, 'gamma': 0.007230940231938194, 'kernel': 'rbf'} (17.282 seg)
2) 0.785 (+/-0.016) para {'C': 23.3630800690762, 'gamma': 0.008153187684193973, 'kernel': 'rbf'} (16.129 seg)
3) 0.785 (+/-0.018) para {'C': 25.45315542842208, 'gamma': 0.007726894696118861, 'kernel': 'rbf'} (16.325 seg)
Mostrando 3 mejores resultados de 10...

REPORTE DE CLASIFICACIÓN:
************************
              precision    recall  f1-score   support

No Deserción       0.74      0.80      0.77      1480
   Deserción       0.79      0.71      0.75      1480

    accuracy                           0.76      2960
   macro avg       0.76      0.76      0.76      2960
weighted avg       0.76      0.76      0.76      2960
```

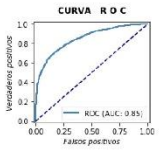




In [5]:
```
# 3) Multi-Layer perceptron
from sklearn.neural_network import MLPClassifier

# Valor del RANDOM STATE
np.random.seed(1998)

params_dist = {
    'hidden_layer_sizes': [(10,80,2), (25,30,1), (100)],
    'activation': ['tanh', 'relu','identity','logistic'],
    'alpha': stats.uniform(1e-10, 1e-2)
}

mlp_model = model.find_best_params_(MLPClassifier(max_iter = 500), params_dist)
```

```
Iniciar búsqueda de parametros...

RESULTADOS:
**********
Tiempo de ejecución: 8 mín 17.313 seg
Modelo optimo: MLPClassifier(activation='logistic', alpha=0.006818012653610943,
              batch_size='auto', beta_1=0.9, beta_2=0.999, early_stopping=False,
              epsilon=1e-08, hidden_layer_sizes=(25, 30, 1),
              learning_rate='constant', learning_rate_init=0.001, max_iter=500,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
Mejor resultado: 0.781

ANALISIS DE PARAMETROS:
**********************
1) 0.781 (+/-0.018) para {'activation': 'logistic', 'alpha': 0.006818012653610943, 'hidden_layer_sizes': (25, 30, 1)} (20.612 seg)
2) 0.777 (+/-0.035) para {'activation': 'tanh', 'alpha': 0.002433029942487875, 'hidden_layer_sizes': (10, 80, 2)} (13.379 seg)
3) 0.774 (+/-0.018) para {'activation': 'logistic', 'alpha': 0.00387740210177553, 'hidden_layer_sizes': (10, 80, 2)} (20.516 seg)
Mostrando 3 mejores resultados de 10...

REPORTE DE CLASIFICACIÓN:
*************************
              precision    recall  f1-score   support

No Deserción       0.75      0.79      0.77      1480
   Deserción       0.78      0.74      0.76      1480

    accuracy                           0.76      2960
   macro avg       0.77      0.76      0.76      2960
weighted avg       0.77      0.76      0.76      2960
```



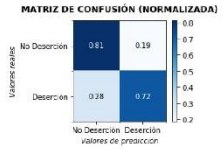


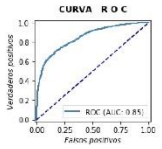
7

```
In [6]: # 4) Random forest
        from sklearn.ensemble import RandomForestClassifier

        # Valor del RANDOM STATE
        np.random.seed(1998)

        params_dist = {
            'n_estimators': stats.randint(50, 120),
            #'max_features': ['auto', 'sqrt', 'log2', None],
            'max_depth': stats.randint(10, 110),
            'min_samples_split': stats.randint(2, 20),
            #'min_samples_leaf': stats.randint(1, 8),
            'bootstrap': [True, False],
            'criterion': ['gini', 'entropy'],
        }

        rnf_model = model.find_best_params_(RandomForestClassifier(max_features = 'sqrt', min_samples_leaf = 6), params_dist)
```

8

```
Iniciar búsqueda de parametros...

RESULTADOS:
**********
Tiempo de ejecución: 38.213 seg
Modelo optimo: RandomForestClassifier(bootstrap=False, class_weight=None, criterion='entropy',
                       max_depth=44, max_features='sqrt', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=6, min_samples_split=11,
                       min_weight_fraction_leaf=0.0, n_estimators=97,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
Mejor resultado: 0.794

ANALISIS DE PARAMETROS:
**********************
1) 0.794 (+/-0.010) para {'bootstrap': False, 'criterion': 'entropy', 'max_depth': 44, 'min_samples_split': 11, 'n_estimators': 97} (1.310 seg)
2) 0.794 (+/-0.012) para {'bootstrap': False, 'criterion': 'entropy', 'max_depth': 89, 'min_samples_split': 17, 'n_estimators': 74} (0.990 seg)
3) 0.793 (+/-0.013) para {'bootstrap': True, 'criterion': 'entropy', 'max_depth': 11, 'min_samples_split': 2, 'n_estimators': 107} (0.817 seg)
Mostrando 3 mejores resultados de 10...

REPORTE DE CLASIFICACIÓN:
*************************
              precision    recall  f1-score   support

No Deserción       0.74      0.81      0.78      1480
   Deserción       0.79      0.72      0.75      1480

    accuracy                           0.77      2960
   macro avg       0.77      0.77      0.76      2960
weighted avg       0.77      0.77      0.76      2960
```





9

Exportando

In [7]:
```
knn_model._export('clf/knn_model.clf')
svm_model._export('clf/svm_model.clf')
mlp_model._export('clf/mlp_model.clf')
rnf_model._export('clf/rnf_model.clf')
```

```
Exportado correctamente clf/knn_model.clf (0.024 seg)...
Exportado correctamente clf/svm_model.clf (0.000 seg)...
Exportado correctamente clf/mlp_model.clf (0.000 seg)...
Exportado correctamente clf/rnf_model.clf (0.047 seg)...
```

10

ANEXO 11

ETAPA 4 - PRECISIÓN DE MODELOS

4. Precisión de modelos

Importando datos

In [1]:
```
import numpy as np
import pandas as pd

# Importando
muestra = pd.read_csv('csv/muestra.csv', sep = ',', header = 0, encoding = 'la
tin1')

# Seleccionando atributos importantes
best_features = pd.read_csv('csv/best_features.csv', sep = ',', header = 0, en
coding = 'latin1')
muestra = muestra.loc[:, best_features.Feature]

X = muestra.drop('desercion', 1)
Y = muestra['desercion']
```

Dividiendo los datos

In [2]:
```
from sklearn.model_selection import StratifiedKFold

# Valor del RANDOM STATE
np.random.seed(1998)

# Número de divisiones
n_folds = 10

# StratifiedKFold para que haya el igual número de casos de desercion y no des
ercion
skf = StratifiedKFold(n_splits = n_folds, shuffle = True)

folds = np.array([])
for train_index, test_index in skf.split(X, Y):
    folds = np.append(folds, {
        'X': X.iloc[test_index],
        'Y': Y.iloc[test_index]
    })
```

1

In [3]:
```
folds.shape
```
Out[3]: (10,)

Ejecutando los modelos

In [4]:
```
from lib.Model import Model

# Arreglo para recorrer y obtener datos
models = {
    'name': ['knn', 'svm', 'mlp', 'rnf'],
    'clf_sc': [
        Model()._import('clf/knn_model.clf'),
        Model()._import('clf/svm_model.clf'),
        Model()._import('clf/mlp_model.clf'),
        Model()._import('clf/rnf_model.clf')
    ],
    'data': []
}

# Recorriendo los modelos
for fold in folds:
    _data = {}
    for model_name, model in list(zip(models['name'], models['clf_sc'])):
        # Predicción
        Y_pred, Y_pred_prob = model._predict(fold['X'])

        # Métricas
        precision, recall, f1_score, auc = model.get_metrics(fold['Y'], Y_pred
, Y_pred_prob)

        # Solo almacenamos la presición del modelo
        _data[model_name] = precision * 100
    models['data'].append(_data)

# Información para crear DataFrame
data = models['data']

index = [ f'Fold {fold:2d}' for fold in range(1,n_folds + 1) ]

columns = models['name']

# Crear DataFrame con los resultados
resultados = pd.DataFrame(data, index = index, columns = columns)
```

```
Importado correctamente clf/knn_model.clf (0.725 seg)...
Importado correctamente clf/svm_model.clf (0.002 seg)...
Importado correctamente clf/mlp_model.clf (0.005 seg)...
Importado correctamente clf/rnf_model.clf (0.213 seg)...
```

2

In [5]:
```
resultados
```
Out[5]:

	knn	svm	mlp	rnf
Fold 1	74.903628	78.240317	76.549321	78.776621
Fold 2	74.693224	75.415868	75.672289	78.624462
Fold 3	73.435870	75.375358	75.096578	75.464274
Fold 4	73.688455	77.529782	76.869038	77.652767
Fold 5	76.804250	79.555138	76.285630	78.207835
Fold 6	75.602579	76.338558	75.296247	77.648877
Fold 7	73.611558	75.265788	75.177448	78.013946
Fold 8	73.477900	75.949386	76.188869	77.048784
Fold 9	74.223537	77.194026	77.750000	79.058877
Fold 10	73.513187	78.961780	76.377687	78.208498

Exportando

In [6]:
```
resultados.to_csv(r'csv/resultados.csv', header=True)
```

3

ANEXO 12

ETAPA 5 - PRUEBA DE HIPÓTESIS

5. Prueba de hipótesis

Importando datos

In [1]:
```
import numpy as np
import pandas as pd

# Importando
resultados = pd.read_csv('csv/resultados.csv', sep = ',', header = 0, index_co
l = 0, encoding = 'latin1')
```

In [2]:
```
resultados
```

Out[2]:

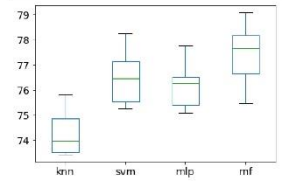
	knn	svm	mlp	rnf
Fold 1	74.903928	78.240317	76.549321	78.778621
Fold 2	74.663224	75.415858	75.672289	76.624482
Fold 3	73.435870	75.375358	75.096878	75.464274
Fold 4	73.698465	77.629762	76.899038	77.652767
Fold 5	75.804250	76.555138	76.285530	78.207835
Fold 6	75.662879	76.339558	75.256247	77.648877
Fold 7	73.511558	75.265788	75.177448	78.013946
Fold 8	73.477900	75.949386	76.188869	77.048784
Fold 9	74.223537	77.194026	77.750000	79.058877
Fold 10	73.513187	76.961780	76.377687	78.268408

In [3]:
```
resultados.describe(percentiles = [.5])
```

Out[3]:

	knn	svm	mlp	rnf
count	10.000000	10.000000	10.000000	10.000000
mean	74.286429	76.482697	76.129371	77.468293
std	0.913850	1.004238	0.840630	1.139055
min	73.435870	75.265788	75.096878	75.464274
50%	73.980998	76.447348	76.237200	77.850817
max	75.804250	78.240317	77.750000	79.058877

In [4]:
```
%matplotlib inline
import matplotlib.pyplot as plt

# Gráfico de cajas y bigotes
resultados.boxplot(grid = False, fontsize = 14)
plt.show()
```



ANOVA de medidas repetidas

In [5]:
```
import numpy as np
import pandas as pd

from lib.AnovaRM import ptl_anovaRM
```

In [6]:
```
F, p = ptl_anovaRM(resultados)
```

```
Trat.(Between groups):  SS_bg =  53.0281,  df_bg =   3,  MS_bg =  17.6760
      (Within groups):  SS_wg =  34.6213,  df_wg =  36
     Residual(Error):   SS_er =  14.7273,  df_er =  27,  MS_er =   0.5455
    Sujetos(Subject):   SS_sb =  19.8939,  df_sb =   9
               TOTAL:   SS_t  =  87.6494,   df_t =  39

F =  32.4059, p =   0.0000
```

Conclusión: Con un nivel de confianza 0.01 podemos afirmar que existen diferencias significativas.

Prueba posterior Tukey

In [7]:
```
# Cambiando la estructura para poder realizar la prueba estadística
data = []
for model_name, values in resultados.items():
    for fold, value in values.items():
        data.append(( model_name, value))
data = np.rec.array(data, dtype = [('model','U3'),('precision', '<f8')])
```

In [8]:
```
data
```

Out[8]:
```
rec.array([('knn', 74.90362846), ('knn', 74.69122439),
           ('knn', 73.43586992), ('knn', 73.69845466),
           ('knn', 75.80424976), ('knn', 75.60267857),
           ('knn', 73.51155767), ('knn', 73.47790003),
           ('knn', 74.22353742), ('knn', 73.51318692),
           ('svm', 78.24031739), ('svm', 75.4158584 ),
           ('svm', 75.37535766), ('svm', 77.5297619 ),
           ('svm', 76.55513766), ('svm', 76.33955753),
           ('svm', 75.26578765), ('svm', 75.94938578),
           ('svm', 77.19402566), ('svm', 76.96177958),
           ('mlp', 76.54932052), ('mlp', 75.67228883),
           ('mlp', 75.0969781 ), ('mlp', 76.89903842),
           ('mlp', 76.28553014), ('mlp', 75.29624708),
           ('mlp', 75.17744818), ('mlp', 76.18886932),
           ('mlp', 77.75      ), ('mlp', 76.37798684),
           ('rnf', 78.77662127), ('rnf', 76.52446161),
           ('rnf', 75.46427418), ('rnf', 77.65275674),
           ('rnf', 78.20783486), ('rnf', 77.64887679),
           ('rnf', 78.01394603), ('rnf', 77.04678363),
           ('rnf', 79.05887732), ('rnf', 76.26849777)],
          dtype=[('model', '<U3'), ('precision', '<f8')])
```

In [9]:
```
from statsmodels.stats.multicomp import MultiComparison
mc = MultiComparison(data['precision'], data['model'] )
print(mc.tukeyhsd(0.01))
```

```
Multiple Comparison of Means - Tukey HSD, FWER=0.01
=====================================================
group1 group2 meandiff p-adj   lower  upper  reject
-----------------------------------------------------
   knn    mlp   1.8429  0.001  0.3762 3.3097   True
   knn    rnf   3.1799  0.001  1.7131 4.6466   True
   knn    svm   2.1963  0.001  0.7295  3.663   True
   mlp    rnf   1.3369 0.0214 -0.1298 2.8037  False
   mlp    svm   0.3533  0.835 -1.1134 1.8201  False
   rnf    svm  -0.9836  0.131 -2.4503 0.4831  False
-----------------------------------------------------
```

Conclusión: Sólo *knn* es diferente a los demás modelos.