Pontificia Universidad Católica del Ecuador

Facultad de Hábitat, Infraestructura y Creatividad

TEMA:

Modelo predictivo para analizar la sostenibilidad financiera de las carreras de una institución de educación superior.

AUTOR:

Jorge Armando Tatayo Chanaluisa

Trabajo Previo a la Obtención del Título de Magister en Sistemas de Información

Mención Data Science

TUTOR:

Msc. Miguel Ortiz Navarrete

Quito, marzo de 2025

DEDICATORIA

A ti, Adriana, por ser mi compañera de vida y estar a mi lado en cada paso que doy, por brindarme tu apoyo en esta meta que un día me propuse cumplir.

A mis padres, que siempre me brindaron su apoyo desde el inicio de mi carrera y ahora ven como cada día sigo creciendo profesionalmente gracias al impulso que un día me dieron.

A mi familia, que siempre ha creído en mí y su confianza me llena de fuerza para seguir cumpliendo mis objetivos.

AGRADECIMIENTO

Quiero expresar mi agradecimiento y gratitud a mi asesor de tesis por su guía y apoyo en la elaboración de este trabajo, a La Pontificia Universidad Católica del Ecuador por abrirme las puertas y darme la oportunidad de formar parte de esta gran institución, a cada uno de los docentes que compartieron sus conocimientos y experiencia durante el desarrollo de este programa académico, al Instituto Cordillera y sus autoridades que me brindaron su apoyo y el acceso a los recursos técnicos para el desarrollo de este proyecto.

RESUMEN

Este proyecto se desarrolló con el objetivo de crear un modelo predictivo que brinde la información necesaria para analizar la sostenibilidad financiera de las carreras de una institución de educación superior, la información obtenida por el modelo servirá de apoyo para que las autoridades puedan tomar decisiones estratégicas.

En la búsqueda del mejor modelo que posea un alto porcentaje de exactitud en sus predicciones, se realizó pruebas con los algoritmos de Regresión Logística, Random Forest, Percepción Multicapa, si bien todos los modelos de clasificación arrojaron métricas muy similares dando un porcentaje de exactitud mayor al 90%, finalmente se seleccionó el modelo con Random Forest ya que tiene una alta precisión, tiene un buen manejo de los datos faltantes y la aleatoriedad reduce el riesgo de sobre ajuste.

En el desarrollo de este proyecto se utilizó la investigación de tipo aplicada que tiene como objetivo resolver problemas concretos, y se utilizó un enfoque cuantitativo ya que se analizó los datos históricos académicos y financieros de los últimos 10 años de la institución, para la parte de minería de datos se trabajó con CRISP - DM que brindó una estructura clara y flexible para la comprensión de los datos.

Gracias a los resultados obtenidos durante el análisis de los datos y de los modelos desarrollados se puede indicar que existen procesos en el área financiera que se deben revisar para obtener una información más precisa y de esta forma mejorar la exactitud de los modelos, existen carreras que deben ser evaluadas a fondo para tomar la decisión de reestructurarlas, finalmente se recomendó revisar los procesos administrativos internos para que sirvan de apoyo a los estudiantes en la culminación de sus estudios.

ABSTRACT

This project was developed with the objective to create a predictive model that provides the necessary information to analyze the financial sustainability of the careers in a higher education institution. The information obtained from the model will provide support to the authorities in order to make strategic decisions.

In the search for the best model with a high accuracy percentage in its predictions, tests were conducted using the Logistic Regression, Random Forest, and Multilayer Perceptron algorithms. All classification models produced very similar metrics, achieving an accuracy rate of over 90%, the Random Forest model was ultimately selected due to its high precision, effective handling of missing data, and the randomness factor that reduces the risk of overfitting.

This project was carried out using applied research, which aims to solve specific problems, and a quantitative approach was employed since historical academic and financial data from the past 10 years of the institution were analyzed. For data mining, the CRISP-DM methodology was used, providing a clear and flexible structure for understanding the data.

Based on the results obtained from the data analysis and the developed models, it was determined that certain processes in the financial area should be reviewed to obtain more accurate information and thus improve model accuracy. Additionally, some programs need to be thoroughly evaluated to decide whether restructuring is necessary. Finally, it was recommended to review internal administrative processes to better support students in completing their studies.

CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

CAPÍTULO I: INTRODUCCIÓN

1. Introducción

El Consejo de Aseguramiento de la Calidad de la Educación Superior (CACES) es el ente regulador de la educación superior en el Ecuador, este organismo es el responsable de la calidad de las carreras que ofertan las instituciones técnicas y tecnológicas, actualmente en nuestro país existen un total de 206 institutos técnicos y tecnológicos de los cuales 145 son particulares y el resto son públicos que del estado (CES, 2024), las instituciones privadas deben competir entre ellas por tratar de mantener y sobre todo de aumentar el número de estudiantes matriculados, esto hace que enfrenten constantes desafíos para asegurar su sostenibilidad financiera.

El Instituto Superior Tecnológico Cordillera está ubicado en la ciudad de Quito y tiene una trayectoria de más de 30 años de funcionamiento brindando educación superior de calidad a la población, actualmente cuenta con 16 carreras divididas en diferentes áreas de estudio como la Administración, el Diseño y Comunicación, Informática y Tecnología, Marketing, Salud y Educación, con un promedio de estudiantes matriculados que sobrepasa los 4 mil alumnos (Rendición de cuentas 2023 , 2023), antes de la pandemia el número de estudiantes era mayor a los 6 mil, actualmente la institución no cuenta con herramientas de analítica de datos que le permitan prever posibles problemas financieros.

El Instituto Cordillera quiere mejorar su capacidad para evaluar la viabilidad económica de sus programas académicos, pero no cuenta con herramientas modernas como el Machine Learning (ML) que le permita anticipar problemas financieros y le ayude a optimizar la asignación de sus recursos.

La capacidad financiera de la institución depende netamente de factores como la matriculación de nuevos estudiantes, de mantener un alto porcentaje en la retención de sus alumnos, de la diversificación de sus ingresos y de un manejo adecuado de los recursos económicos, pero la crisis económica y social por la que atraviesa el país afecta significativamente la estabilidad financiera de todas las instituciones del país.

Con la creación de este modelo predictivo queremos apoyar al Instituto Cordillera para que analice la situación financiera de las carreras que oferta y para esto vamos a buscar patrones o tendencias en los datos obtenidos de la institución que nos ayuden a identificar posibles riesgos financieros para la institución.

En el desarrollo de nuestro proyecto vamos a utilizar CRISP-DM como metodología ya que es la más utilizada en proyectos de minería de datos, vamos a utilizar algoritmos de machine lerarning como Regresión lineal, Regresión logística, Random forest y Redes neuronales que nos ayuden a predecir la sostenibilidad financiera de las carreras del instituto.

Para la recolección de la información vamos a realizar consultas SQL a la base de datos del sistema académico y financiero de la institución y para el procesamiento y análisis de datos vamos a utilizar Python como herramientas de apoyo.

Como resultado final de este trabajo se espera obtener un modelo predictivo confiable que le permita al Instituto Cordillera realizar un análisis de la sostenibilidad económica de sus carreras y generar proyecciones que apoyen a las autoridades en la toma de decisiones estratégicas.

1.1. Planteamiento del problema

El Instituto Cordillera es una institución privada y depende únicamente de los ingresos generados por los servicios que oferta, es por esto que factores como la disminución en el número

de estudiantes, los cambios en las normativas de la educación superior, el aumento de los costos operativos y las variaciones en el mercado laboral le afectan directamente en sus ingresos.

Al iniciar un nuevo periodo académico se realiza una planificación académica donde se define el número de cursos que se deben abrir por cada nivel, los horarios de cada docente y se verifica si es necesario la contratación de nuevo personal, esta definición se la vienen realizando de forma estimada lo que ha provocado que se tenga que abrir o cerrar cursos dependiendo de la demanda de alumnos, estos cambios afectan directamente a los estudiantes ya que les obliga a modificar sus horarios o realizar gestiones adicionales que les causa malestar y en ocasiones ha provocado su retiro de la institución que debe realizar la devolución de aranceles.

Las modificaciones en el número de cursos también afectan a la contratación del personal docente porque no se pueden realizar las contrataciones con anticipación, dando como resultado que se deba iniciar el periodo académico sin la planta docente completa, todos estos cambios realizados sin una planificación adecuada afectan negativamente la imagen institucional.

1.2. Objetivos

1.2.1. Objetivo general

Desarrollar un modelo predictivo que permita evaluar la sostenibilidad financiera de los programas académicos de una institución de educación superior.

1.2.2. Objetivos específicos

- Analizar la literatura referente a los modelos predictivos y sus aplicaciones en el ámbito de la gestión financiera.
- Identificar los factores que afectan la sostenibilidad financiera de los programas académicos ofertados por la institución.

- Crear y validar un modelo predictivo basados en datos históricos y actuales.
- Validar la precisión del modelo utilizando datos reales de la institución.
- Presentar los resultados obtenidos por el modelo a las autoridades de forma clara y entendible.

1.3. Justificación

Las instituciones de educación superior particulares dependen financieramente de su gestión y sobre todo del número de estudiantes matriculados en cada periodo académico, también de los ingresos que puedan generar por autogestión, esto provoca gran incertidumbre cada nuevo periodo que inicia por posible número de estudiantes que ingresarán en la institución a cada uno de los programas académicos.

Se entiende que si se reduce el número de alumnos los ingresos serán menores y por ende se deben recortar los gastos operativos, desde esta perspectiva resulta de vital importancia tener una herramienta que me permita analizar en el tiempo que tan sostenible es cada una de las carreras que se ofertan, ya que no se debería destinar recursos a una carrera que no sostiene financieramente.

Con el desarrollo de este modelo queremos brindar de una herramienta a la alta dirección que les ayude a tomar decisiones más informadas y planificar las futuras inversiones que se realizarán a largo plazo, también optimizar la asignación de los recursos e identificar los posibles riesgos como pueden ser el cierre de alguna carrera.

1.4. Alcance

En el desarrollo de nuestro proyecto se ha contemplado el diseño, desarrollo y evaluación de un modelo predictivo que se enfoca en analizar la sostenibilidad financiera de las carreras del Instituto Tecnológico Superior Cordillera, este modelo se basará en el análisis de la información

histórica de la institución, basándonos en el número de estudiantes matriculados, ingresos por aranceles, costos operativos, costos de personal, margen de rentabilidad y otros indicadores financieros que nos pueda aportar información.

El alcance de este proyecto está limitado al análisis de la información obtenida del Instituto Tecnológico Superior Cordillera y al análisis de los aspectos financieros de la institución, no se tomarán en cuenta temas como el rendimiento académico o el grado de satisfacción de los estudiantes.

Para el desarrollo del proyecto se utilizarán herramientas de software como Python y técnicas de Machine Learning, el resultado final será un modelo predictivo que le permita a la institución mejorar su capacidad de planificación en la asignación de recursos de las carreras que oferta.

Este modelo está desarrollado para el Instituto Tecnológico Superior Cordillera, pero puede adaptarse a otras instituciones de educación superior, la implementación de los cambios sugeridos en la gestión administrativa y financiera dependerá de la institución y no forman parte de este proyecto.

CAPÍTULO II: MARCO TEÓRICO Y CONCEPTUAL

2. Marco Teórico y Conceptual

2.1. Sostenibilidad financiera

Se puede conceptualizar a la expresión “sostenibilidad financiera” (SF) como la capacidad de una persona o una entidad de realizar sus operaciones de manera eficiente a través del tiempo, mediante un adecuado equilibrio entre recursos económicos, humanos y técnicos (Macías Macías & Toala Mendoza , 2022).

La sostenibilidad financiera de la institución al ser de tipo privada se sustenta básicamente del número de estudiantes matriculados, por este motivo la capacidad de proyectar cierta información importante como el número de estudiantes, los ingresos, los costos, etc. con base en la información histórica es de gran importancia para la institución ya que le permite hacer un uso más adecuado de los recursos.

2.2. Flujo de caja

El flujo de caja es una herramienta que contribuye a la toma de decisiones y la distribución del capital en forma óptima, para mantener una estructura sólida, apta para enfrentar amenazas del mercado sin perder de vista las metas y objetivos propuestos, permitiendo definir la viabilidad de las oportunidades de negocio y estableciendo la necesidad de obtener un financiamiento, realizar una inversión o cumplir con obligaciones (Taborda Blandón, Castaño Zuluaga, Durán Vásquez, Conto López, & Reyes Moreno, 2024).

El flujo de caja es un indicador importante dentro la institución ya que nos permite conocer la disponibilidad de efectivo o el déficit de caja para identificar los momentos en que la institución

puede realizar inversiones y posibles endeudamientos, la proyección de este tipo de indicadores ayuda a la institución a realizar planificaciones más efectivas analizando diferentes escenarios como la reducción de matrículas, el incremento de costos (compra de equipos tecnológicos, mantenimiento de infraestructura, etc.).

Con esta información proyectada se podrán realizar cambios en las políticas internas como el incremento de descuentos o la modificación de las políticas de pago (cobros mensuales vs. semestrales o trimestrales).

2.3. Margen de utilidad neto

"El margen neto es un indicador de la rentabilidad de una empresa después de deducir todos los costos y gastos asociados con la producción o venta de bienes o servicios, incluyendo los impuestos" (Marchena, 2023).

Basados en esta definición podemos indicar que la utilidad neta es una métrica financiera que indica el porcentaje de ingresos que una empresa retiene como ganancia después de deducir sus costos y gastos.

El margen de utilidad neto en conjunto con otros indicadores como el flujo de caja me sirve para determinar la sostenibilidad financiera de la institución y de las carreras, para esto se puede hacer el ejercicio de proyectar ciertos valores como el margen de rentabilidad con diferentes escenarios como el incremento o reducción del número de matriculados, ajustar los costos operativos, etc.

En base a la información obtenida de las proyecciones la institución puede tomar decisiones como abrir nuevos programas, cerrar las carreras con márgenes negativos o cambiar precios de matrícula, etc.

2.4. Análisis predictivo

El análisis predictivo es un área de la minería de datos que consiste en la extracción de información existente en los datos y su utilización para predecir tendencias y patrones de comportamiento, pudiendo aplicarse sobre cualquier evento desconocido, ya sea en el pasado, presente o futuro. El análisis predictivo se fundamenta en la identificación de relaciones entre variables en eventos pasados, para luego explotar dichas relaciones y predecir posibles resultados en futuras situaciones (Timón, 2017).

Según lo descrito anteriormente, podemos indicar que el análisis predictivo en nuestra institución se utilizará para optimizar la gestión financiera anticipando tendencias de alumnos matriculados, ingresos por cobro de aranceles y costos operativos, además de permitir la identificación de posibles riesgos financieros y áreas de mejora, esta información nos facilita la planificación de los presupuestos, la asignación eficiente de los recursos y la identificación de carreras que no son rentables.

2.5. Modelos de regresión y clasificación

2.5.1. Regresión logística

La regresión logística, es una de las técnicas estadísticas utilizadas en machine learning como paradigma del aprendizaje supervisado. Cuya finalidad es hacer clasificaciones. Con ello podemos crear un modelo que tome una variable y determine la posibilidad de la ocurrencia de un fenómeno (Becerra Correa & Leguizamón Páez, 2024).

Como indica la definición anterior este algoritmo nos permite predecir la probabilidad de que un vente ocurra, este algoritmo es muy utilizado en clasificación binaria donde el resulto puede

ser si/no o 1/0, precisamente por estas características este algoritmo lo vamos a utilizar para predecir si las carreras pueden ser rentables a largo plazo.

2.5.2. Random forest

El método Random Forest, tal y como su nombre lo indica, consiste en una gran cantidad de árboles de decisión individuales que operan como un conjunto, formando así un bosque. Dentro de este bosque, cada árbol individual genera una predicción de clase y la clase con más votos se convierte en la predicción de nuestro modelo (Serra Marrugat, 2020).

Como se muestra en el párrafo anterior Random Forest crea un bosque de subconjuntos aleatorios donde el objetivo es llegar a un resultado único, en nuestro caso esto nos ofrece un modelo que es preciso y robusto, además ayuda a reducir el sobreajuste ya que evita memorizar datos.

2.5.3. Redes neuronales

Las redes artificiales de neuronas tratan, en cierto modo, de replicar el comportamiento del cerebro, donde tenemos millones de neuronas que se interconectan en red para enviarse mensajes unas a otras. Esta réplica del funcionamiento del cerebro humano es uno de los "modelos de moda" por las habilidades cognitivas de razonamiento que adquieren (Sandoval, 2018).

Con el afán de explorar las mejores opciones para la creación de nuestro modelo vamos a utilizar algoritmos de redes neuronales, como se indicó en la definición anterior estos algoritmos tratan de replicar el funcionamiento de nuestro cerebro, es decir que están creadas para trabajar con datos muy complejos donde los otros algoritmos no funcionan bien, si los resultados obtenidos no son buenos este algoritmo puede ser una mejor opción.

2.5.4. Regresión lineal múltiple

La regresión lineal múltiple también es una técnica estadística que se utiliza para analizar la relación entre una variable dependiente (lo que queremos predecir) y dos o más variables independientes (lo que usamos para hacer la predicción), el uso de todas estas variables permite hacer una predicción más precisa (Bruce, Bruce, & Gedeck, 2022).

En base a lo señalado anteriormente podemos indicar que este modelo de regresión nos permitirá evaluar la relación entre una variable dependiente como el margen de utilidad de una carrera y varias variables independientes como costos operativos, costos de personal y número de estudiantes, esto nos permitirá identificar cuáles son los factores que tienen mayor peso en la viabilidad financiera de una carrera, de esta forma se podrá tomar decisiones para mejorar la rentabilidad como ajustar los costos o replantear los valores del servicio.

2.6. Métricas de evaluación de modelos

La evaluación de la calidad de un modelo implica usar una combinación de métricas para asegurar que el modelo sea preciso, confiable y generalice bien a los nuevos datos, dependiendo del tipo de modelo y de la naturaleza de los datos se debe elegir las métricas que mejor se alineen con los objetivos de la institución.

2.6.1. Mean Squared Error (MSE)

Es una forma de medir qué tan buenas son las predicciones que realiza un modelo comparadas con los valores reales, por ejemplo en el caso de tener un modelo que predice las temperaturas diarias para saber qué tan cerca están esas predicciones de las temperaturas reales se debe calcular la diferencia entre cada valor predicho y el valor real, se eleva al cuadrado cada una de esas diferencias (para que todos los errores sean positivos y se dé más peso a los errores grandes)

obtenemos el promedio de esos valores cuadrados y el resultado es el MSE, Un MSE más bajo significa que las predicciones están más cerca de los valores reales, lo que nos indica un modelo más preciso (Bruce, Bruce, & Gedeck, 2022).

El Mean Squared Error (MSE) se lo va a utilizar como una métrica para evaluar la precisión del modelo, en nuestro proyecto buscamos que el valor del MSE sea bajo ya que esto nos indica que los valores predichos están más cerca de los valores reales, indicando que nuestro modelo es preciso y tiene buen rendimiento.

Con esta métrica también vamos a comparar entre diferentes modelos para seleccionar el que tenga un MSE más bajo ya que esto nos asegura que el modelo elegido es el más confiable para hacer predicciones precisas.

2.6.2. El R-cuadrado (R^2)

Es una medida que nos dice qué tan bueno es un modelo de regresión lineal dependiendo de cómo se ajusta a los datos reales, en términos generales R-cuadrado nos indica el porcentaje de la variación en la variable que estamos tratando de predecir (variable dependiente) que puede ser explicado por las variables que usamos para hacer la predicción (variables independientes), si se tiene como resultado un R-cuadrado de 0.80, quiere decir que el 80% de la variación en la variable dependiente puede ser explicada por el modelo, mientras que el 20% restante se debe a otros factores que el modelo no puede explicar (Bruce, Bruce, & Gedeck, 2022).

El R-cuadrado (R^2), o coeficiente de determinación lo vamos a utilizar para medir la proporción de la variabilidad en la variable dependiente que puede ser explicada por el modelo, es decir que el R^2 indica qué tan bien los datos predichos por el modelo se ajustan a los datos observados.

Un valor de R^2 cercano a 1 sugiere que el modelo explica la mayor parte de la variabilidad de los datos lo que confirma que es un buen modelo predictivo.

2.6.3. La validación cruzada

La estrategia de Validación Cruzada consiste en la división de un conjunto de muestras que se pueden analizar en dos conjuntos disjuntos de datos. Uno de estos conjuntos entrenará las muestras que contiene, y los resultados obtenidos se aplicarán al otro conjunto que será utilizado para la clasificación de muestras. Hay que tener en cuenta que la división de las muestras en dos conjuntos fijos es una simplificación de la implementación de una división en k subconjuntos (Murillo, 2019).

Esta técnica se va a utilizar en el proyecto para evaluar la precisión del modelo predictivo, nos sirve para asegurar que los resultados obtenidos no dependan de una partición específica de los datos, se dividirá los datos en varios subconjuntos, se entrenará el modelo con una parte de ellos y se lo probara con la parte restante, este proceso se lo repite varias veces y los resultados se promedian para obtener una estimación más confiable del rendimiento del modelo.

2.6.4. Matriz de confusión

La matriz de confusión se define como una herramienta que permite la visualización del desempeño de un algoritmo que se emplea en aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real. Uno de los beneficios de las matrices de confusión es que facilitan ver si el sistema está confundiendo las diferentes clases o resultados de la clasificación (Serra Marrugat, 2020).

Como se indica en la definición anterior lo que buscamos con la utilización de la matriz de confusión es medir el desempeño de nuestro modelo, esta métrica nos muestra que tan bien nuestro modelo está clasificando las clases y en donde se equivoca más, también es útil para ver si el modelo sesgado hacia la clase mayoritaria.

2.6.5. La curva de ROC

"La curva ROC es una herramienta estadística que representa en un solo gráfico los valores de sensibilidad y especificidad para los diferentes puntos de corte, permitiendo visualizar la variación del rendimiento del modelo en función de estos puntos" (Aznar Gimeno, 2024).

Según la definición anterior se utilizará la curva de ROC para evaluar nuestros modelos de clasificación ya que nos permite observar que tan bien el modelo clasifica los datos en las diferentes clases.

2.6.6. F1-score

"La medida F1-score fusiona las métricas accuracy con recall, presentando diferencias en el rendimiento de un clasificador que no son revelados únicamente con la accuracy" (Borja Robalino, Monleón Getino, & Rodellar, 2020).

Como se indica en la definición anterior es métrica nos ayuda cuando tenemos clases desbalanceadas ya que combina precisión y sensibilidad para evaluar un modelo de clasificación binaria.

2.7. Minería de datos

La minería de dato constituye el proceso de extraer conocimiento útil y comprensible, previamente desconocido, desde grandes cantidades de datos almacenados en distintos formatos,

para encontrar modelos a partir de los datos. Para que este proceso sea efectivo, debería ser automático y el uso de los patrones descubiertos debería ayudar a la toma de decisiones. Dentro de los principales métodos utilizados en la minería de datos se encuentran: búsqueda de asociaciones, detección de ciclos temporales, predicción (Rodríguez Rodríguez, Rodríguez González, Pino Tarragó, & Domínguez Gálvez, 2021).

En el desarrollo del proyecto vamos a realizar una minería de datos en la información obtenida de la institución recolectando y limpiando los datos históricos más relevantes, luego se realizará un análisis exploratorio para identificar patrones y seleccionar las variables significativas, finalmente se utilizarán algoritmos de aprendizaje automático para crear modelos y entrenarlos con los datos históricos, una vez entrenados los modelos se deben validar para seleccionar los mejores.

2.8. Técnicas de minería de datos

2.8.1. Clasificación

La clasificación es la técnica de extracción de datos más comúnmente aplicada. Utiliza un conjunto de apoyo para la elección entre varias líneas de acción, permitiendo explorar los posibles resultados para varias opciones, y evaluar el riesgo y las recompensas para cada posible curso de acción. Estas decisiones generan reglas, que luego se usan para clasificar los datos (Zárate Valderrama, Bedregal Alpaca, & Cornejo Aparicio, 2020).

Como se indicó en la definición anterior la técnica de clasificación en Machine Learning nos permiten asignar reglas a las variables basados en sus características, para finalmente clasificar los datos, esta técnica es muy utilizada en problemas de clasificación binaria como: detección de

anomalía, predicción de categorías, diagnóstico y detección, en nuestro caso particular la usaremos para predecir si las carreras serán rentables o no.

2.8.2. Regresión

Al igual que la clasificación, la regresión (regression) es una tarea de aprendizaje inductivo que ha sido ampliamente estudiada y utilizada. Se puede definir, de forma informal, como un problema de «clasificación con clases continuas». Es decir, los modelos de regresión predicen valores numéricos en lugar de etiquetas de clase discretas. A veces también nos podemos referir a la regresión como «predicción numérica» (Casas Roma, Nin Guerrero, & Julbe López, 2019).

Uno de los modelos que vamos a desarrollar en este proyecto estará basado en el concepto de la regresión ya que es una herramienta eficaz para construir modelos predictivos, este método utiliza datos históricos relevantes, como matrículas y costos operativos, para poder identificar relaciones entre las variables, al ajustar esta información a una ecuación lineal, se pueden predecir tendencias financieras futuras, como ingresos proyectados o déficits, con base en factores como los cambios en el número de alumnos matriculados, ajustes en los costos de matrícula o políticas de descuentos.

2.9. Python

Python es un lenguaje consistente y maduro, utilizado en diversos ámbitos: web, desarrollo de interficies gráficas, programación de sistemas, networking, bases de datos, cálculo numérico y aplicaciones científicas, programación de juegos y multimedia, gráficos e inteligencia artificial, entre muchos otros. Se trata de un lenguaje multiplataforma, es decir, disponible para los principales sistemas operativos, y se incluye automáticamente en las distribuciones Linux y en los ordenadores Macintosh. Además, proporciona todas las herramientas para escribir de manera

sencilla programas portables, es decir, que se comportan del mismo modo si se ejecutan sobre distintas plataformas (Buttu, 2020).

Se utilizará Python como lenguaje de programación para la construcción del modelo aprovechando su simplicidad, versatilidad y la extensa cantidad de bibliotecas específicas para análisis de datos y aprendizaje automático, con Python se puede limpiar y procesar los datos usando bibliotecas como Pandas y NumPy, también se puede visualizar los datos utilizando Matplotlib y Seaborn, además nos permite evaluar los modelos predictivos con herramientas como Scikit-learn.

Las bibliotecas indicadas nos proporcionan funciones y algoritmos para realizar tareas como regresión, clasificación o agrupamiento, facilitando la creación, prueba y ajuste del modelo.

2.9.1. Pandas

Pandas es una herramienta de manipulación de datos de alto nivel desarrollada por Wes McKinney. En su inicio es construido sobre Numpy y ahora compatible con Arrow y permite el análisis de datos que cuenta con las estructuras de datos que necesitamos para limpiar los datos en bruto y que sean aptos para el análisis (por ejemplo, tablas). Pandas permite realizar tareas importantes, como alinear datos para su comparación, fusionar conjuntos de datos, gestión de datos perdidos, etc., se ha convertido en una librería muy importante para procesar datos a alto nivel en Python (Zapata, 2024).

En el desarrollo de este proyecto esta librería es fundamental ya que nos permitirá manipular los datos y realizar operaciones como filtrado, agrupación y combinación de datos, también nos ayudará con el manejo de datos faltantes, transformación de columnas y la realización del análisis exploratorio inicial.

2.9.2. NumPy

NumPy es una libreria de Algebra Lineal para Python, la razón por la cual es tan importante para Data Science con Python es que casi todas las librerías del ecosistema de PyData confían en NumPy como uno de sus principales componentes (Zapata, 2024).

El uso de esta librería complementa el manejo de los datos ya que nos permitirá realizar operaciones rápidas en arreglos y matrices, que son la base de muchas operaciones matemáticas en la creación de modelos predictivos.

2.9.3. Matplotlib

Matplotlib es una librería Python open source, desarrollada inicialmente por el neurobiólogo John Hunter en 2002. El objetivo era visualizar las señales eléctricas del cerebro de personas epilépticas. Para conseguirlo, quería replicar las funcionalidades de creación gráfica de MATLAB con Python. Tras el fallecimiento de John Hunter en 2012, Matplotlib ha sido mejorado a lo largo del tiempo por numerosos contribuidores de la comunidad open source. Se ha utilizado para crear gráficas y diagramas de gran calidad. Es una alternativa open source a MATLAB (Seaborn: todo sobre la herramienta de Data Visualization Python, 2023).

En el desarrollo del proyecto la librería Matplotlib nos permitirá crear gráficos como histogramas o gráficos de líneas para entender las tendencias y distribuciones.

2.9.4. Seaborn

Seaborn es una biblioteca para crear gráficos estadísticos en Python. Está basada en Matplotlib, y se integra con las estructuras de Pandas. Esta biblioteca es tan potente como

Matplotlib, pero aporta simplicidad y funciones inéditas. Permite explorar y comprender rápidamente los datos (Seaborn: todo sobre la herramienta de Data Visualization Python, 2023).

Esta librería nos permitirá la creación de gráficos estadísticos avanzados como mapas de calor, gráficos de dispersión y gráficos de distribución, que nos ayudarán a descubrir las relaciones entre variables.

2.9.5. Scikit-learn

"Scikit-learn es una librería de software libre de aprendizaje automático en lenguaje Python. Contiene múltiples utilidades para el tratamiento de conjuntos de datos, algoritmos de clasificación, de clustering, de regresión etc" (Valverde, 2018).

Con esta biblioteca se va a crear el modelo predictivo también se utilizará Pandas y NumPy para cargar, analizar y dar formato a los datos que se van a utilizar, con estos datos se alimentará el modelo de Scikit-learn que luego se utilizará para hacer predicciones.

2.10. Metodología CRISP-DM

La metodología CRISP-DM (Cross-Industry Standard Process for Data Mining) es un modelo de proceso que proporciona un marco estructurado para llevar a cabo proyectos de minería de datos.

La estructura de CRISP-DM se compone de seis fases principales que se detallan a continuación:

1. **Comprensión del negocio:** En esta fase se debe definir los objetivos del proyecto y los requisitos desde la perspectiva del negocio, es decir se debe traducir esto en un problema de minería de datos.

2. **Comprensión de los datos:** Recolectar datos iniciales y proceder a explorar y familiarizarse con ellos, identificando los posibles problemas de calidad y descubriendo patrones iniciales.
3. **Preparación de los datos:** Una vez que la información ha sido revisada y comprendida se debe seleccionar, limpiar y transformar los datos para que estén listos para el modelado.
4. **Modelado:** En esta fase se debe aplicar técnicas de modelado a los datos ya preparados, aquí es donde todo el trabajo comienza a tener sentido, se crearán y evaluarán varios modelos basados en técnicas como la regresión.
5. **Evaluación:** Se debe evaluar el modelo para asegurarse de que cumple con los objetivos de la institución y que es confiable, esta fase puede llevar a realizar revisiones de las fases anteriores.
6. **Despliegue:** Esta es la fase final donde se debe implementar el modelo en un entorno de producción y asegurarse de que los resultados sean utilicen para tomar decisiones informadas.

"Cada fase de la metodología es iterativa y puede requerir de revisiones y ajustes a medida que se avanza en el proyecto, lo que permite una flexibilidad que es crucial en el campo de la minería de datos" (Shimaoka, 2024).

El uso de la metodología CRISP-DM nos permitirá tener un enfoque estructurado para el desarrollo del proyecto, a continuación, se indica de forma resumida como se desarrollará cada una de las fases:

1. Primero es necesario entender los objetivos financieros de la institución es una reunión con el área financiera.

2. Luego debemos recopilar y explorar los datos relevantes como ingresos, gastos y número de matriculados, etc.
3. Vamos a preparar los datos limpiándolos y transformándolos según sea necesario.
4. Debemos seleccionar y entrenar el modelo de regresión de forma adecuada.
5. Vamos a evaluar su rendimiento con métricas específicas para cada modelo.
6. Debemos desplegar el modelo en el entorno de la institución.

CAPÍTULO III: MARCO METODOLÓGICO

3. Marco Metodológico

3.1. Marco Metodológico de Investigación

Para el desarrollo de nuestro proyecto vamos a utilizar una investigación de tipo aplicada este tipo de investigación nos permite resolver problemas prácticos transformando el conocimiento teórico existente en una herramienta que podamos utilizar para analizar los datos de la institución.

Vamos a utilizar un enfoque cuantitativo que se basa en la recolección y el análisis de datos numéricos, en nuestro caso será la información financiera de la institución (como los ingreso por matrículas, costos operativos, tasas de graduación), para lo cual utilizaremos un proceso deductivo mediante el cual se analizará la información recolectada para identificar patrones y relaciones que nos permitan alimentar a nuestro modelo.

La población de estudio incluye a todas las carreras activas de la institución en los últimos 10 años, en este caso la muestra estará compuesta por todos los estudiantes matriculados en las carreras durante esos años, esto nos asegura que no se omitan datos relevantes.

La información con la cual se va a desarrollar el proyecto se recolectará de la base de datos institucional, el Instituto Cordillera utiliza un sistema académico y financiero que se encarga de gestionar toda la información, esta será nuestra fuente de información ya que contiene todos datos académicos y financieros que se requieren para nuestro análisis.

Una vez que hemos recolectado la información estos datos serán analizados utilizando herramientas estadísticas y de machine learning, se utilizarán técnicas como el análisis exploratorio para identificar las posibles relaciones iniciales entre variables, se utilizarán modelos de regresión

y clasificación que nos permitan realizar predicciones que ayuden a confirmar la sostenibilidad de las carreras.

Las principales limitaciones para el desarrollo de nuestro proyecto incluyen la dependencia de los datos históricos disponibles y la dificultad para medir o clasificar algunas de las variables cualitativas que puedan influir en la sostenibilidad financiera de la institución.

Este proyecto cumple con los principios éticos al garantizar la confidencialidad de la información proporcionada por la institución, su uso será exclusivamente para fines académicos, los datos recolectada no contienen información personal de alumnos, docentes o personal administrativo de la institución.

3.2. Marco Metodológico de Ciencia de Datos

Para el desarrollo de la metodología de la Ciencia de Datos se utilizará el enfoque de CRISP-DM, que incluye una descripción de las fases que se debe seguir en un proyecto de minería de datos, las tareas que son necesarias en cada fase y una explicación de las relaciones entre las tareas, con esta información tenemos una guía bien estructurada con la cual podemos desarrollar nuestro proyecto.

El primer paso que debemos abordar según la metodología CRISP-DM es la “Comprensión del negocio”, en esta fase vamos a recopilar la información sobre la situación actual de la institución, lo que nos permitirá definir los objetivos que buscan cumplir las autoridades del instituto, también es necesario determinar la disponibilidad de recursos, los requisitos del proyecto y evaluar los posibles riesgos, finalmente se debe definir los criterios que se utilizarán para establecer si el proyecto ha tenido éxito.

La siguiente etapa es la “Comprensión de los datos”, aquí se van a recolectar la información desde la base de datos institucional y se procederá a realizar una exploración de los mismos, procurando identificando los posibles problemas de calidad y buscando patrones iniciales, para este proceso utilizaremos herramientas como SQLyog que nos permitirá extraer la data y Python con el cual vamos a realizar un análisis estadístico simple de la data.

La tercera etapa según la metodología es la “Preparación de los datos”, una vez que la información ha sido revisada y comprendida se debe seleccionar, limpiar y transformar los datos para que estén listos para la siguiente etapa que es el modelado, en este paso si es necesario se puede integrar la información con otras fuentes de datos.

La cuarta etapa que debemos desarrollar es el “Modelado”, en esta fase vamos a aplicar técnicas de modelado a los datos que ya fueron preparados, aquí es donde todo el trabajo realizado anteriormente comienza a tener sentido, se crearán y evaluarán varios modelos basados en técnicas de machine learning como la regresión y clasificación, para evaluar el modelo será necesario dividir los datos en conjuntos de entrenamiento, prueba.

La siguiente etapa del proyecto es la “Evaluación”, en esta fase vamos a poner a prueba el modelo analizando sus métricas de desempeño como la precisión, sensibilidad y especificidad de los resultados obtenidos, debemos verificar que tan confiable es el modelo, dependiendo del resultado obtenido en esta fase puede ser necesario realizar una revisión de las fases anteriores para ajustar el modelo.

Para finalizar nuestro proyecto nos queda la etapa del “Despliegue”, en esta fase se realizará una presentación para comunicar los hallazgos de forma clara y no técnica, se utilizarán gráficos

y tablas que permitan entender los resultados que se han obtenido y explicar cómo pueden beneficiar a la institución.

Siguiendo cada una de las fases de la metodología CRISP-DM vamos a tener un enfoque estructurado para el desarrollo de nuestro proyecto, esta metodología es flexible y nos permite realizar revisiones y ajustes en cada una de las fases.

CAPÍTULO IV: RESULTADOS

4. Aplicación de las técnicas de minerías de datos para la creación de un modelo predictivo

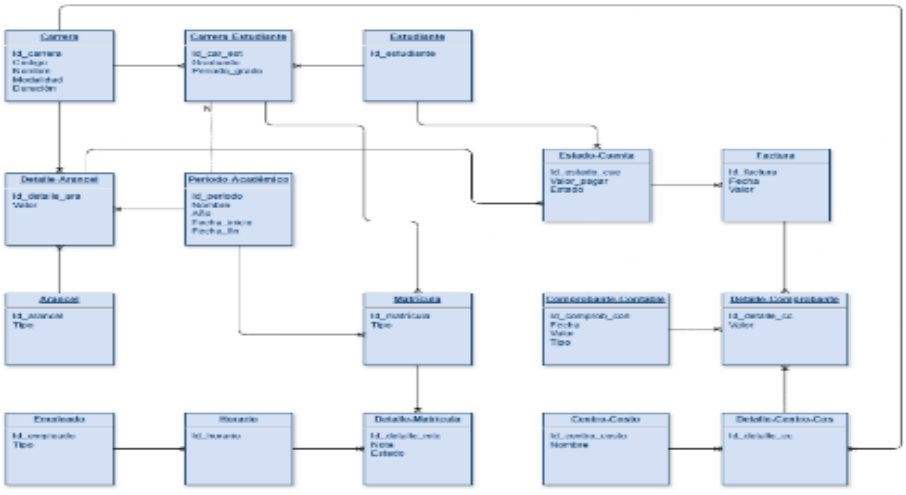
4.1. Comprensión del negocio

La información con la cual se va a desarrollar el proyecto se encuentra almacenada en la base de datos de la institución, el Instituto Cordillera utiliza MariaDB como motor de base de datos, esta será nuestra fuente de información ya que contiene toda la información académica y financiera necesaria para nuestro proyecto.

A continuación, podemos observar la Figura 1, que muestra una parte del Modelo Entidad – Relación de la base de datos del ERP institucional, aquí podemos ver las tablas y campos de donde se va a obtener la data para el desarrollo de nuestro proyecto.

Figura 1

Modelo de datos de la información académica y financiera de la institución



Nota. Parte del modelo ER del ERP institucional. Fuente: Elaboración propia.

En esta fase vamos a realizará el análisis de los ingresos y egresos de cada carrera tomando en cuenta variables como el número de estudiantes matriculados, los costos administrativos, los salarios del personal docente y otros gastos operativos.

4.2. Comprensión de los datos

Los datos disponibles en la institución incluyen registros de alumnos, docentes y personal administrativo, existen datos financieros y académicos, es importante entender la estructura y el contenido de estos datos para su correcto análisis, para el desarrollo de nuestro proyecto se van a utilizar las siguientes variables:

Año: Periodo al que corresponden los registros (formato AAAA), se obtiene de la fecha inicial del periodo académico.

Periodo académico: No indica el nombre del periodo académico, incluye mes - año de inicio y fin.

Código de la carrera: Es un identificador único asignado a cada carrera.

Nombre de la carrera: Esta es la denominación oficial de la carrera.

Modalidad: Corresponde a las formas en las que los estudiantes pueden cursar la carrera (presencial, hibrida o en línea).

Duración de la carrera: Número de niveles (semestres) que el estudiante debe cursar para terminar la carrera.

Número de estudiantes matriculados: Número total de estudiantes que ingresar a la carrera por periodo académico.

Porcentaje de deserción: Proporción de estudiantes que abandonaron la carrera sin completarla, para obtener esta información se tomó en cuenta a los estudiantes que no tienen registrado una nota en uno o más aportes de cada materia.

Porcentaje de alumnos graduados: Proporción de estudiantes que terminaron exitosamente la carrera, se obtiene dividiendo el número de graduados para el número de matriculados por cada periodo académico, se debe tomar en cuenta que existen estudiantes que se gradúan en otro periodo porque no cumplen con los requisitos necesarios.

Ingresos por matrícula: Monto total facturado por concepto de matrículas y aranceles en el período académico por carrera.

Otros ingresos: Otros ingresos asociados a la carrera que se han facturado a los estudiantes (ejemplo: trámites, certificados, recargos, titulación).

Costos docentes: Valor asociado al pago de sueldos y beneficios de los docentes, para obtener esta información se tuvo que realizar una clasificación auxiliar ya que el área financiera considera a las carreras de forma general (sistemas, diseño, farmacias, etc.), mientras que el área académica la clasificación es más detallada por ejemplo existen las carreras tecnológicas y las tecnológicas universitarias y también existen diferentes modalidades de estudio. Para este proceso se obtuvo el detalle de los costos de docentes por periodo y se los dividió según las carreras vigentes en ese periodo, en el caso de tener más carreras similares se dividió el total de los costos para el número de matriculados y en base a este valor se realizó la asignación, el rango de fechas para obtener estar información está dado por la fecha inicial y final de periodo.

Costos administrativos y de servicios: Valor relacionado con el pago de sueldos del personal administrativo (secretarias, prácticas preprofesionales, vinculación), de servicios y

mantenimiento, el departamento financiero maneja estos valores de forma global, en este caso para obtener este valor se obtuvo el total de los costos del personal administrativo y de servicios y se distribuyó según el número de matriculados en cada carrera y periodo, el rango de fechas para obtener estar información está dado por la fecha inicial y final de periodo.

Costos varios: Otros gastos operativos no contemplados en las categorías anteriores (servicios básicos, equipos, marketing, suministros, etc.), estos gastos también se manejan de forma global, en este caso para obtener este valor se obtuvo el total de los costos varios y se distribuyó según el número de estudiantes matriculados en cada carrera y periodo, el rango de fechas para obtener estar información está dado por la fecha inicial y final de periodo

Porcentaje de utilidad: Este valor se obtiene de sacar la diferencia entre los ingresos totales y costos totales por carrera y periodo académico y se nos indica el porcentaje de ganancia o pérdida de la carrera.

Rentable: Esta es una variable categórica que indica si la carrera ha sido rentable o no en el periodo de tiempo analizado, los datos que almacena son de tipo numérico (rentable = 1, no rentable = 0), el departamento financiero de la institución considera que una carrera es rentable si obtiene un porcentaje igual o mayor al 10%.

4.3. Recopilación de los datos iniciales

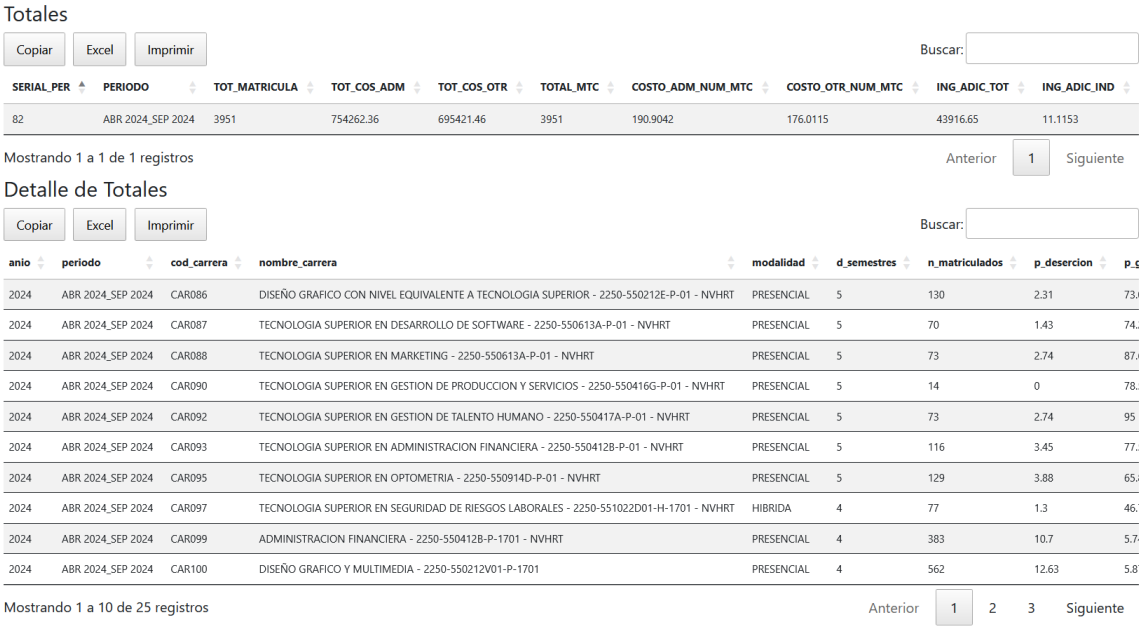
Para este estudio se recopilarán los datos académicos y financieros de los últimos diez años, está información se encuentra almacenada en la base de datos del ERP (Enterprise Resource Planning) institucional (MariaDB).

Para obtener la data fue necesario desarrollar un reporte que permita integrar la información de matriculados, graduados, deserciones. ingresos y egresos de cada periodo académico y carrera,

además de realizar los cálculos de los porcentajes de graduación, deserción, utilidad y verificar que carrera fue o no rentable en ese periodo, en la Figura 2 se puede observar el formato del reporte generado.

Figura 2

Reporte de información académica y financiera de la institución.



Totales

Copiar Excel Imprimir Buscar:

SERIAL_PER	PERIODO	TOT_MATRICULA	TOT_COS_ADM	TOT_COS_OTR	TOTAL_MTC	COSTO_ADM_NUM_MTC	COSTO_OTR_NUM_MTC	ING_ADIC_TOT	ING_ADIC_IND
82	ABR 2024_SEP 2024	3951	754262.36	695421.46	3951	190.9042	176.0115	43916.65	11.1153

Mostrando 1 a 1 de 1 registros Anterior 1 Siguiente

Detalle de Totales

Copiar Excel Imprimir Buscar:

anio	periodo	cod_carrera	nombre_carrera	modalidad	d_semestres	n_matriculados	p_desercion	p_g
2024	ABR 2024_SEP 2024	CAR086	DISEÑO GRAFICO CON NIVEL EQUIVALENTE A TECNOLOGIA SUPERIOR - 2250-550212E-P-01 - NVHRT	PRESENCIAL	5	130	2.31	73.
2024	ABR 2024_SEP 2024	CAR087	TECNOLOGIA SUPERIOR EN DESARROLLO DE SOFTWARE - 2250-550613A-P-01 - NVHRT	PRESENCIAL	5	70	1.43	74.
2024	ABR 2024_SEP 2024	CAR088	TECNOLOGIA SUPERIOR EN MARKETING - 2250-550613A-P-01 - NVHRT	PRESENCIAL	5	73	2.74	87.
2024	ABR 2024_SEP 2024	CAR090	TECNOLOGIA SUPERIOR EN GESTION DE PRODUCCION Y SERVICIOS - 2250-550416G-P-01 - NVHRT	PRESENCIAL	5	14	0	78.
2024	ABR 2024_SEP 2024	CAR092	TECNOLOGIA SUPERIOR EN GESTION DE TALENTO HUMANO - 2250-550417A-P-01 - NVHRT	PRESENCIAL	5	73	2.74	95
2024	ABR 2024_SEP 2024	CAR093	TECNOLOGIA SUPERIOR EN ADMINISTRACION FINANCIERA - 2250-550412B-P-01 - NVHRT	PRESENCIAL	5	116	3.45	77.
2024	ABR 2024_SEP 2024	CAR095	TECNOLOGIA SUPERIOR EN OPTOMETRIA - 2250-550914D-P-01 - NVHRT	PRESENCIAL	5	129	3.88	65.
2024	ABR 2024_SEP 2024	CAR097	TECNOLOGIA SUPERIOR EN SEGURIDAD DE RIESGOS LABORALES - 2250-551022D01-H-1701 - NVHRT	HIBRIDA	4	77	1.3	46.
2024	ABR 2024_SEP 2024	CAR099	ADMINISTRACION FINANCIERA - 2250-550412B-P-1701 - NVHRT	PRESENCIAL	4	383	10.7	5.7
2024	ABR 2024_SEP 2024	CAR100	DISEÑO GRAFICO Y MULTIMEDIA - 2250-550212V01-P-1701	PRESENCIAL	4	562	12.63	5.8

Mostrando 1 a 10 de 25 registros Anterior 1 2 3 Siguiente

Nota. Este reporte nos permite integrar la información necesaria para nuestro estudio. Fuente: Elaboración propia.

Para la creación del reporte (ver Figura 1), se dividió la información en dos partes, primero se obtuvo la información académica desde las siguientes tablas:

- Carrera
- Carrera-Estudiante
- Periodo-Académico
- Matricula
- Detalle-Matrícula

- Horario
- Empleado (Docente)

Mediante consultas y relacionando estas tablas se realizó sentencias SQL a la base de datos desde el reporte (ver Figura 1) para obtener la siguiente información:

- El año que se obtiene de la tabla Periodo-Académico.
- El código de la carrera que se obtiene de la tabla Carrera.
- El nombre de la carrera que se obtiene de la tabla Carrera.
- La modalidad de estudio que se obtiene de la tabla Carrera.
- La duración en semestres que se obtiene de la tabla Carrera.
- El número de matriculados por periodo y carrera que obtiene de las tablas Matricula, Periodo-Académico, Carrera-Estudiante y Carrera.
- El porcentaje de estudiantes graduados: Cabe indicar que los estudiantes no siempre se titulan inmediatamente después de terminar sus estudios y este valor representa el número de estudiantes que se registró en la SENESCYT es ese periodo académico, esto se puede obtener de las tablas Estudiante-Carrera y Periodo-Académico.
- El porcentaje de deserción: Para realizar el cálculo de este dato primero se obtuvo el número de estudiantes que no culminaron el periodo académico, para esto se consideró a aquellos alumnos que no tiene registrada la nota del tercer parcial, una vez obtenido este dato podemos calcular el porcentaje en función del número de matriculados, esta información se obtiene de las tablas Matrícula, Detalle-Matricula, Horario, Periodo-Académico.
- La distribución de los docentes en cada carrera que se puede obtener de las tablas Horario y Empleado.

Para la segunda parte del desarrollo del reporte (ver Figura 1) se obtuvo la información financiera de ingresos y egresos de las siguientes tablas:

- Estado-Cuenta
- Factura
- Comprobante-Contable
- Centro-Costo
- Detalle-Comprobante
- Detalle-Centro-Cos

Se realizaron consultas a la base de datos utilizando SQL para obtener la información necesaria para nuestro proyecto, la información obtenida se muestra en el reporte, a continuación, se explica que información se obtuvo:

El valor total de ingresos por matriculas: Este dato se obtiene de las tablas Estado-Cuenta y Factura.

El valor total de otros ingresos: Este dato también se obtiene de las tablas Estado-Cuenta y Factura.

Costo de docencia: Para obtener este dato se realizó una clasificación auxiliar por carrera ya que la información histórica no tenía la misma estructura que la data más actual, esta información se encuentra en las tablas Comprobante-Contable, Detalle-Comprobante, Centro-Costo, Detalle_Centro-Cos, Carrera y Periodo-Académico.

Costo del personal administrativo y de servicios: Se debe tomar en cuenta que esta información no está clasificada por carrera, para asignar estos valores se lo realizó en proporción al número de estudiantes matriculados, esta información se encuentra en las tablas Comprobante-

Contable, Detalle-Comprobante, Centro-Costo, Detalle_Centro-Cos, Carrera y Periodo-Académico.

Costos varios: Al igual que el punto anterior esta información tampoco está dividida por carrera, entonces para asignar estos valores se lo realizó en proporción al número de estudiantes matriculados, esta información se encuentra en las tablas Comprobante-Contable, Detalle-Comprobante, Centro-Costo, Detalle_Centro-Cos, Carrera y Periodo-Académico.

Porcentaje de utilidad: Para calcular este dato primero necesitamos conocer el valor de la utilidad, esto lo obtenemos restando el total de ingresos menos el total de los gastos, ahora que conocemos la utilidad la dividimos para el total de los ingresos y lo multiplicamos por 100 para expresarlo en porcentaje.

Rentable: Para indicar si la carrera fue rentable o no en ese periodo académico verificamos que el porcentaje de utilidad sea mayor o igual a 10, en este caso lo representamos con el 1 (uno) y caso contrario será igual a 0 (cero).

4.4. Descripción de los datos

Se ha obtenido un total de 288 registros distribuidos en 19 periodos académicos y 10 años, a continuación, se explican los datos que vamos a utilizar para la creación de nuestro modelo predictivo y son los siguientes:

anio: Esta es una variable de tipo numérica y los datos que almacena están entre el año 2015 y el 2024.

periodo: Esta variable es de tipo texto y almacena el nombre del periodo académico, como, por ejemplo: “OCT 2018_MAR 2019”, “ABR 2019_SEP 2019”.

cod_carrera: Esta es una variable de tipo texto que contiene un identificador de la carrera (CAR004, CAR006, CAR008).

nombre_carrera: Esta variable es de tipo texto y contiene el nombre completo de la carrera.

modalidad: Esta es una variable categórica que contiene la modalidad de estudio (PRESENCIAL, HIBRIDA, EN LÍNEA).

d_semestres: Esta variable es de tipo numérica y almacena el número de niveles que el estudiante debe aprobar para culminar la carrera, esta expresada en semestres y sus datos pueden estar entre 2 y 6.

n_matriculados: Esta variable es de tipo numérica y contiene el número de estudiantes matriculados en cada carrera y periodo académico.

p_deserción: Esta variable nos indica el porcentaje de estudiantes que no terminaron el periodo académico y deben volver a cursarlo, los datos que almacena son de tipo numérico.

p_graduados: Esta variable indica el porcentaje de alumnos que se registran como graduados en la SENESCYT en cada periodo académico, los datos que almacena son de tipo numérico

i_matrícula: Esa variable contiene los valores totalizados por carrera y periodo que los estudiantes cancelan por concepto de matrícula, los datos que almacena son de tipo numérico.

i_otros: Esta variable contiene los valores totalizados por carrera y periodo que los estudiantes cancelan por conceptos varios como tramites, certificados, titulación, etc. Los datos que almacena son de tipo numérico.

c_docentes: Esta variable indica el costo totalizado por carrera y periodo académico que la institución destina al pago de docentes, los datos que almacena son de tipo numérico.

c_adminis: Esta variable indica el costo totalizado por carrera y periodo académico que la institución destina al pago del personal administrativo y de servicios, los datos que almacena son de tipo numérico.

c_otros: Esta variable indica el costo totalizado por carrera y periodo académico que la institución destina a los pagos como servicios básicos, mantenimiento, marketing, suministros, etc. Los datos que almacena son de tipo numérico.

p_utilidad: Esta variable indica el porcentaje de utilidad de la carrera en cada periodo académico, los datos que almacena son de tipo numérica.

rentable: Esta es una variable categórica que indica si la carrea es rentable o no, puede contener valores como 0 (cero) o 1 (uno), los datos que almacena son de tipo numérico.

4.5. Exploración de los datos

En esta etapa se utilizará Python como herramienta de apoyo para realizar la exploración de los datos, esta herramienta nos permitirá ejecutar estadísticos sobre la data disponible y crear gráficos para entender de mejor forma los datos.

anio: La columna es de tipo entero y contiene un total de 288 registros, no se encontraron valores nulos y tiene 10 valores únicos que van desde 2015 hasta 2024.

periodo: La columna es de tipo texto y contiene 288 registros en total, no se encontró valores nulos y tiene 19 valores únicos, sus datos mantiene este formato “ABR 2024_SEP 2024”.

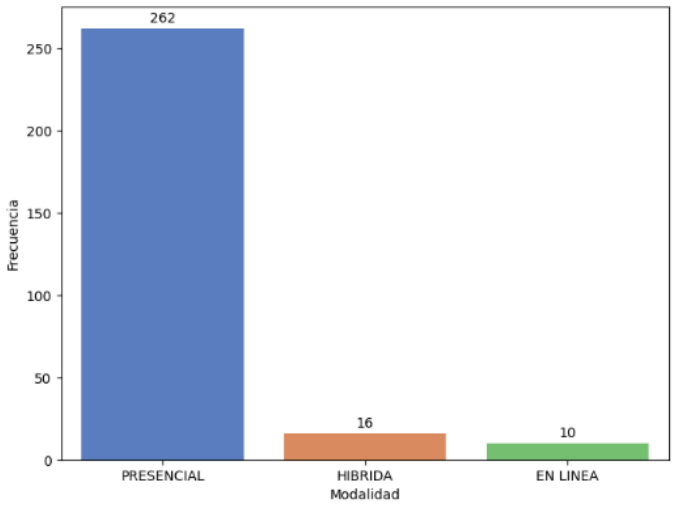
cod_carrera: La columna es de tipo texto y contiene un total de 288 registros, no se encontró valores nulos y tiene 40 valores únicos, sus datos mantiene este formato “CAR006, CAR115”.

nombre_carrera: La columna es de tipo texto y contiene 288 registros en total, no se encontró valores nulos y tiene 40 valores únicos.

modalidad: Esta columna es de tipo texto y contiene en total 288 registros, no se encontraron valores nulos y tiene 3 valores únicos, como se puede observar en la Figura 3, la modalidad Presencial es la predominante seguida por la Hibrida y finalmente En Línea, se debe tomar en cuenta que estás dos últimas fueron creadas en la pandemia del COVID 19.

Figura 3

Frecuencia de Modalidades



Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

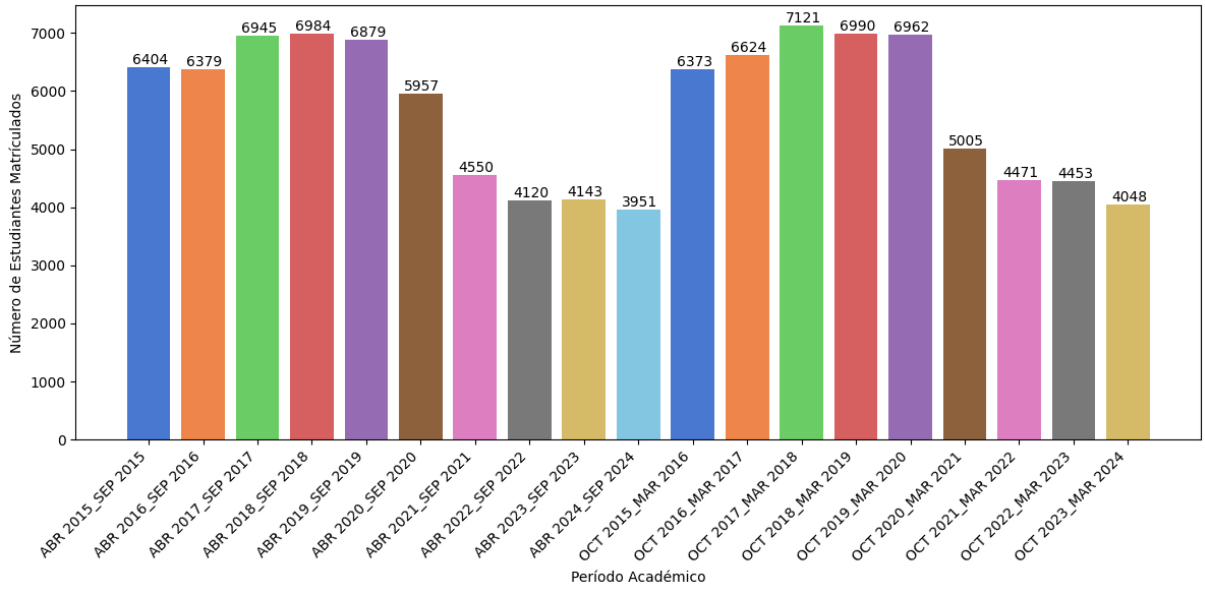
d_semestres: La columna es de tipo entero y contiene un total de 288 registros, no se encontraron valores nulos y tiene 4 valores únicos que van desde 2 hasta 6.

n_matriculados: Esta columna es de tipo entero y contiene 288 registros en total, no se encontró valores nulos y sus valores están entre 1 y 1248.

En la Figura 4, podemos observar el número total de estudiantes matriculados por cada periodo académico, se puede evidenciar que en los dos últimos años la institución a experimentado una reducción en el número de alumnos matriculados, Se debe tomar en cuenta que unos de los factores para esta reducción es la modificación de las políticas educativas, que redujeron el tiempo para culminar las carreras técnicas y tecnológicas. Otro factor que se debe considerar es la situación económica del país y el nivel de desempleo existente, estos factores afectan de forma negativa en la cantidad de estudiantes que ingresan a la institución.

Figura 4

Número de Estudiantes Matriculados por Período Académico



Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

p_deserción: La columna es de tipo flotante y contiene un total de 288 registros, no se encontraron valores nulos, pero si tenemos registros con 0 que se considera validos ya que nos indica que en dicha carrera no existió deserción. También encontramos registros con valores de 100 que nos indican que esa carrera se quedó sin estudiantes, estos casos se deben analizar a profundidad para decidir si se mantienen los registros o se los puede eliminar ya que no aportan valor a nuestro estudio.

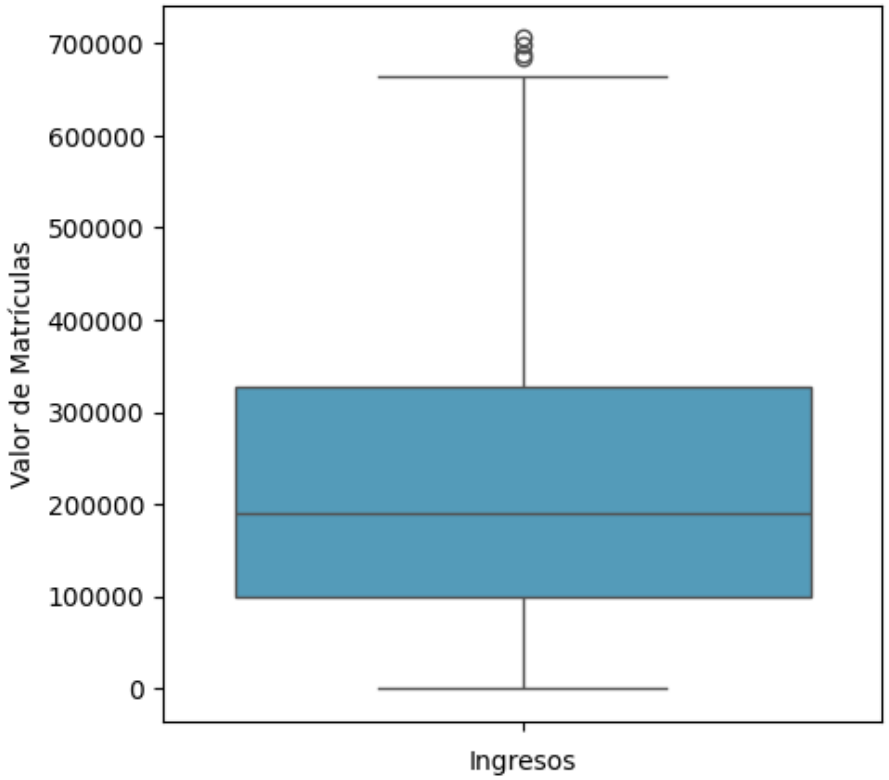
p_graduados: Esta columna es de tipo flotante y contiene 288 registros en total, no se encontró valores nulos, pero si registros con 0, estos valoren se considera validos ya que nos indica que en esta carrera y periodo no se registraron alumnos graduados, el valor máximo es de 95.96.

Hay que considerar que el registro de alumnos graduados se lo hace en el periodo académico en curso como referencia, se puede dar el caso que sea un alumno egresado que terminó sus estudios en periodos anteriores, pero está completando su proceso de titulación en el periodo actual.

i_matrícula: La columna es de tipo flotante y contiene un total de 288 registros, no se encontraron valores nulos, como se puede ver en la Figura 5, el 50% de los datos está entre aproximadamente 100000 y 300000, la mayoría de los datos están dentro del rango de 0 a 650000, también podemos observar que existen valores atípicos (puntos encima de la línea superior) indicando que algunos valores de matrícula están significativamente por encima del resto, estos datos se deben analizar para decidir si hay que incluirlos o no en nuestro análisis.

Figura 5

Ingreso por Concepto de Matrículas

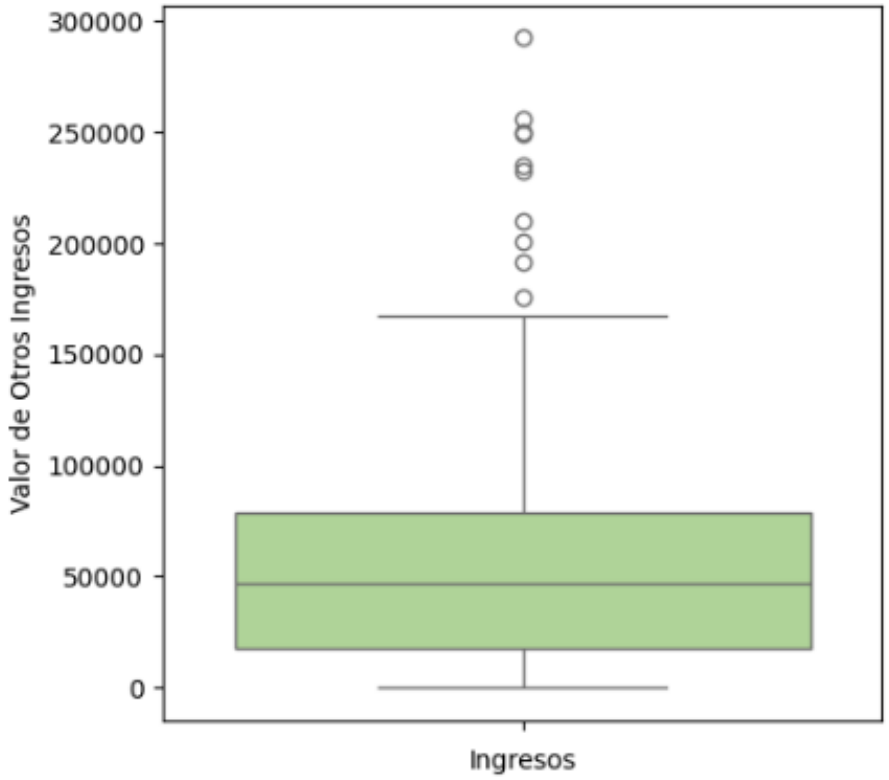


Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

i_otros: Esta columna es de tipo flotante y contiene un total de 288 registros, no se encontraron valores nulos, como se puede ver en la Figura 6, el 50% de los datos está entre aproximadamente 10000 y 70000, la mayoría de los datos están por debajo del umbral de los 175000. Se observan varios outliers por encima de 175000 y el mayor valor atípico ronda los 300000, lo que nos esa indica que hay casos de ingresos muy altos en comparación a la mayoría de los datos, estos valores atípicos se deben analizar para decidir si hay que incluirlos o no en nuestro análisis.

Figura 6

Otros Ingresos

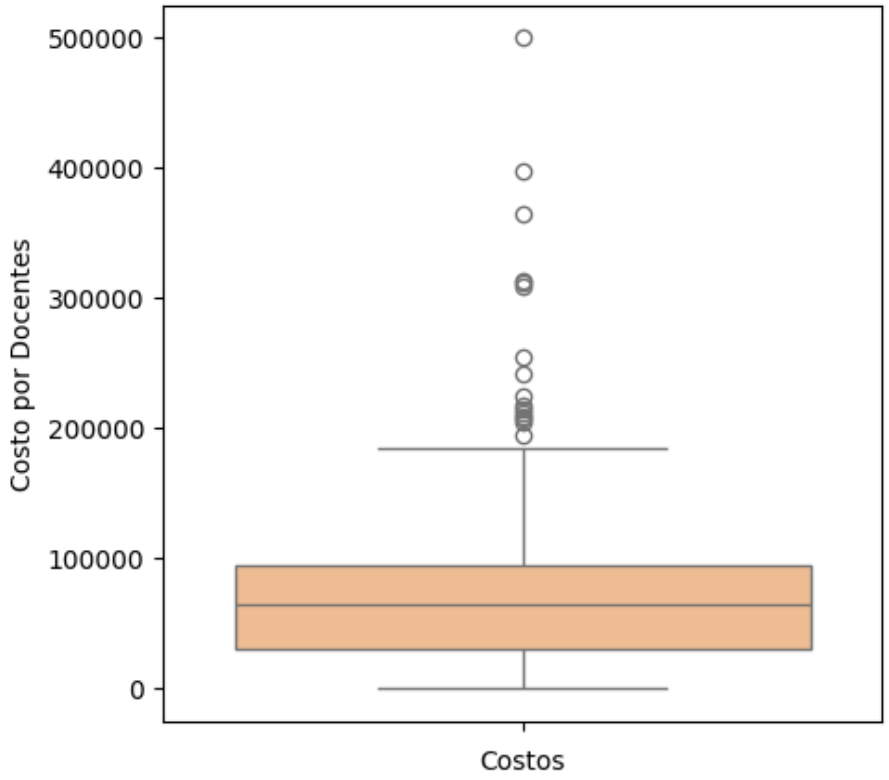


Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

c_docentes: Esta columna es de tipo flotante y contiene un total de 288 registros, no se encontraron valores nulos, pero existen registros con valor 0 que se debería eliminar ya que no son valores reales, como se puede ver en la Figura 7, el 50% de los datos está entre aproximadamente 40000 y 120000, el valor máximo sin tomar en cuenta los outliers está cerca de los 200000. Se observan valores atípicos por encima de los 200000 y que llegan a los 500000, estos valores atípicos se deben analizar para decidir si hay que incluirlos o no en nuestro análisis.

Figura 7

Costos del Personal Docente

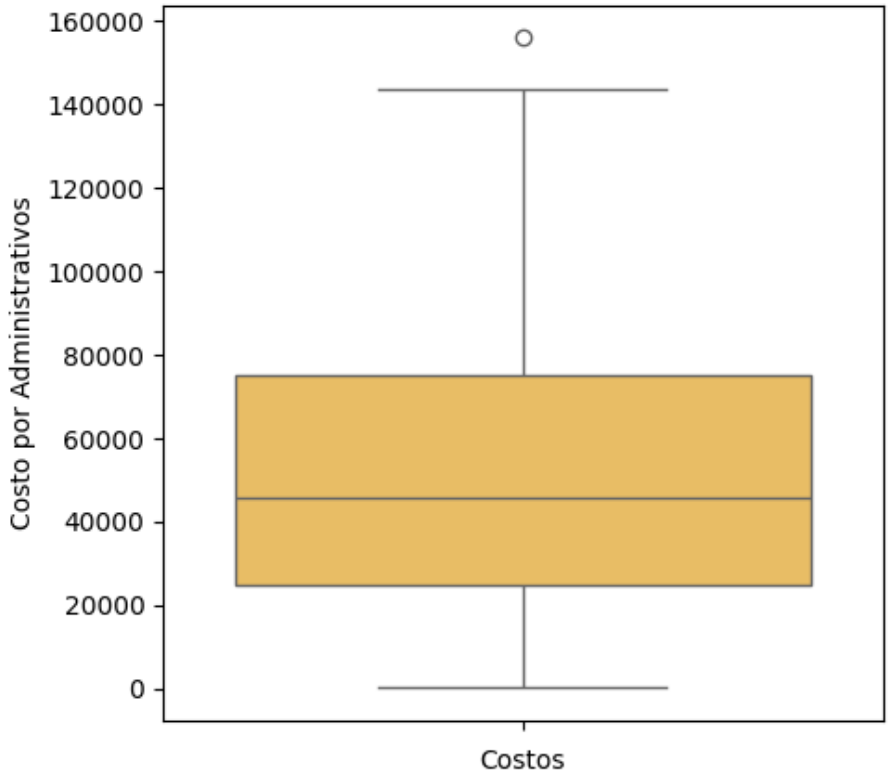


Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

c_adminis: Esta columna es de tipo flotante y contiene un total de 288 registros, no se encontraron valores nulos, como se puede ver en la Figura 8, el 50% de los datos está entre aproximadamente 20000 y 70000, el valor máximo son outliers está cerca de los 140000, también podemos observar que existen un outlier cerca de los 160000 indicando que un valor de estos costos es mayor que el resto, este dato se deben analizar para decidir si hay que incluirlos o no en nuestro análisis.

Figura 8

Costos del Personal Administrativo y de Servicios

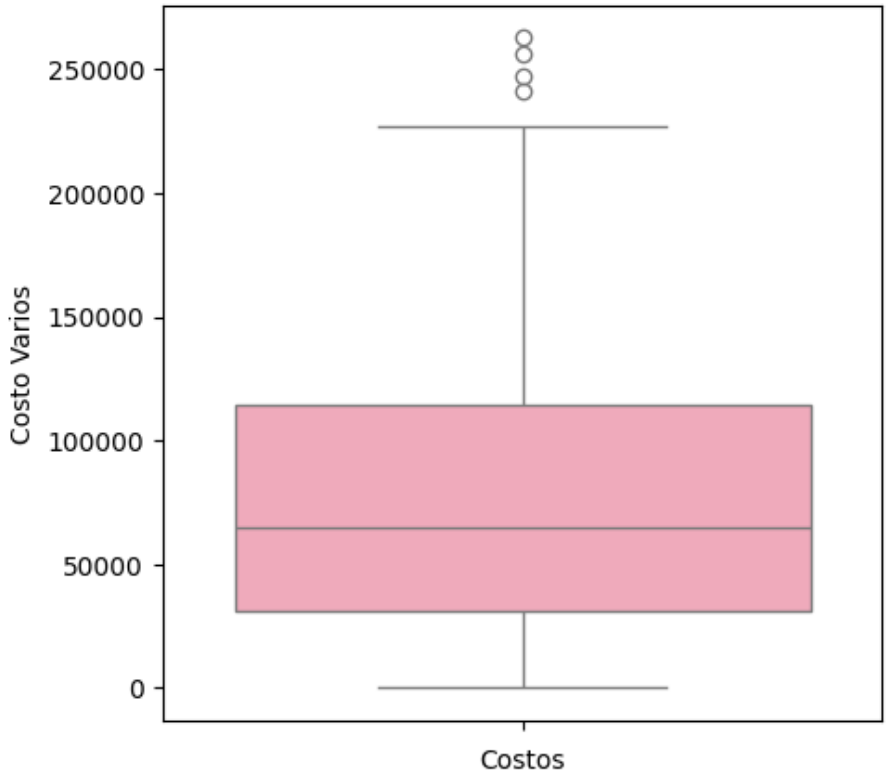


Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

c_otros: La columna es de tipo flotante y contiene un total de 288 registros, no se encontraron valores nulos, como se puede ver en la Figura 9, el 50% de los datos está entre 25000 y 120000 aproximadamente, los valores máximos sin tomar en cuenta los outliers se encuentra cerca de los 220000, también podemos observar que existen valores atípicos (puntos encima de la línea superior) entre 250000 y 270000, estos datos se deben analizar para decidir si hay que incluirlos o no en nuestro análisis.

Figura 9

Costos Varios



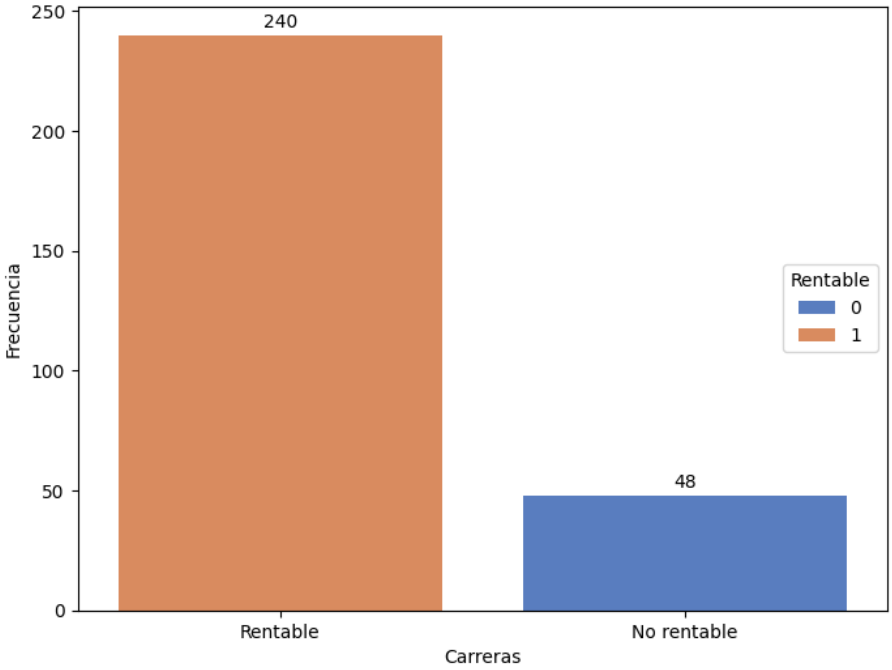
Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

p_utilidad: Esta columna es de tipo flotante y contiene un total de 288 registros, no se encontraron valores nulos, pero tenemos valores negativos ya que en algunos casos los gastos superan los ingresos, esto se puede dar cuando una carrera tiene un número bajo de estudiantes matriculados, también puede darse el caso que la carrera no se aperturó por falta de alumnos.

rentable: Esta columna es de tipo entero y contiene un total de 288 registros, no se encontraron valores nulos, sus valores pueden ser 1 o 0 que nos indica si una carrea se considera Rentable (1) o No rentable (0), como podemos observar en la Figura 10, la cantidad de careras que son rentables es considerablemente mayor a las no rentables.

Figura 10

Frecuencia de Carreras Rentables y No rentables



Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

4.6. Verificación de calidad de los datos

En función de los resultados de la exploración de datos podemos indicar que no existen valores nulos ni errores tipográficos o de formato en los datos, la data ha sido obtenida directamente desde la base de datos institucional y estos ya han sido tratados previamente.

Los datos se alimentan desde el ERP institucional y estos son validados desde su origen antes de ser almacenados en la base de datos, en este sentido podemos concluir que la data no tiene errores que afecten al desarrollo del proyecto.

También se debe considerar que al ser datos históricos ya han sido validado por las diferentes áreas de la institución como es el caso del departamento financiero que anualmente presenta sus balances y verifica la integridad de los datos.

4.7. Preparación y muestreo de los datos

4.7.1. Selección de datos

Dado el limitado número de registros que se obtuvieron de la base de datos institucional, nos vemos en la necesidad de utilizar toda la data, es importante aprovechar la mayor cantidad de variables disponibles para obtener los mejores resultados de nuestro modelo.

4.7.2. Limpieza de datos

Como se pudo ver en la fase de “Exploración de datos” no se encontraron valores nulos, sin embargo, lo que si se pudo observar en la data son variables con valores de 0, para el desarrollo de nuestro proyecto todas estás variables nos aportan información, como se puede ver en al Figura 11, uno de los campos con valor de cero es c_docentes, que representa el costo de los docentes, esto nos indica que al no existir un costo por docencia esa carrera en ese periodo no se abrió, esto se comprobó haciendo un conteo del número de alumnos matriculados que dio como resultado un valor menor que 10,

Figura 11

Columnas que tiene el valor 0

	Columna	Cantidad de ceros
0	p_desercion	9
1	p_graduados	73
2	c_docentes	2
3	rentable	48

Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

4.8. Ejecutar las técnicas del modelado

Para la fase de modelado vamos a utilizar 3 algoritmos de machine learning que por el tipo de información que se obtuvo de la institución son los que mejor se acoplan y nos permitirán predecir la sostenibilidad de las carreras.

Los algoritmos que se utilizaron para el desarrollo del proyecto son modelos supervisados que se nos permiten buscar patrones para realizar predicciones:

- Regresión logística
    - Modelo básico e ideal para problemas de clasificación binaria.
    - Funciona con pocos recursos computacionales.
    - Nos proporciona varios coeficientes que nos permiten evaluar el modelo.
- Random forest
    - Se utiliza con datos no lineales ya que captura relaciones complejas.
    - Ayuda a reducir el sobreajuste porque combina varios árboles de decisión.
    - Funciona muy bien con datos faltantes y outliers.
- Percepción multicapa (MLP)
    - Encuentra relaciones incluso cuando no son evidentes.
    - No se necesita definir reglas manuales ya que la red neuronal aprende por si sola las características más importantes.
    - Se utiliza en problema de análisis médico, proyección y análisis financiero.
- Regresión lineal múltiple
    - Nos ayuda a comprender como cada variable independiente (predictora) afecta a la variable dependiente (objetivo).
    - Es fácil de implementar y no requiere de muchos recursos computacionales.
    - Se la puede aplicar en diversas áreas como las finanzas, salud, etc.

A continuación, vamos a detallar el proceso de modelado y los resultados obtenidos por cada uno de estos modelos.

4.8.1. Regresión logística

Este algoritmo se lo utiliza en problemas de clasificación binaria, es decir cuando la variable dependiente (o resultado) tiene solo dos posibles valores de respuesta (Si o No, 1 o 0, Verdadero o Falso), este algoritmo se ajusta a nuestro proyecto ya que nuestro objetivo es predecir si una carrera es sostenible o no.

4.8.1.1. Selección de variables

Una vez que hemos cargado la data en nuestra herramienta de análisis (Python) vamos a eliminar las columnas de tipo texto (anio, periodo, cod_carrera, nombre carrera) que no que aportan a nuestro análisis, esto se debe realizar ya que algoritmo de regresión logística solo trabaja con datos numéricos.

La columna modalidad también es de tipo texto, pero esta la vamos a convertir en formato numérico utilizando la técnica One-Hot Encoding.

Finalmente, ya se han eliminado las variables de tipo texto y se ha convertido la modalidad estás son las variables de tipo numérico con las que vamos a trabajar:

- d_semestres (Independiente)
- p_desercion (Independiente)
- p_graduados (Independiente)

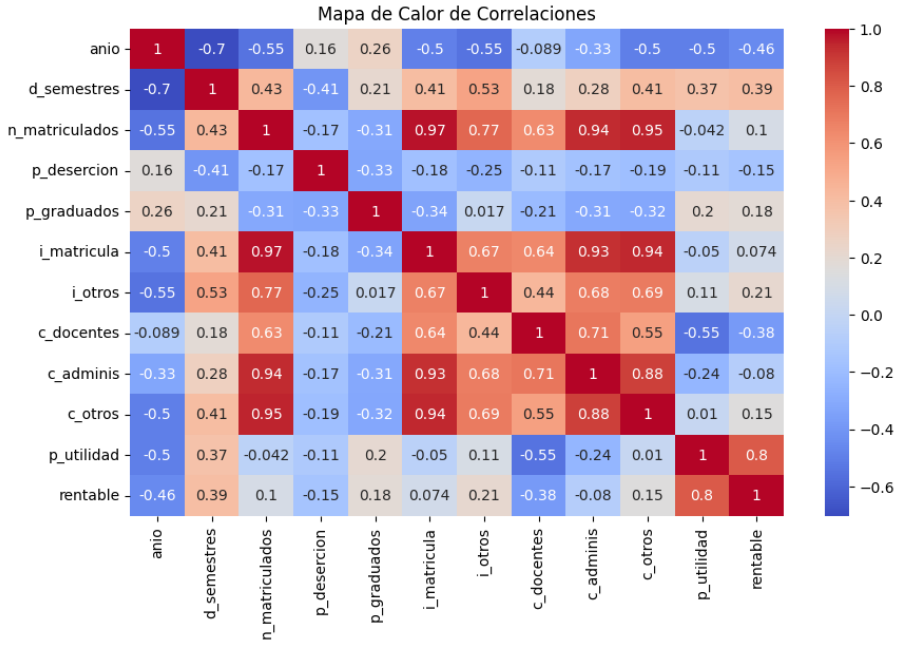
- c_docentes (Independiente)
- p_utilidad (Independiente)
- ingresos_totales (Independiente)
- mod_en_linea (Independiente)
- mod_hibrida (Independiente)
- mod_presencial (Independiente)
- rentable (Dependiente)

4.8.1.2. Construcción de nuevos datos

Se elaboro un mapa de calor que nos permita visualizar la correlación que existe entre las variables de mi dataset, como se puede ver en la Figura 12, existe una alta correlación (multicolinealidad) entre las variables de ingresos, costos y el número de matriculados, por lo cual se decidió sumar las columnas i_matricula e i_otros en la columan total_ingresos, también se eliminó las columnas de c_adminis y c_otros,

Figura 12

Mapa de calor de correlación de variables



Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

4.8.1.3. Dividir los datos en entrenamiento y prueba

Es necesario dividir los datos para evitar posibles sobreajustes ya que so utilizamos todos los datos el modelo puede aprender patrones específicos y no reglas generales que es nuestro objetivo.

La división de los datos también nos ayuda a detectar posibles problemas, si el modelo tiene un excelente resultado con la data de entrenamiento, pero un mal desempeño con los datos de prueba quiere decir que el modelo esta sobre ajustado (overfitting),

En nuestro caso vamos a dividir los datos en un porcentaje de 70% para entrenamiento y 30% para pruebas.

4.8.1.4. Balancear las clases

Una vez que se han divido los datos en entrenamiento y pruebas podemos ver que existe un desbalance de clases, existe un mayor número de muestras para la clase 1 (Rentable), esto puede provocar que nuestro modelo este sesgado hacia la clase mayoritaria.

Para mejorar el rendimiento de nuestro modelo vamos a utilizar la técnica de remuestreo utilizando la función SMOTE de Python, esta función nos permite generar nuevos datos para la clase minoritaria (No rentable), de esta forma tenemos balanceadas las clases.

En la Figura 13 se puede observar como estaban distribuidas las clases antes y después de aplicar SMOTE.

Figura 13

Balanceo de clases

```
Antes de SMOTE: Counter({1: 170, 0: 31})
Después de SMOTE: Counter({0: 170, 1: 170})
```

Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

Otra recomendación es obtener más datos de la clase minoritaria sin embargo en nuestro caso esto no es posible.

4.8.1.5. Escalamiento de los datos

Como se pudo ver en la fase de exploración de datos tenemos variables numéricas con rangos muy distintos y también existen valores atípicos, para evitar que el modelo asigne más peso a estas variables se realizó el escalamiento de los datos, utilizamos la función “StandardScaler” de Python que convierte los datos en una distribución con media 0 y desviación estándar 1, de esta forma se reduce el impacto de los valores extremos.

4.8.1.6. Creación del modelo

Como se pudo detectar en las fases anteriores los datos presentan ciertos problemas como multicolinealidad y desbalance de clases, por lo cual vamos utilizar las técnicas de regularización como:

- Ridge: Agrega una penalización L2 que reduce el tamaño de los coeficientes, pero no elimina variables.
- Lasso: Agrega una penalización L1 que puede reducir los coeficientes hasta llegar a cero y eliminar variables que no aportan al modelo.

El uso de estas técnicas nos ayudara a mejorar el desempeño de nuestro modelo. Para la selección de los hiperparámetros del modelo vamos a utilizar la función GridSearchCV que nos ayuda a seleccionar los mejores valores para nuestro modelo.

En la Figura 14 se puede ver el valor que la función seleccionó para el modelo de Ridge y Lasso y también podemos ver la exactitud de los modelos creados.

Figura 14

Modelos de Ridge y Lasso

```
Mejor valor de C para Ridge: 100
Mejor exactitud para Ridge: 0.9941176470588236

Mejor valor de C para Lasso: 10
Mejor exactitud para Lasso: 0.9941176470588236

Exactitud en el conjunto de prueba para Ridge: 0.9770114942528736
Exactitud en el conjunto de prueba para Lasso: 0.9885057471264368
```

Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

4.8.1.7. Evaluación del modelo

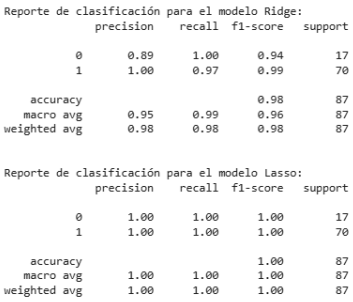
Una vez generados nuestros modelos vamos a obtener un reporte de los resultados que arroja, con esta información vamos a evaluar nuestros modelos, para la evaluación utilizaremos las siguientes métricas:

- **Exactitud (Accuracy):** Nos indica que tan bien clasifica el modelo.
- **Precisión (Precision):** Nos indica que tan confiables son las predicciones para la clase rentable, su fórmula es VP / (VP+FP), como ejemplo podemos tomar que, si se predice 10 carrera rentables, pero solo 7 realmente lo son la precisión del modelo es 7/10 = 70%.
- **Sensibilidad (Recall):** Nos indica cuantos valores de la clase rentable reales detecto correctamente el modelo, su fórmula es: VP / (VP+FN), como ejemplo podríamos decir que si hay 10 carreras rentables y el modelo detecta a 8, la sensibilidad es 8/10 = 80%.
- **F1 Score:** Esta medida nos ayuda cuando tenemos datos desbalanceados ya que la exactitud podría no ser real, representa un balance entre precisión y sensibilidad
- **Matriz de Confusión:** Nos muestra una tabla de los aciertos y errores que ha tenido el modelo.
- **AUC-ROC:** Nos muestras de forma gráfica como se separan las clases del modelo.

En la Figura 15 podemos observar un reporte de las métricas obtenidas en cada modelo (Ridge y Lasso).

Figura 15

Resultados de los modelos con Ridge y Lasso



```
Reporte de clasificación para el modelo Ridge:
              precision    recall  f1-score   support

           0       0.89      1.00      0.94        17
           1       1.00      0.97      0.99        70

    accuracy                           0.98        87
   macro avg       0.95      0.99      0.96        87
weighted avg       0.98      0.98      0.98        87


Reporte de clasificación para el modelo Lasso:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       1.00      1.00      1.00        70

    accuracy                           1.00        87
   macro avg       1.00      1.00      1.00        87
weighted avg       1.00      1.00      1.00        87
```

Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

A continuación, en la Tabla 1 se muestra un resumen de las métricas obtenidas al entrenar nuestro modelo.

Tabla 1

Reporte conjunto de Ridge y Lasso

Métrica	**Ridge (Clase 0)**	**Ridge (Clase 1)**	**Lasso (Clase 0)**	**Lasso (Clase 1)**
Precision	0.89	1	1	1
Recall	1	0.97	1	1
F1-Score	0.94	0.99	1	1
Support	17	70	17	70

Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

Según los resultados obtenidos podemos indicar que el modelo de Lasso es mejor en general porque tiene una exactitud de 100%, sin embargo, en nuestro proyecto vamos a tomar el modelo de Ridge para nuestras predicciones ya que es más recomendado para datasets pequeños porque no elimina variables.

4.8.1.8. Matriz de confusión

En la Figura 16 podemos ver el resultado de la matriz de confusión de los dos modelos que nos muestran que Lasso tiene una exactitud del 100%, este resultado no es normal y puede existir un sobre ajuste.

Figura 16

Matriz de confusión de Ridge y Lasso

```
Ridge
[[17  0]
 [ 2 68]]
Exactitud: 97.70%

Lasso
[[17  0]
 [ 0 70]]
Exactitud: 100.00%
```

Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

4.8.1.9. Validación cruzada

Dado los resultados obtenidos vamos analizar de forma más profunda los modelos, para esto utilizaremos la técnica de validación crazada que nos permite dividir los datos en varias partes (folds) para entrenar y probar el modelo con la parte que queda, de esta forma obtenemos un promedio de los resultados que nos indica que tan bien está funcionando el modelo:

- Exactitud media del modelo Ridge con validación cruzada: 0.91
- Exactitud media del modelo Lasso con validación cruzada: 0.98

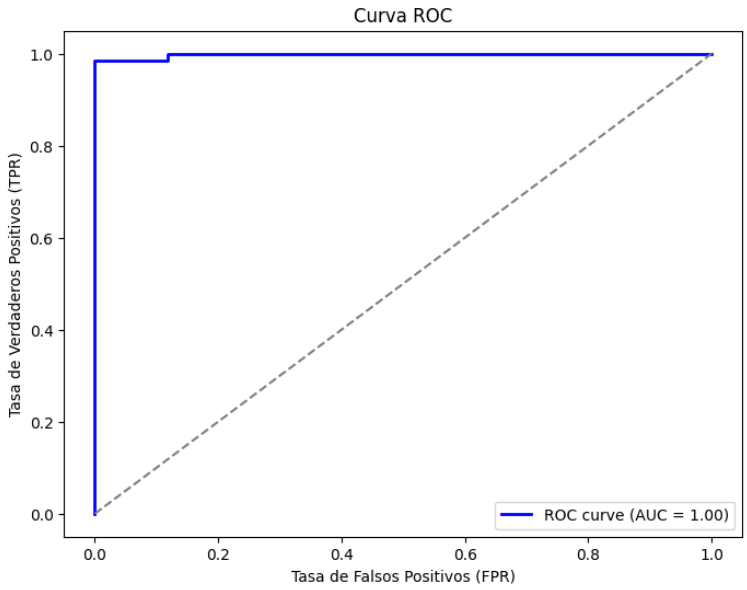
Estos resultados nos muestran que el modelo de Lasso tiene un mejor desempeño, pero esto puede ser el resultado de eliminar algunas variables, los que le da una mayor exactitud, pero esto también es signo de sobreajuste.

4.8.1.10. AUC-ROC (Área bajo la curva ROC)

Para este análisis se tomo el modelo de Ridge ya que con este modelo vamos a realizar las predicciones, como se puede observar en la Figura 17, nuestro modelo tiene valor de AUC de 1.00 que indica un rendimiento perfecto al distinguir los positivos y negativos, este valor nos indica que el modelo pude clasificar correctamente todos los casos.

Figura 17

Área bajo la curva de ROC



Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

4.8.2. Random Forest

Este es un algoritmo supervisado que se utiliza en problemas de clasificación y regresión, su funcionamiento se basa en la creación de varios árboles de decisión durante el entrenamiento del modelo que permiten mejorar la precisión del modelo reduciendo el sobreajuste, vamos a

utilizar este algoritmo ya que se ajusta a nuestros datos y a nuestro objetivo que es predecir si una carrera es sostenible o no.

4.8.2.1. Selección de variables

Al igual que para el modelo anterior es necesario eliminar las columnas de tipo texto (anio, periodo, cod_carrera, nombre_carrera) que no que aportan a nuestro análisis ya que al algoritmo random forest solo trabaja con datos numéricos, a diferencias del modelo anterior aquí también se va a eliminar las variables modalidad ya que no aportó mayor información en el análisis.

A continuación, esta es la lista de variables numéricas con las que vamos a trabajar aplicando el algoritmo de random forest:

- d_semestres (Independiente)
- p_desercion (Independiente)
- p_graduados (Independiente)
- c_docentes (Independiente)
- p_utilidad (Independiente)
- ingresos_totales (Independiente)
- rentable (Dependiente)

4.8.2.2. Construcción de nuevos datos

Como se pudo evidenciar en la Figura 12, existe una alta correlación entre las variables de ingresos, costos y número de matriculados por esta razón para este modelo vamos a eliminar la variable n_matriculados, y se va a sumar las columnas i_matricula e i_otros en la columna

total_ingresos, también vamos a sumar las columnas c_adminis y c_otros en una nueva llamada otros_costos, de esta forma se quiere reducir el sobreajuste en nuestro modelo.

4.8.2.3. Dividir los datos en entrenamiento y prueba

Para este caso de igual forma que para el modelo de regresión logística vamos a dividir los datos en un porcentaje de 70% para entrenamiento y 30% para pruebas.

4.8.2.4. Balancear las clases

Como ya vimos en el modelo anterior existe un desbalance en las clases por lo tanto es necesario balancearlas, para esto vamos a utilizar la técnica de remuestreo con la función SMOTE de Python, esta función como ya sabemos nos permite generar nuevos datos para la clase minoritaria (No rentable), de esta forma balanceamos las clases.

4.8.2.5. Escalamiento de los datos

Esta es una práctica que la vamos a realizar para evitar el sobreajuste de nuestro modelo, utilizaremos la función “StandardScaler” de Python que convierte los datos en una distribución con media 0 y desviación estándar 1, de esta forma se reduce el impacto de los valores extremos.

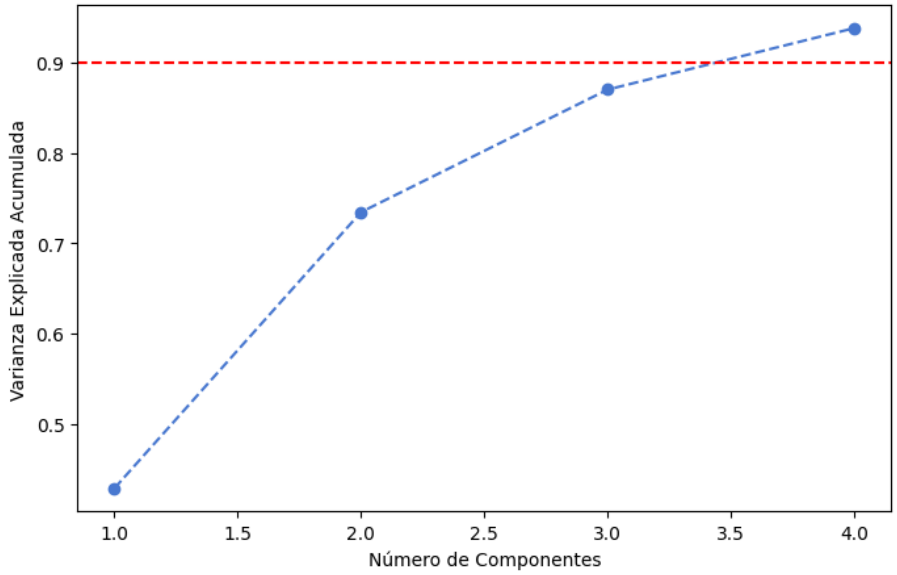
4.8.2.6. Análisis de componentes principales (PCA)

Esta es una técnica que nos permite transformas las variables de nuestro dataset en componente principales que capturan la mayor cantidad de información (varianza) utilizando menos dimensiones, esta técnica nos ayuda a reducir la redundancia de información y mantener solo la más importante.

Como se puede ver en la Figura 18, con 4 componente se puede explicar mas del 90% de la varianza, por lo tanto, vamos a crear el modelo con este número de componentes.

Figura 18

Selección del número óptimo de componentes



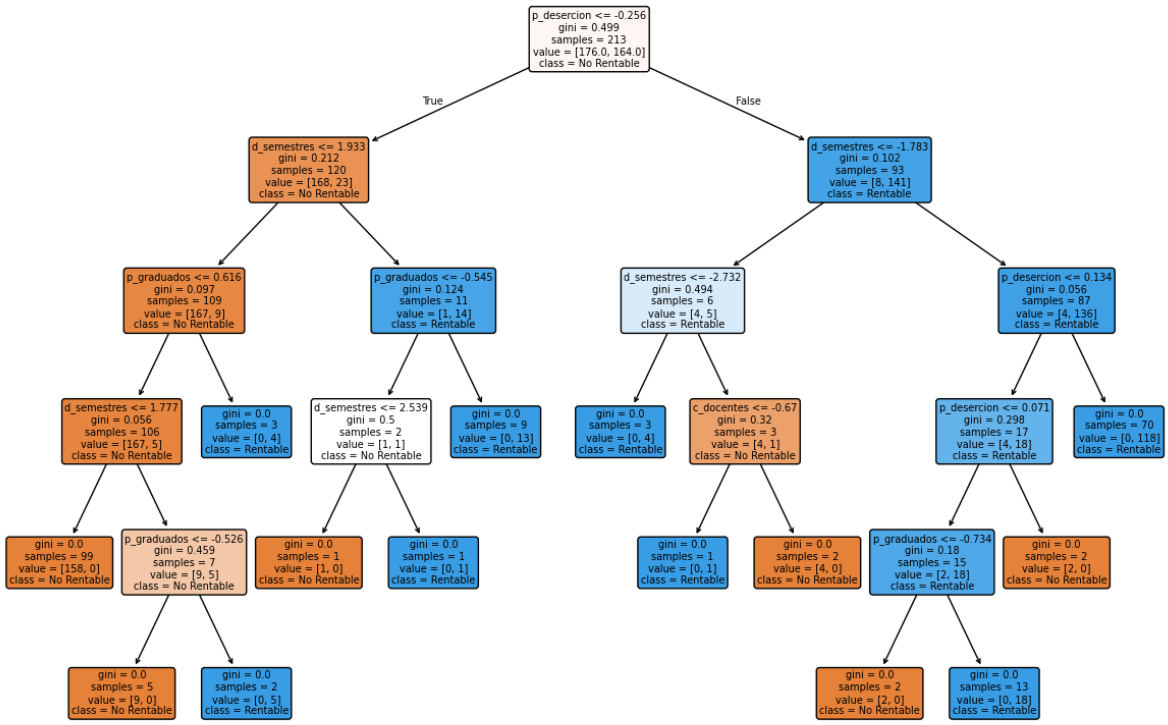
Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

4.8.2.7. Creación del modelo

Luego de haber realizado el remuestreo, escalamiento y selección de componente principales estamos listos para crear nuestro modelo de random forest y entrenarlo para analizar los resultados que nos permitan evaluar el modelo, en la Figura 19 podemos observar el primer árbol de nuestro bosque.

Figura 19

Primer árbol del bosque



Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

4.8.2.8. Evaluación de modelo

Una ves que tenemos generado el modelo vamos a analizar los estadísticos que nos permitan conocer cuál es el desempeño que tiene nuestro modelo, en la Figura 19 podemos ver que tiene un 92% de exactitud en las predicciones totales.

En la Tabla 2 podemos observar el resumen de las métricas de resultado, que nos indican que el modelo tiene un buen rendimiento en general, sin embargo, se puede ver que existe un desbalance en las clases, el modelo es más preciso para la clase 1 que para la clase 0, esto afecta

el desempeño del modelo hacia la clase con menos muestras, antes de generar el modelo se realizó el remuestreo, escalamiento y selección de componente principales para intentar evitar estos problemas, sin embargo se están presentando, lo que nos queda para intentar mejorar el modelo es entrenarlo con más datos lo que por el momento no es posible.

Tabla 2

Reporte del modelo Random Forest

Métrica	**Clase 0**	**Clase 1**
Precision	0.86	0.93
Recall	0.71	0.97
F1-Score	0.77	0.95
Support	17	70

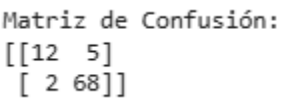
Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

4.8.2.9. Matriz de confusión

En la Figura 20 podemos ver el resultado de la matriz de confusión del modelo que nos muestran que es muy bueno detectando la clase 1 pero no lo es tanto para detectar la clase 0 que para nuestro caso es la no rentable.

Figura 20

Matriz de confusión de Random Forest



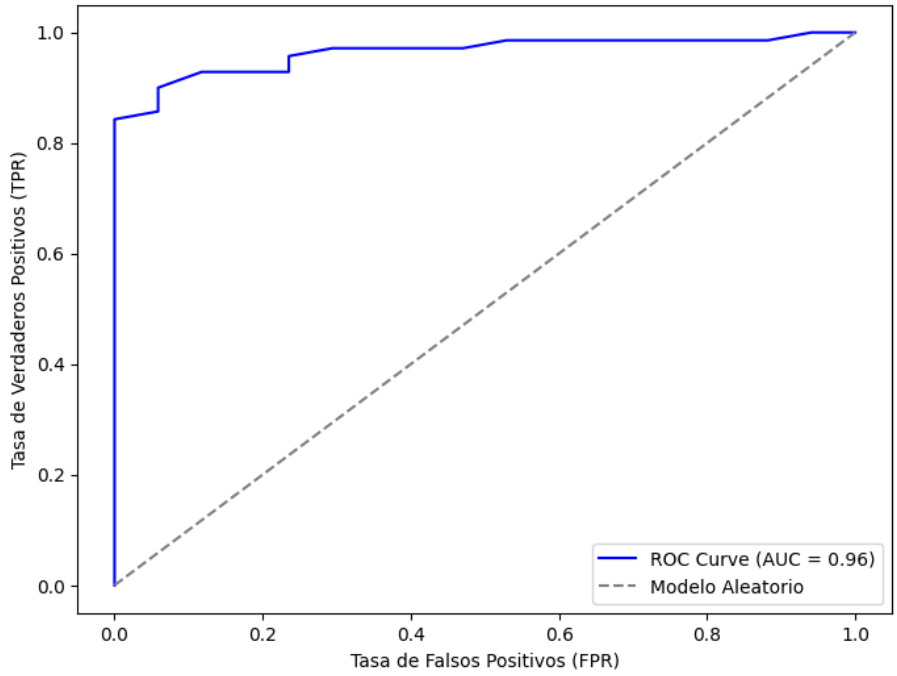
Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

4.8.2.10. AUC-ROC (Área bajo la curva ROC)

Como se puede observar en la Figura 21, nuestro modelo tiene valor de AUC de 0.96 que indica un rendimiento muy alto al distinguir los positivos y negativos, este valor nos indica que el modelo pude clasificar correctamente todos los casos.

Figura 21

Curva ROC para el modelo Random Forest



Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

4.8.3. Percepción multicapa (MLP)

Este es un modelo de red neuronal densa ya que todas las neuronas están conectadas entre sí, se lo puede utilizar para problemas clasificación binaria, diagnóstico de enfermedades, detección de fraudes, etc. Este modelo se ajusta a nuestro proyecto ya que nuestro objetivo es predecir si una carrera es sostenible o no.

4.8.3.1. Selección de variables

Para la creación de este modelo vamos a eliminar las columnas de tipo texto (anio, periodo, cod_carrera, nombre_carrera, modalidad) que no que aportan a nuestro análisis ya que este algoritmo solo trabaja con datos numéricos.

A continuación, esta es la lista de variables numéricas con las que vamos a trabajar aplicando el algoritmo MLP:

- d_semestres (Independiente)
- p_desercion (Independiente)
- p_graduados (Independiente)
- c_docentes (Independiente)
- c_adminis (Independiente)
- c_otros (Independiente)
- p_utilidad (Independiente)
- ingresos_totales (Independiente)
- rentable (Dependiente)

4.8.3.2. Construcción de nuevos datos

Este modelo no requiere definir manualmente las características ya que aprende cuales son las más importantes por sí solo, en este caso únicamente vamos a eliminar la variable n_matriculados, y se va a sumar las columnas i_matricula e i_otros en la columna total_ingresos.

4.8.3.3. Dividir los datos en entrenamiento y prueba

Para este caso de igual forma que en los modelos anteriores vamos a dividir los datos en un porcentaje de 70% para entrenamiento y 30% para pruebas.

4.8.3.4. Escalamiento de los datos

Esta es una buena práctica que la vamos a realizar para evitar el sobreajuste de nuestro modelo, utilizaremos la función “RobustScaler” de Python, que utiliza la media y el rango intercuartil para escalar los datos, este tipo de escalamiento de recomienda utilizar cuando existen muchos valores extremos.

4.8.3.5. Creación del modelo

Vamos a crear una red neuronal de tipo secuencial se llama así porque los datos fluyen de una capa a otra capa, esta red neuronal se compone de cuatro capas, una de entrada, dos capas ocultas y una salida.

La capa de entrada permite definir el número de variables que tiene el dataset en nuestro caso existen 8 columnas, esta capa le dice a la red neuronal cuantas variables debe recibir.

La primera capa oculta añade 16 neuronas y la función de activación ReLU que indica si la neurona debe activarse o no, esto ayuda acelerar el entrenamiento, esta capa sirve para transformar las variables de entrada en información más útil para la tarea.

La segunda capa oculta añade 8 neuronas más y la función de activación ReLU, esta capa ayuda a reducir la dimensionalidad de las características de la capa anterior, además de extraer patrones más complejos.

La capa de salida tiene una neurona y una función de activación sigmoide, se utiliza solo una ya que el modelo será utilizado para clasificación binaria.

Para el entrenamiento del modelo vamos a utilizar 20 épocas, cada una de estas es un ciclo que el modelo analiza nuestro conjunto de datos, este valor es el recomendado para iniciar y si es necesario se puede seguir aumentando mas épocas, como se puede ver en la Figura 22, en cada época se analiza la data y se obtiene un valor de exactitud (accuracy) que va mejorando en cada paso.

Figura 22

Épocas de la red neuronal

```
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9667 - loss: 0.1870 - val_accuracy: 0.9195 - val_loss: 0.2222
Epoch 15/20
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9746 - loss: 0.1488 - val_accuracy: 0.9080 - val_loss: 0.2090
Epoch 16/20
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9647 - loss: 0.1437 - val_accuracy: 0.9540 - val_loss: 0.1941
Epoch 17/20
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9831 - loss: 0.1212 - val_accuracy: 0.9425 - val_loss: 0.1844
Epoch 18/20
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9700 - loss: 0.1250 - val_accuracy: 0.9540 - val_loss: 0.1782
Epoch 19/20
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9658 - loss: 0.1235 - val_accuracy: 0.9425 - val_loss: 0.1715
Epoch 20/20
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9630 - loss: 0.1188 - val_accuracy: 0.9425 - val_loss: 0.1645
```

Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

4.8.3.6. Evaluación del modelo

Una vez que tenemos nuestro modelo generado es momento de evaluar las métricas en este caso el modelo presenta una precisión del 97.70% lo que hace que el modelo tenga un excelente rendimiento.

En la Tabla 3 podemos observar el resumen de las métricas del modelo, que nos indican que el modelo tiene un buen rendimiento en general, sin embargo, se puede ver que existe un

desbalance en las clases, el modelo es más preciso para la clase 1 que para la clase 0, esto afecta el desempeño del modelo hacia la clase con menos muestras.

Tabla 3

Reporte del modelo MLP

Métrica	**Clase 0**	**Clase 1**
Precision	0.87	0.95
Recall	0.82	0.97
F1-Score	0.84	0.96

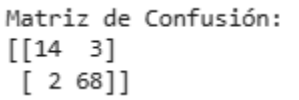
Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

4.8.3.7. Matriz de confusión

En la Figura 23 podemos ver el resultado de la matriz de confusión del modelo que nos muestran que es muy bueno detectando la clase 1 pero no lo es tanto para detectar la clase 0 que para nuestro caso es la no rentable.

Figura 23

Matriz de confusión de MLP



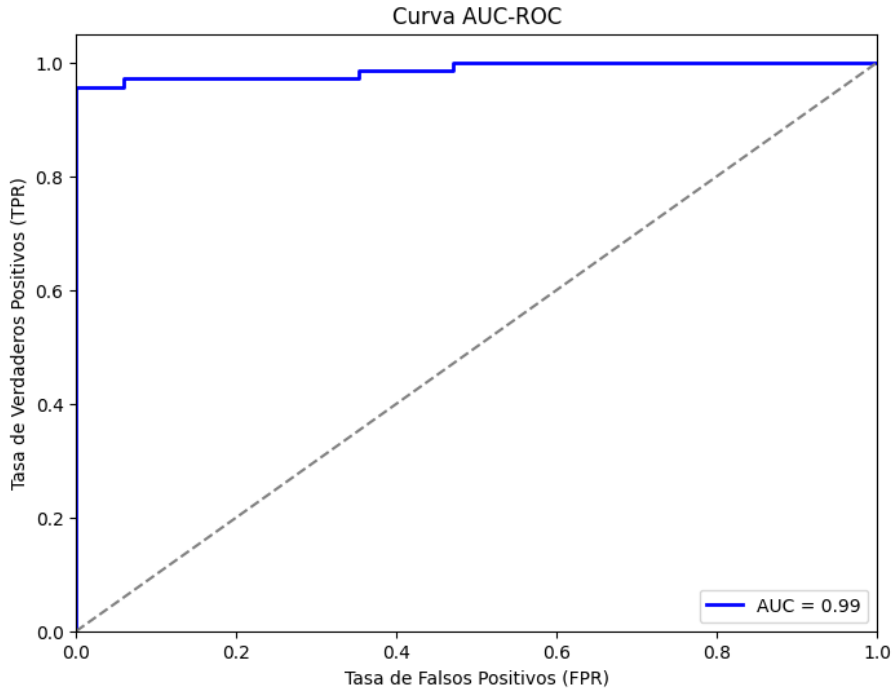
Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

4.8.3.8. AUC-ROC (Área bajo la curva ROC)

Como se puede observar en la Figura 24, nuestro modelo tiene valor de AUC de 0.99 que indica un rendimiento casi perfecto al distinguir los positivos y negativos, este valor nos indica que el modelo pude clasificar correctamente todos los casos.

Figura 24

Curva AUC-ROC del modelo MLP



Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

4.8.4. Regresión lineal múltiple

La regresión lineal múltiple es un modelo de aprendizaje supervisado que nos permite predecir un valor numérico utilizando variables independientes, para nuestro proyecto vamos a predecir el porcentaje de utilidad tomando en cuenta que este valor depende de los ingresos y costos de la institución.

4.8.4.1. Selección de variables.

Para la creación de este modelo vamos a utilizar solamente las variables de ingresos y costos ya que el objetivo es predecir el porcentaje de utilidad de la institución.

A continuación, esta es la lista de las variables numéricas con las que vamos a trabajar aplicando este modelo de regresión:

- i_matricula (Independiente)
- i_otros (Independiente)
- c_docentes (Independiente)
- c_adminis (Independiente)
- c_otros (Independiente)
- p_utilidad (Dependiente)

4.8.4.2. Eliminar outliers

Como se pudo evidenciar en la fase de exploración de los datos existen valores atípicos que para este caso vamos a eliminar para que no afecten al desempeño del modelo, a continuación, este es el resultado de la eliminación de los outliers:

- Datos antes de eliminar outliers: 288
- Datos después de eliminar outliers: 262

4.8.4.3. Trasformación de los datos

Vamos a utilizar la función logarítmica natural de Python para transformar nuestros datos con esto se quiere reducir la dispersión de los datos, y hacer que la distribución de los datos sea más simétrica.

4.8.4.4. Dividir los datos en entrenamiento y prueba

En este caso vamos a dividir los datos en un porcentaje de 80% para entrenamiento y 20% para pruebas.

4.8.4.5. Creación del modelo

Una vez que ya tenemos nuestros datos transformados procedemos a generar el modelo de regresión lineal múltiple y entrenarlo para posteriormente evaluar los resultados obtenidos.

4.8.4.6. Evaluación del modelo

Para evaluar nuestro modelo vamos a utilizar el MSE que mide el error promedio cuadrado entre los valores reales y los predichos mientras más pequeño es el MSE el modelo es mejor, el R-cuadrado mide en que porcentaje el modelo explica la variabilidad del p_utilidad.

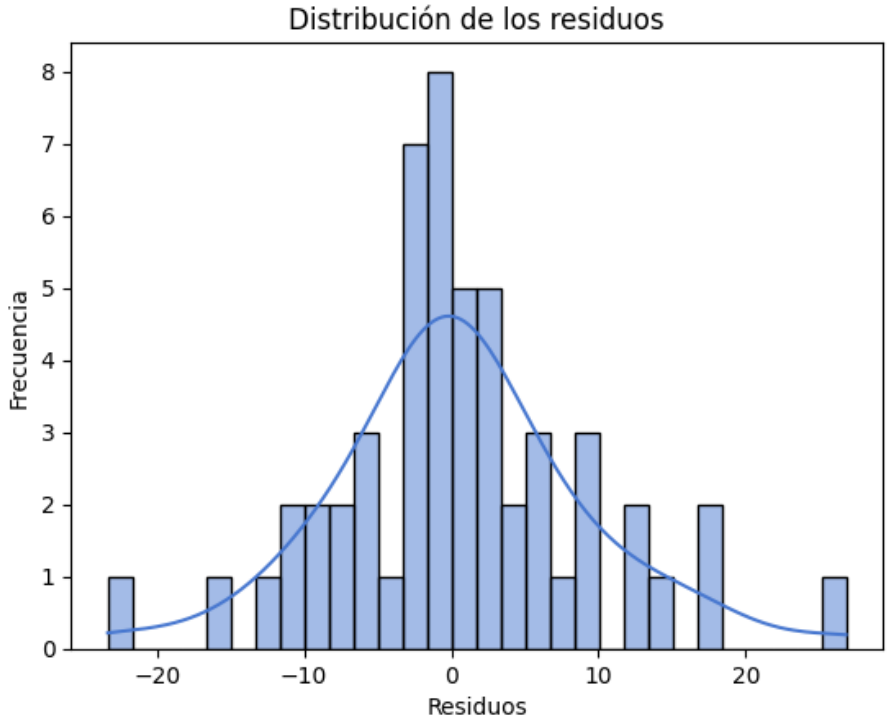
En este caso los resultados del modelo son que el Mean Squared Error (MSE) es igual a 75.1766505436887 y el R-squared es igual a 0.8931615542927687, lo que nos indica que el modelo explica en un 89% la variabilidad del p_utilidad este resultado nos dice que el modelo predice baste bien, el valor de MSE también es bajo por lo tanto indica un mejor ajuste.

4.8.4.7. Distribución de los residuos

Los residuos son la diferencia entre los valores reales y los valores predichos por el modelo de regresión, estos valores deben seguir una distribución normal para que el modelo sea adecuado, como se puede ver en la Figura 25, la distribución de los residuos sigue una distribución normal.

Figura 25

Distribución de los residuos



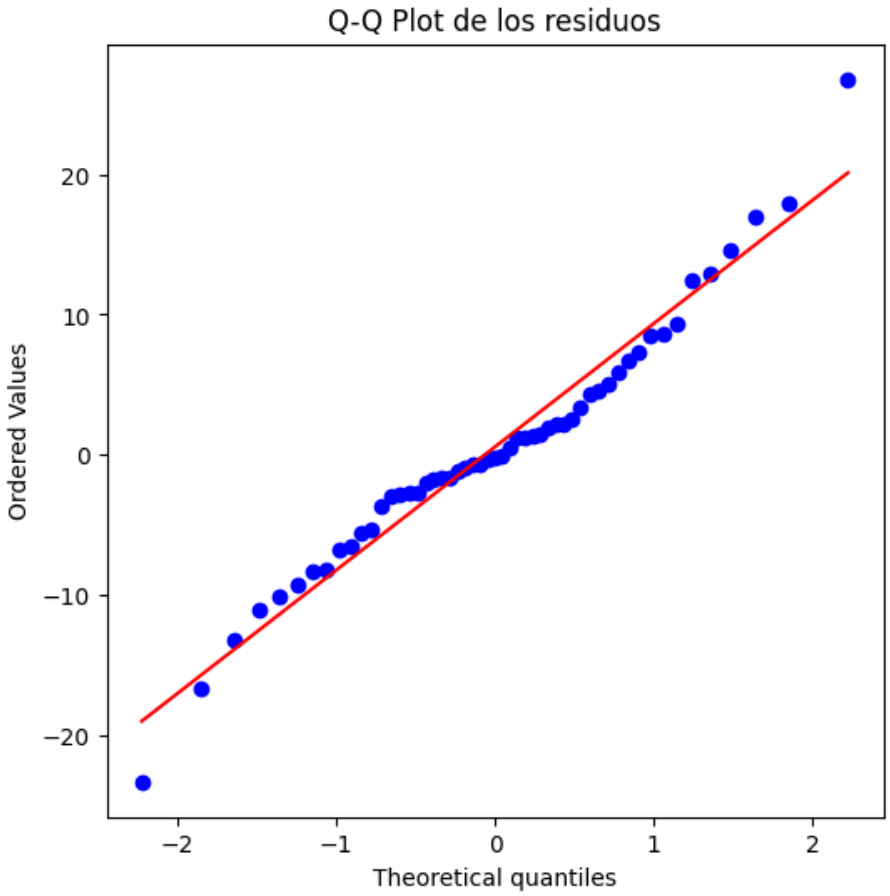
Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

4.8.4.8. Gráfico Q-Q plot (Quantile-Quantile)

Este grafico nos permite comparar la distribución de los residuos con una distribución normal teórica, como se puede observar en la Figura 26, los puntos azules están alineados con la línea roja lo que nos indica que los residuos siguen una distribución normal.

Figura 26

Gráfico Q-Q de los residuos



Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

4.8.4.9. Estadístico de Shapiro-Wilk

Esta prueba nos permite evaluar si los residuos siguen una distribución normal, los resultados obtenidos son: Estadístico de Shapiro-Wilk: 0.9732, p-valor: 0.2770, con este resultado del estadístico p que es mayor que 0.05 no se rechaza la hipótesis nula, se concluye entonces que los residuos siguen una distribución normal, el valor de Shapiro-Wilk es cercano a uno lo que refuerza la hipótesis de que los residuos siguen una distribución normal.

4.9. Evaluación

En esta fase vamos a medir la precisión de los modelos creados, se debe tomar en cuenta que tenemos 3 modelos de clasificación y uno de regresión, en la Tabla 4 se puede observar las métricas que cada uno de los modelos obtuvo en la evaluación individual.

Tabla 4

Resumen de las métricas de los modelos

Modelo	**Accuracy**	**Precision**		**Recall**		**F1-Score**		**roc_auc**	**mse**	**R ²**
		C0	C1	C0	C1	C0	C1			
Regresión Logística	0.98	0.89	1.00	1.00	0.97	0.94	0.99	1.00		
Random Forest	0.92	0.86	0.93	0.71	0.97	0.77	0.95	0.96		
MLP (Perceptrón Multicapa)	0.90	0.81	0.94	0.76	0.96	0.79	0.95	0.96		
Regresión Lineal Múltiple	-	-	-	-	-	-	-	-	75.15	0.89

Nota. Datos obtenidos mediante la herramienta Python. Fuente: Elaboración propia.

Antes de recomendar un modelo es necesario tomar en cuenta cuando se debe utilizar cada uno, en la Tabla 5 podemos observar una referencia de que tipo de modelo es y cuando se debe utilizarlo dependiendo de sus características.

Tabla 5

Resumen del tipo de modelo

Modelo	Tipo	¿Cuándo usarlo?
Regresión Logística	Clasificación	Cuando se necesita un modelo rápido y fácil de entender
Random Forest	Clasificación	Cuando se necesita mayor precisión, aunque dependiendo de los datos puede ser más lento
MLP (Perceptrón Multicapa)	Clasificación	Cuando los datos son muy complejos y se necesita detectar patrones más complicados
Regresión Lineal Múltiple	Regresión	Cuando se requiere predecir valores numéricos y la relación que existe entre las variables es clara.

Nota. Fuente: Elaboración propia.

4.9.1.1. Selección del mejor modelo

Las métricas de evaluación son muy similares para cada modelo, pero la recomendación el caso de clasificación seria Random Forest ya que este modelo tiene una alta precisión en la clasificación y reduce el riesgo de sobre ajuste gracias a la combinación de varios árboles de decisión, además tiene un buen manejo de los datos no lineales y de las variables categóricas.

4.9.1.2. Cumplimiento de objetivos

Cada uno de los modelos desarrollados nos ayuda a cumplir el objetivo de evaluar la situación financiera de cada una de las carreras, los modelos de clasificación nos permiten saber si una carrera será rentable o no en función de las variables independientes, mientras que el modelo de regresión nos ayuda a predecir el porcentaje de utilidad de las carreras en función de los ingresos

y costos, esta información nos muestra un panorama mas claro de como se van a desempeñar las carreras en el tiempo.

4.9.1.3. Limitaciones del modelo

Los modelos están limitados por la información que se tiene para analizar en este sentido la falta de información relevante puede afectar la precisión de los modelos.

4.9.1.4. Insights obtenidos

Luego de analizar los datos y evaluar todos los modelos se han obtenido los siguientes insights que considero relevantes para la institución.

- Factores externos como las políticas educativas y la situación económica afectan negativamente a la matriculación de los estudiantes.
- Las carreras de modalidad presencial tienen una matriculación estable a lo largo del tiempo mientras que las otras modalidades presentan variaciones altas en el número de matriculados.
- La mayoría de carreas son rentables sin embargo hay algunas que no llegan a cubrir sus costos de operación.
- La asignación proporcional de los costos puede generar un sesgo en la rentabilidad de algunas carreras
- Se debe analizar el impacto que tienen los procesos administrativos en el porcentaje de graduados.

- Las carreras con bajos ingresos deberían ser evaluadas para tomar la decisión de reestructurarlas o eliminarlas.
- Se deben implementar planes de seguimiento y retención en las carreras de modalidad hibrida y en línea para reducir el porcentaje de deserción.

4.10. Despliegue

Esta fase no está contemplada en el alcance del proyecto ya que se requieren recursos técnicos y tecnológicos adicionales que no son parte de este proyecto. Pero si se comunicará a la institución de los hallazgos para que sean considerados en la planificación operativa.

CAPÍTULO V: CONCLUSIONES Y RECOMENDACIONES

5. Conclusiones y recomendaciones

5.1. Conclusiones

- **Impacto de las políticas educativas en el número de matriculados:** Se ha evidenciado una reducción en el número de estudiantes que ingresan a la institución en los últimos años, esto está relacionado con el cambio de las políticas educativas y también con los factores económicos de la población, por este motivo el modelo debería incluir variables de tipo socioeconómicas que le permitan mejorar su precisión.

- **Relación entre los costos y rentabilidad:** Los costos de docencia, administrativos y otros costos se manejan de forma global y se asignaron forma proporcional lo que podría afectar en la clasificación y predicción de la rentabilidad.

- **Validación de los datos**: La validación realizada en el desarrollo del proyecto confirma que no existen valores nulos ni errores tipográficos en la base de datos, pero si existen valores atípicos en ingresos y costos esto puede afectar el rendimiento del modelo.

- **Precisión de los modelos:** Para la creación de los modelos se utilizó los datos de últimos 10 años de la institución procurando obtener la información más representativa, no obstante, la existencia de valores atípicos puede afectar el rendimiento de nuestros modelos.

5.2. Recomendaciones

- **Incorporar variables socioeconómicas:** Para mejorar la precisión de los modelos en general se recomienda incluir variables socioeconómicas que puedan influir en la matriculación de los estudiantes y en la rentabilidad de las carreras.
- **Método de distribución de costos:** Se recomienda buscar un método mas preciso para la asignación de los costos de docencia, administrativos y otros costos para evitar los sesgos en los modelos predictivos y de clasificación.
- **Tratamiento de valores atípicos:** Es necesario aplicar técnicas de limpieza y transformación a estos valores para tratar de mitigar los sesgos en los modelos.
- **Análisis profundo de la deserción:** Se recomienda realizar un estudio mas a fondo de las razones por las que los alumnos no culminan el periodo académico, agregando más información como por ejemplo el porcentaje de asistencia a clases.
- **Diversificación de los ingresos:** Se recomienda incrementar fuentes de ingreso alternativas como cursos extracurriculares o alianzas estratégicas con otras instituciones.
- **Mejor segmentación de mercado:** Se recomienda identificar el perfil de los estudiantes con mayor permanencia y enfocar las estrategias de captación y retención en ellos.
- **Monitoreo y validación continua:** Ya que los datos cambian en el tiempo es necesario monitorear y si es necesario reentrenar los modelos con nueva información para asegurar su precisión.

6. Referencias bibliográficas

Becerra Correa, N., & Leguizamón Páez, M. A. (05 de Julio de 2024). *Regresión Logística Técnica de Machine Learning para predicciones académicas.* Obtenido de XIKUA Boletín Científico de la Escuela Superior de Tlahuelilpan: https://repository.uaeh.edu.mx/revistas/index.php/xikua/article/view/12746

Bruce, P., Bruce, A., & Gedeck, P. (2022). *Estadística práctica para ciencia de datos con r y python.* Marcombo, S.A.

Buttu, M. (2020). *El Gran Libro de Python.* Marcombo, S.A.

Casas Roma, J., Nin Guerrero, J., & Julbe López, F. (2019). *Big Data : Análisis de Datos en Entornos Masivos.* Editorial UOC.

CES. (2024). *Estadísticas IES*. Obtenido de Consejo de Educación Superior: https://www.ces.gob.ec/?page_id=5248

El flujo de caja. (s.f.). Obtenido de El flujo de caja: https://www.munich-business-school.de/es/l/diccionario-de-estudios-empresariales/flujo-de-caja

Macías Macías, L. M., & Toala Mendoza , S. T. (Julio de 2022). *DESAFÍOS DE SOSTENIBILIDAD FINANCIERA EN ENTIDADES MUNICIPALES AUTOFINANCIADAS DEL CANTÓN SANTA ANA.* Obtenido de Editorial Universitaria ULEAM - Publicaciones Científicas Digitales: https://publicacionescd.uleam.edu.ec/index.php/corporatum-360/article/view/347/557

Murillo, N. A. (Febrero de 2019). Obtenido de https://rinacional.tecnm.mx/jspui/handle/TecNM/810

Quintana, C. (Agosto de 2022). *Margen de utilidad: qué es, cómo se calcula y para qué sirve*. Obtenido de Margen de utilidad: qué es, cómo se calcula y para qué sirve: https://www.oberlo.com/es/blog/margen-de-utilidad

Rendición de cuentas 2023 . (2023). Obtenido de Tecnológico Universitario Cordillera: https://www.cordillera.edu.ec/nosotros/rendicion-de-cuentas/rendicion-de-cuentas-2023/

Rodríguez Rodríguez, A., Rodríguez González, A., Pino Tarragó, J., & Domínguez Gálvez, D. (Marzo de 2021). *Implementación de algoritmos de Inteligencia Artificial en la predicción de nuevos conocimientos mediante enseñanza constructivista.* Obtenido de Dialnet: https://dialnet.unirioja.es/servlet/articulo?codigo=8590452

Sandoval, L. J. (15 de 10 de 2018). *Algoritmos de aprendizaje automático para análisis y predicción de datos.* Obtenido de Digital de Ciencia y Cultura de El Salvador REDICCES: http://redicces.org.sv/jspui/handle/10972/3626

Seaborn: todo sobre la herramienta de Data Visualization Python. (2023). Obtenido de DataScientest: https://datascientest.com/es/seaborn-la-herramienta-de-data-visualization-python

Serra Marrugat, A. (07 de Julio de 2020). *Comparación de algoritmos de clasificación supervisada.* Obtenido de Universitat Politècnica de Catalunya BarcelonaTech: https://upcommons.upc.edu/handle/2117/330482

Shimaoka, A. M. (Octubre de 2024). *The evolution of CRISP-DM for Data Science.* Obtenido de Digital Open Library of the Brazilian Computing Society Journals: https://journals-sol.sbc.org.br/index.php/reviews/article/view/3757/2996

Taborda Blandón, G. E., Castaño Zuluaga, B. S., Durán Vásquez, J. M., Conto López, R., & Reyes Moreno, E. R. (2024). *Propuesta de modelo de analítica para flujo de caja en mipymes en Colombia.* Obtenido de Instituto Tecnológico Metropolitano: https://www.redalyc.org/journal/6381/638176083008/638176083008.pdf

Timón, C. E. (16 de Enero de 2017). *Análisis predictivo : Técnicas y modelos utilizados y aplicaciones del mismo - herramientas Open Source que permiten su uso.* Obtenido de Universitat Oberta de Catalunya: https://openaccess.uoc.edu/handle/10609/59565

Valverde, C. M. (Junio de 2018). *Librería de métodos de poda en conjuntos de clasificadores para Scikit-Learn.* Obtenido de UNIVERSIDAD AUTONOMA DE MADRID: https://repositorio.uam.es/handle/10486/688291

Zapata, J. R. (Noviembre de 2024). *Manipulación de Datos con Python*. Obtenido de https://joserzapata.github.io/courses/python-ciencia-datos/pandas/

Zárate Valderrama, J., Bedregal Alpaca, N., & Cornejo Aparicio, V. (16 de 10 de 2020). *Modelos de clasificación para reconocer patrones de deserción en estudiantes universitarios.* Obtenido de Ingeniare Revista chilena de ingeniería: https://www.scielo.cl/scielo.php?pid=S0718-33052021000100168&script=sci_arttext&tlng=en

7. Anexos

7.1. Código fuente de los modelos

1 Librerías necesarias

[1]:
```
import pickle
import pandas
as pd import
numpy as np
import matplotlib.pyplot as
plt import seaborn as sns
from sklearn.preprocessing import MinMaxScaler, StandardScaler,
RobustScaler
from sklearn.model_selection import train_test_split, GridSearchCV,␣
  ↪cross_val_score
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.metrics import accuracy_score, confusion_matrix,␣
  ↪classification_report, f1_score, roc_curve
from sklearn.metrics import roc_auc_score, auc, make_scorer,
precision_score,␣
  ↪recall_score, mean_squared_error, r2_score
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from statsmodels.stats.outliers_influence import variance_inflation_factor
from imblearn.over_sampling import SMOTE
from collections import Counter
from tabulate import tabulate
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Input
```

2 Comprensión de los datos

2.1 Recopilación de datos iniciales

[2]:
```
from google.colab import drive drive.mount('/content/drive') Mounted at
/content/drive
```

[3]:
```
# Cargar la data
file_path = '/content/drive/MyDrive/repositorio/PUCE/Tesis/Datos modelo␣
  ↪predictivo/data_modelo.xlsx' df = pd.read_excel(file_path)
  df.head()
```

[3]:

	anio	periodo	cod_carrera
0	2015	ABR 2015_SEP 2015	CAR004
1	2015	ABR 2015_SEP 2015	CAR006
2	2015	ABR 2015_SEP 2015	CAR008
3	2015	ABR 2015_SEP 2015	CAR010

```
4  2015  ABR 2015_SEP 2015      CAR014

                                     nombre_carrera  modalidad
                                     d_semestres  \
0                                    DISEÑO GRAFICO  PRESENCIAL
6
1  ADMINISTRACION PARA EL DESARROLLO DEL TALENTO …  PRESENCIAL          6
2                 ADMINISTRACION TURISTICA HOTELERA  PRESENCIAL
6
3             ADMINISTRACION DE BOTICAS Y FARMACIAS  PRESENCIAL
6
4                           ANALISTAS DE SISTEMAS  PRESENCIAL
6

   n_matriculados p_desercion  p_graduados  i_matricula   i_otros  \
0             859       10.13         5.59    386083.19  232623.50
1             914        6.46        12.91    412986.00  209527.19
2             475        4.84        14.95    249061.50  108889.95
3             360        7.22         8.06    158389.32   82527.12
4             646        8.51        10.68    279789.68  148090.33

   c_docentes  c_adminis   c_otros  p_utilidad  rentable
0    81263.87  108641.85  140110.23       52.56         1
1    91791.64  115597.97  149081.20       47.83         1
2    78673.22   60075.53   77476.56       53.64         1
3    77715.66   45530.93   58719.07       43.84         1
4    82553.70   81702.72  105368.11       48.35         1
```

2.2 Descripción de los datos

[4]:
```
# Identificar tipos de datos y valores nulos
print(df.info())
```

```
<class
'pandas.core.frame.DataFrame'>
RangeIndex: 288 entries, 0 to
287
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------
                -----
 0   anio            288 non-null    int64
 1   periodo         288 non-null    object
 2   cod_carrera     288 non-null    object
 3   nombre_carrera  288 non-null    object
 4   modalidad       288 non-null    object
 5   d_semestres     288 non-null    int64
```

```
 6   n_matriculados  288 non-null    int64
 7   p_desercion     288 non-null    float64
 8   p_graduados     288 non-null    float64
 9   i_matricula     288 non-null    float64
 10  i_otros         288 non-null    float64
 11  c_docentes      288 non-null    float64
 12  c_adminis       288 non-null    float64
 13  c_otros         288 non-null    float64
 14  p_utilidad      288 non-null    float64
 15  rentable        288 non-null
int64 dtypes: float64(8), int64(4),
object(4) memory usage: 36.1+ KB
None
```

2.3 Exploración de los datos (EDA)

[5]:
```
# Contar el número de años
num_anios =
df['anio'].nunique()
print(f'Número de años:
{num_anios}')
```

```
Número de años: 10
```

[6]:
```
# Contar el número de periodos académcos
num_periodos = df['periodo'].nunique()
print(f'Número de periodo académicos: {num_periodos}')
```

```
Número de periodo académicos: 19
```

[7]:
```
# Contar el número de carreras
num_cod = df['cod_carrera'].nunique()
print(f'Número de códigos de carrera: {num_cod}')
```

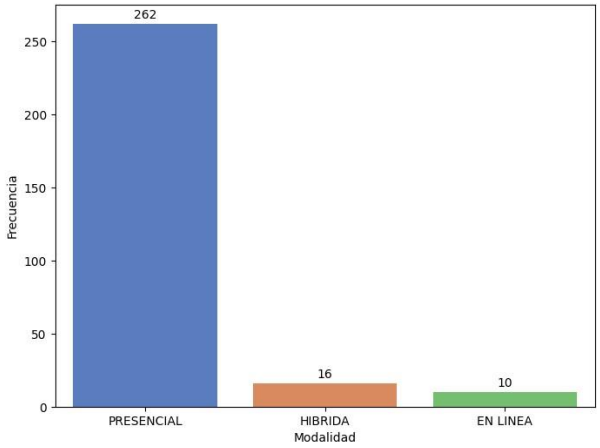
```
Número de códigos de carrera: 40
```

[8]:
```
# Contar el número de carreras
num_carreras = df['nombre_carrera'].nunique()
print(f'Número de carreras: {num_carreras}')
```

```
Número de carreras: 40
```

[9]:
```
# Graficar frecuencia de la modalidad de estudio
plt.figure(figsize=(8, 6))
```

```
ax = sns.countplot(x='modalidad', hue='modalidad', data=df,
palette='muted')

for container in ax.containers:
    ax.bar_label(container, fmt='%d', label_type='edge', fontsize=10,
    padding=2)

plt.xlabel('Modalidad')
plt.ylabel('Frecuencia')

plt.show()
```

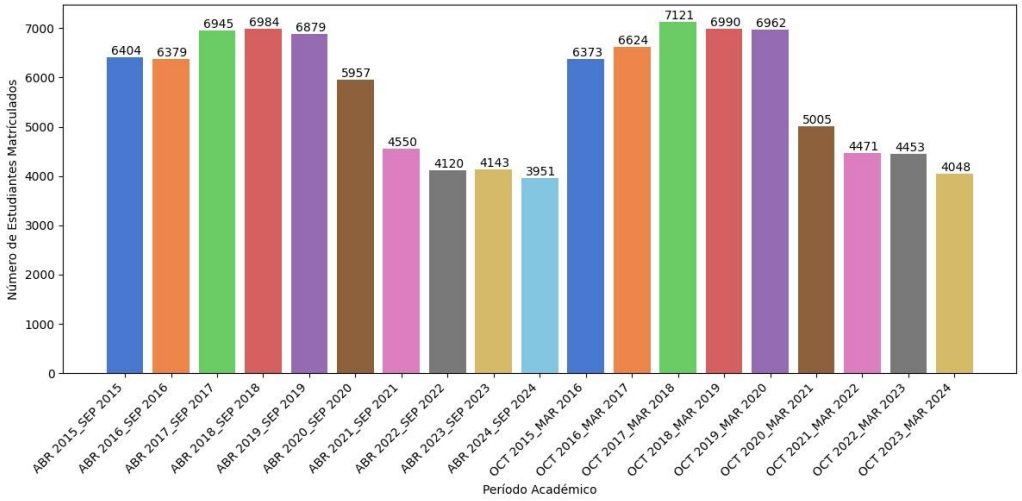


[10]:
```
# Contar el número de carreras
d_semestres = df['d_semestres'].nunique()
print(f'Número de semestres: {d_semestres}')
```

```
Número de semestres: 4
```

[11]:
```
# Graficar el núemero de alumnos por periodo académico
total_por_periodo =
df.groupby('periodo')['n_matriculados'].sum().reset_index()
sns.set_palette('muted')
colors = sns.color_palette('muted', len(total_por_periodo))
plt.figure(figsize=(12, 6))
bar_width = 0.8  # Ancho de las barras
bar_spacing = 0.8  # Wspaciado entre las
barras bars =
plt.bar(total_por_periodo['periodo'],␣
  ↪total_por_periodo['n_matriculados'], color=colors, width=bar_width)
plt.xticks(range(len(total_por_periodo)), total_por_periodo['periodo'],␣
  ↪rotation=45, ha='right')
# Agregar los valores sobre las barras
```

```
    for bar in bars:
        yval = bar.get_height()
        plt.text(bar.get_x() + 0.1, yval, int(yval), va='bottom', ha='left')
    # plt.title('Total de Estudiantes Matriculados por Período')
    plt.xlabel('Período Académico')
    plt.ylabel('Número de Estudiantes
    Matrículados') plt.tight_layout()
    plt.show()
```

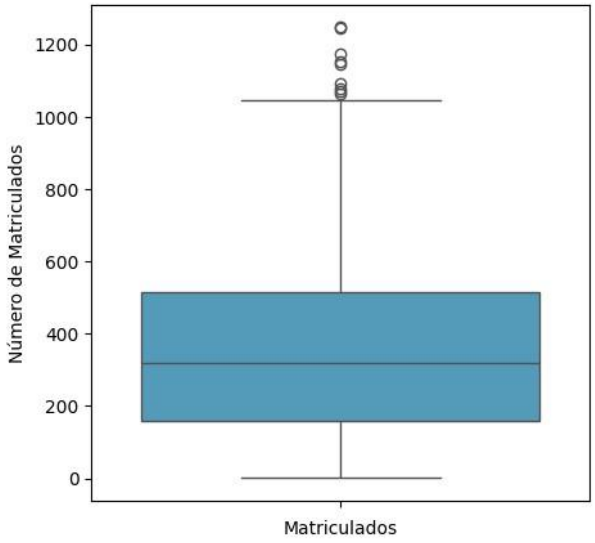


[12]:
```
# Estadísticas básicas para columnas numéricas
print(df.describe())
```

```
              anio  d_semestres  n_matriculados  p_desercion
p_graduados  \ count   288.000000   288.000000     288.000000
288.000000   288.000000 mean   2019.861111    5.243056     376.246528
9.363958    15.899236
std       2.715579     0.771862     277.502133     9.996857
22.558662 min    2015.000000     2.000000       1.000000
0.000000     0.000000
25%    2018.000000     5.000000     157.000000     5.050000     0.000000
50%    2020.000000     5.000000     319.500000     7.725000     8.300000
75%    2022.000000     6.000000     515.500000    10.877500
17.600000 max    2024.000000     6.000000     1248.000000
100.000000    95.960000

         i_matricula        i_otros     c_docentes
c_adminis  \ count     288.000000     288.000000     288.000000
288.000000 mean   223473.540278   56690.243229   75389.287847
51920.047292
std    156833.988837   51522.494362   66251.470057
34805.973662 min        52.000000       7.330000
0.000000     190.900000
```

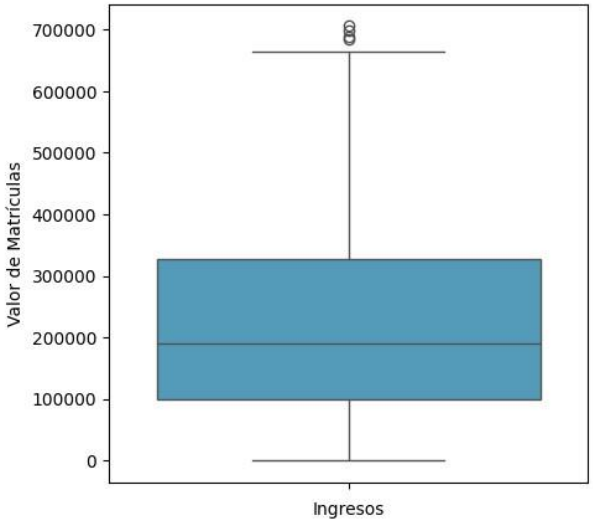
```
25%    100249.855000  17388.655000  31003.552500  24739.617500
50%    190102.565000  46741.755000  64751.070000  45540.690000


75%    327061.230000  78565.732500  94637.000000
75070.297500 max    706369.460000 292340.730000
500205.930000 155837.980000

            c_otros         p_utilidad
rentable count    288.000000 288.000000
288.000000   mean          78473.408681
39.179618              0.833333     std
59222.057832  27.670127    0.373327 min
156.900000  -82.000000    0.000000
25%     31058.245000  22.912500    1.000000
50%     64609.965000  48.490000    1.000000
75%    114298.607500  58.197500
1.000000 max    262774.550000
81.720000    1.000000
```

[13]:
```
# Boxplots para detectar outliers
plt.figure(figsize=(5, 5))
sns.boxplot(data=df['n_matriculados'],color="#43a2
ca") plt.ylabel("Número de Matriculados")
plt.xlabel("Matriculados")
plt.show()
```

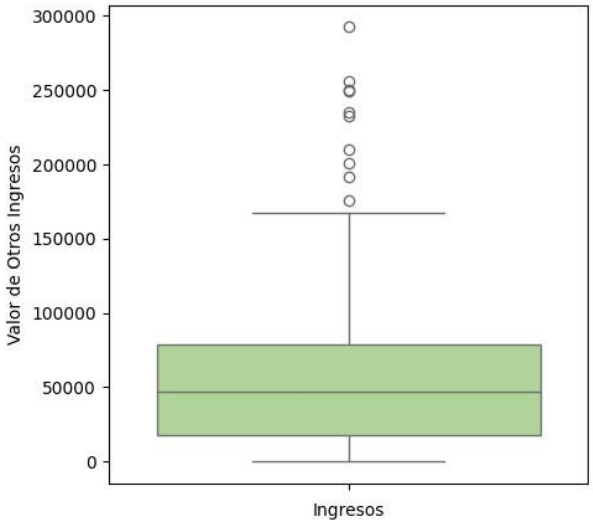


[14]:
```
# Boxplots para detectar outliers
plt.figure(figsize=(5, 5))
sns.boxplot(data=df['i_matricula'],color="#43a2
ca") plt.ylabel("Valor de Matrículas")
plt.xlabel("Ingresos")
```

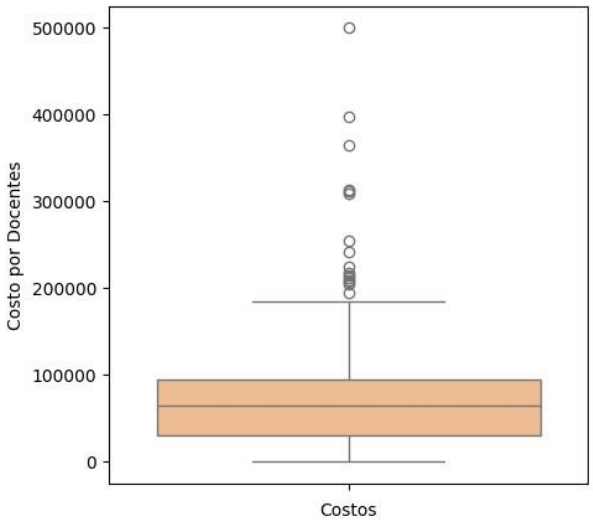
```
plt.show()
```



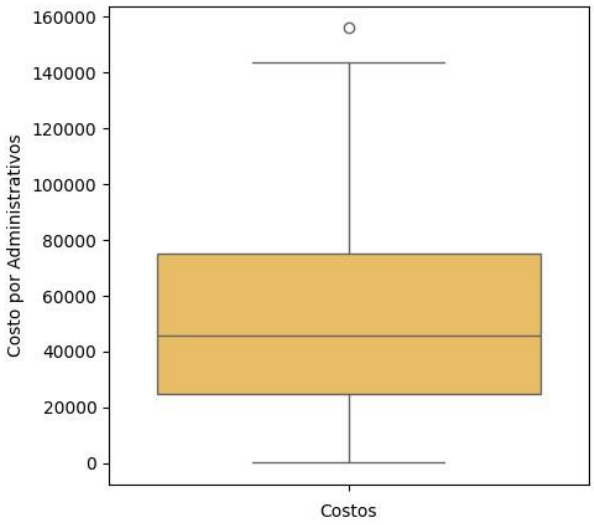
[15]:
```
# Boxplots para detectar outliers
plt.figure(figsize=(5, 5))
sns.boxplot(data=df['i_otros'],color="#addd
8e") plt.ylabel("Valor de Otros Ingresos")
plt.xlabel("Ingresos")
plt.show()
```



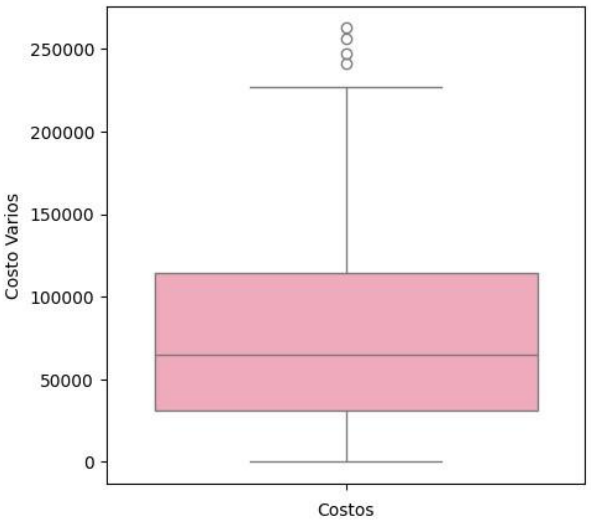
[16]:
```
# Boxplots para detectar outliers
plt.figure(figsize=(5, 5))
sns.boxplot(data=df['c_docentes'],color="#fdbb
84") plt.ylabel("Costo por Docentes")
plt.xlabel("Costos")
plt.show()
```



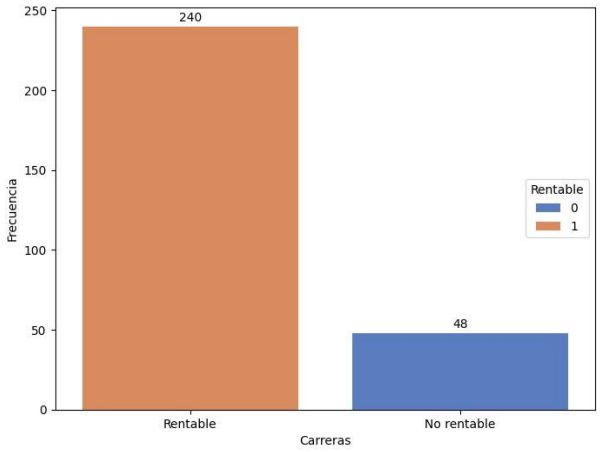
[17]:
```
# Boxplots para detectar outliers
plt.figure(figsize=(5, 5))
sns.boxplot(data=df['c_adminis'],color="#fec4
4f") plt.ylabel("Costo por Administrativos")
plt.xlabel("Costos")
plt.show()
```



[18]:
```
# Boxplots para detectar outliers
plt.figure(figsize=(5, 5))
sns.boxplot(data=df['c_otros'],color="#fa9f
b5") plt.ylabel("Costo Varios")
plt.xlabel("Costos")
plt.show()
```



[19]:
```
# Graficar frecuencia de la rentabilidad
plt.figure(figsize=(8, 6))
orden = df['rentable'].value_counts().index
ax = sns.countplot(x='rentable', hue='rentable',␣
  ↪data=df,order=orden,palette='muted')


# Etiquetas personalizadas
ax.set_xticks(range(len(orden)))
nuevas_etiquetas = ['Rentable', 'No
rentable']
ax.set_xticklabels(nuevas_etiquetas, fontsize=10)

for container in ax.containers:
    ax.bar_label(container, fmt='%d', label_type='edge', fontsize=10,
    padding=2)

plt.legend(title="Rentable", loc='center right' )
plt.xlabel('Carr
eras')
plt.ylabel('Frecue
ncia') plt.show()
```



3 Preparación de los datos

3.1 Limpieza de datos

[20]:
```
conteo_ceros = (df == 0).sum()

# Filtrar solo las columnas que tienen al menos un 0
conteo_ceros = conteo_ceros[conteo_ceros > 0]

# Crear un DataFrame con la información
tabla_ceros = pd.DataFrame({
    'Columna': conteo_ceros.index,
    'Cantidad de ceros': conteo_ceros.values
})

# Mostrar la tabla con formato
print(tabulate(tabla_ceros, headers='keys', tablefmt='grid'))
```

```
+----+-------------+---------------------+
|    | Columna     |   Cantidad de ceros |
+====+=============+=====================+
|  0 | p_desercion |                   9 |
+----+-------------+---------------------+
|  1 | p_graduados |                  73 |
+----+-------------+---------------------+
|  2 | c_docentes  |                   2 |
+----+-------------+---------------------+
|  3 | rentable    |                  48 |
+----+-------------+---------------------+
```

[21]:
```
# Contar
registros columna
```

```
= 'c_docentes'
df.shape
```

[21]:
```
(288, 16)
```

[22]:
```
# Contar valores menores a 12
columna = 'n_matriculados'
cantidad_menores_12 = (df[columna] < 10).sum()
print(f"La columna '{columna}' tiene {cantidad_menores_12} valor menor a 12.")
```
```
La columna 'n_matriculados' tiene 3 valor menor a 12.
```

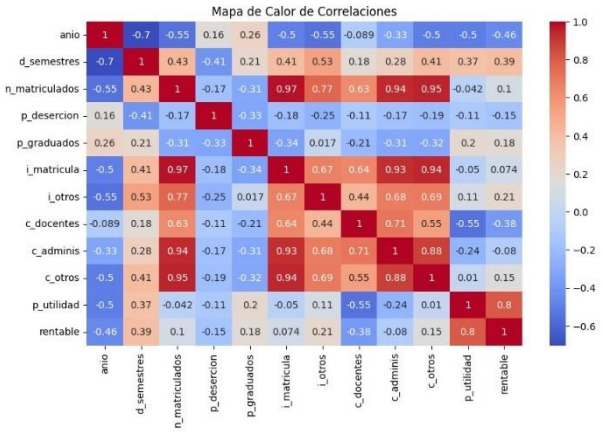
[23]:
```
# Contar registros
columna =
'n_matriculados'
df.shape
```

[23]:
```
(288, 16)
```

3.2 Construcción de nuevos datos

[24]:
```
# Graficar la correlación de las variables numéricas
plt.figure(figsize=(10, 6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title('Mapa de Calor de Correlaciones')
```

[24]:
```
Text(0.5, 1.0, 'Mapa de Calor de Correlaciones')
```



3.2.1 Data set para regresión logística

[25]:
```
# Realizar una copia de dataset original
df_reglog = df.copy()
```

```
# Eliminar columnas que presentan multicolinealidad
df_reglog['ingresos_totales'] =
df_reglog['i_matricula'] + df['i_otros'] df_reglog =
df_reglog.drop(columns=['anio','periodo',␣
  ↪'cod_carrera','nombre_carrera','n_matriculados','i_matricula','i_otros',
'c_adminis','c_otro print(df_reglog.shape)
(288, 8)
```

```
[26]: # Convertir columna la columna modalidad en numérica utilizando one-hot
encoding
df_reglog = pd.get_dummies(df_reglog,
columns=['modalidad'],drop_first=False,␣
  ↪dtype=int) # ,dtype=int
df_reglog.rename(columns={'modalidad_EN LINEA':
'mod_en_linea'}, inplace=True)
df_reglog.rename(columns={'modalidad_HIBRIDA':
'mod_hibrida'}, inplace=True)
df_reglog.rename(columns={'modalidad_PRESENCIAL':
'mod_presencial'},␣
  ↪inplace=True)

df_reglog.head()
```

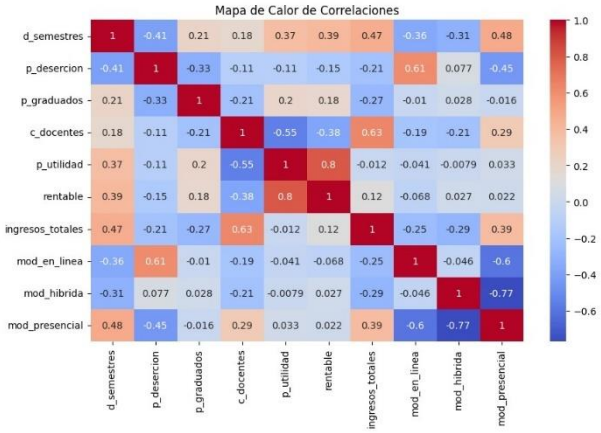
[26]:

	d_semestres	p_desercion	p_graduados	c_docentes	p_utilidad	rentable
0	6	10.13	5.59	81263.87	52.56	1
1	6	6.46	12.91	91791.64	47.83	1
2	6	4.84	14.95	78673.22	53.64	1
3	6	7.22	8.06	77715.66	43.84	1
4	6	8.51	10.68	82553.70	48.35	1

	ingresos_totales	mod_en_linea	mod_hibrida	mod_presencial
0	618706.69	0	0	1
1	622513.19	0	0	1
2	357951.45	0	0	1
3	240916.44	0	0	1
4	427880.01	0	0	1

```
[27]: # Graficar la correlación de las variables numéricas
plt.figure(figsize=(10, 6))
sns.heatmap(df_reglog.corr(numeric_only=True), annot=True,
cmap='coolwarm')
plt.title('Mapa de Calor de Correlaciones')
```

```
[27]: Text(0.5, 1.0, 'Mapa de Calor de Correlaciones')
```



3.2.2 Data set para random forest

```
[28]: # Realizar una copia de dataset original
      df_ranfor = df.copy()
      df_ranfor['ingresos_totales'] = df_ranfor['i_matricula'] +
      df_ranfor['i_otros'] df_ranfor['otros_costos'] =
      df_ranfor['c_adminis'] + df_ranfor['c_otros'] df_ranfor =
      df_ranfor.drop(columns=['anio','periodo',␣
        'cod_carrera','nombre_carrera','modalidad','n_matriculados','i_matricula
        ','i_otros','c_otro

      print('\nDataset random forest')
      print(df_ranfor.shape)

     Dataset random forest
     (288, 8)
```

3.2.3 Data set para red neuronal MLP

```
[29]: # Realizar una copia de dataset original
      df_mlp = df.copy()

      # Eliminar columnas que presentan
      multicolinealidad df_mlp['ingresos_totales'] =
      df_mlp['i_matricula'] + df_mlp['i_otros'] df_mlp =
      df_mlp.drop(columns=['anio','periodo',␣
        'cod_carrera','nombre_carrera','modalidad','i_matricula','i_otros',
        'n_matriculados'])
      print(df_mlp.shape) (288, 9)

[30]: # Data set paar redes neuronales
      df_mlp.head()
```

[30]:
```
   d_semestres  p_desercion  p_graduados  c_docentes  c_adminis   c_otros
\
0            6        10.13         5.59    81263.87  108641.85  140110.23
1            6         6.46        12.91    91791.64  115597.97  149081.20
2            6         4.84        14.95    78673.22   60075.53   77476.56
3            6         7.22         8.06    77715.66   45530.93   58719.07
4            6         8.51        10.68    82553.70   81702.72  105368.11

   p_utilidad  rentable  ingresos_totales
0       52.56         1         618706.69
1       47.83         1         622513.19
2       53.64         1         357951.45
3       43.84         1         240916.44
4       48.35         1         427880.01
```

4 Modelado

4.1 Regresión Logistica

[31]:
```
# Obtenemos la variabe dependiente y las
                independientes
X = df_reglog.drop(columns=['rentable'],axis=1) # Variables independientes
y = df_reglog['rentable'] # Variable dependiente
print(X)
print(y)
```

```
     d_semestres  p_desercion  p_graduados  c_docentes  p_utilidad  \
0              6        10.13         5.59    81263.87       52.56
1              6         6.46        12.91    91791.64       47.83
2              6         4.84        14.95    78673.22       53.64
3              6         7.22         8.06    77715.66       43.84
4              6         8.51        10.68    82553.70       48.35
..           …            …            …           …          …
283            4       100.00         0.00        0.00       65.91
284            6         0.00        94.44     7716.76       40.96
285            4        24.14         0.00    16213.50      -10.99
286            4        35.29         0.00     4832.45       50.35
287            4        21.60        36.00    12879.26       47.72

     ingresos_totales  mod_en_linea  mod_hibrida  mod_presencial
0           618706.69             0            0               1
1           622513.19             0            0               1
2           357951.45             0            0               1
3           240916.44             0            0               1
4           427880.01             0            0               1
..                …             …            …               …
283             63.12             1            0               0
284          15853.78             0            1               0
```

```
285          17635.32            1            0              0
286          18325.48            1            0              0
287          82159.34            1            0              0

[288 rows x 9 columns]
0      1
1      1
2      1
3      1
4      1
      ..
283    1
284    1

285    0
286    1
287    1
Name: rentable, Length: 288, dtype: int64
```

[32]:
```
# Dividimos la data en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
  ↪random_state=42)

print("Distribución en y_train:", Counter(y_train))
print("Distribución en y_test:", Counter(y_test))
```

```
Distribución en y_train: Counter({1: 170, 0:
31}) Distribución en y_test: Counter({1: 70,
0: 17})
```

[33]:
```
# Utilizamos SMOTE para balancear las clases ya que tenemos pocos datos y␣
  ↪queremoss crear más
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train,
y_train)

y_train_resampled = y_train_resampled.astype(int)

print("Antes de SMOTE:", Counter(y_train))
print("Después de SMOTE:", Counter(y_train_resampled))
```

```
Antes de SMOTE: Counter({1: 170, 0:
31}) Después de SMOTE: Counter({0: 170,
1: 170})
```

[34]:
```
# Escalar los datos numéricos
```

```
scaler1 = StandardScaler()

X_train_resam_df = pd.DataFrame(X_train_resampled, columns=X.columns)

# Escalar X_train
X_train_esc = scaler1.fit_transform(X_train_resam_df)
X_train_esc_df = pd.DataFrame(X_train_esc,
columns=X.columns)

# Guardar escalado original para las pruebas
with open('scaler1.pkl', 'wb') as scaler_file:
    pickle.dump(scaler1, scaler_file)

X_train_esc_df.head()
```

[34]:

	d_semestres	p_desercion	p_graduados	c_docentes	p_utilidad	ingresos_totales	mod_en_linea	mod_hibrida mod_presencial
0	-0.905857	0.244635	-0.611288	-0.897989	-1.372703	-1.163187	-0.191273	4.110321 -2.160247
1	1.490962	-0.575071	0.831130	-0.703393	1.206355	-0.326907	-0.191273	-0.243290 0.462910
2	0.292553	-0.623661	0.515618	0.746653	-0.542984	0.061818	-0.191273	-0.243290 0.462910
3	0.292553	-0.379651	1.837063	0.290290	0.024808	-0.146763	-0.191273	-0.243290 0.462910
4	0.292553	0.077736	-0.611288	-0.324174	0.787271	0.854499	-0.191273	-0.243290 0.462910

[35]:
```
# Escalar X_test
X_test_esc = scaler1.transform(X_test)
X_test_esc_df = pd.DataFrame(X_test_esc,
columns=X_test.columns) X_test_esc_df.head()
```

[35]:

	d_semestres	p_desercion	p_graduados	c_docentes	p_utilidad	ingresos_totales	mod_en_linea	mod_hibrida mod_presencial
0	1.490962	-0.347961	-0.280518	0.167600	1.014049			
1	1.490962	-0.822250	0.642586	-1.084183	1.676135			
2	1.490962	0.324916	-0.109956	0.929014	-1.115169			
3	1.490962	-0.675421	-0.196055	-0.075435	1.551761			
4	0.292553	0.002737	-0.366071	-0.042358	-0.016763			

```
0          0.577715    -0.191273    -0.24329        0.46291
1         -0.671636    -0.191273    -0.24329        0.46291
2         -0.212587    -0.191273    -0.24329        0.46291
3          1.227710    -0.191273    -0.24329        0.46291
4          0.238759    -0.191273    -0.24329        0.46291
```

[36]:
```
# Definir el modelo de regresión logística para
                    Ridge (L2)
modelo_ridge = LogisticRegression(solver='liblinear', penalty='l2',␣
  ↪max_iter=300)
# Definir el modelo de regresión logística para Lasso (L1)
modelo_lasso = LogisticRegression(solver='liblinear', penalty='l1',␣
  ↪max_iter=300)


# Rango de valores para C
param_grid = {'C': [0.01, 0.1, 1, 10, 100]}

# Crear los objetos GridSearchCV para Ridge y Lasso
grid_search_ridge = GridSearchCV(estimator=modelo_ridge,
param_grid=param_grid,␣
  ↪cv=5, scoring='accuracy')
grid_search_lasso = GridSearchCV(estimator=modelo_lasso,
param_grid=param_grid,␣
  ↪cv=5, scoring='accuracy')

# Ajustar el modelo Ridge
grid_search_ridge.fit(X_train_esc_df,
y_train_resampled) mejor_modelo_ridge =
grid_search_ridge.best_estimator_

print(f"Mejor valor de C para Ridge:
{grid_search_ridge.best_params_['C']}")
print(f"Mejor exactitud para Ridge: {grid_search_ridge.best_score_}\n")

# Ajustar el modelo Lasso
grid_search_lasso.fit(X_train_esc_df,
y_train_resampled) mejor_modelo_lasso =
grid_search_lasso.best_estimator_
print(f"Mejor valor de C para Lasso:
{grid_search_lasso.best_params_['C']}")
print(f"Mejor exactitud para Lasso: {grid_search_lasso.best_score_}\n")

# Evaluar en el conjunto de prueba
predicciones_ridge = mejor_modelo_ridge.predict(X_test_esc_df)
predicciones_lasso = mejor_modelo_lasso.predict(X_test_esc_df)
```

```
print(f"Exactitud en el conjunto de prueba para Ridge:
{accuracy_score(y_test, ␣
 ↪predicciones_rid
       ge)}")
print(f"Exactitud en el conjunto de prueba para Lasso:
{accuracy_score(y_test, ␣
 ↪predicciones_las
       so)}")

print("\nPredicciones
Ridge:")
print(predicciones_ridg
e) print("\nPredicciones
Lasso:")
print(predicciones_lass
o)
```

```
Mejor valor de C para Ridge: 100
Mejor exactitud para Ridge: 0.9941176470588236

Mejor valor de C para Lasso: 10
Mejor exactitud para Lasso: 0.9941176470588236

Exactitud en el conjunto de prueba para Ridge: 0.9770114942528736
Exactitud en el conjunto de prueba para Lasso: 0.9885057471264368

Predicciones Ridge:
[1 1 0 1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 0
 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 0 1 0 1 0 1 1 1 0 1 1
 1 1 0 1 0 1 1 1 0 1 0 1 1]

Predicciones Lasso:
[1 1 0 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 0
 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 0 1 0 1 0 1 1 1 0 1 1
 1 1 0 1 0 1 1 1 0 1 0 1 1]
```

[37]:
```
# Reporte de clasificación para el modelo Ridge
print("Reporte de clasificación para el modelo
Ridge:") print(classification_report(y_test,
predicciones_ridge))

# Reporte de clasificación para el modelo Lasso

print("\nReporte de clasificación para el modelo Lasso:")
print(classification_report(y_test, predicciones_lasso))
```

```
Reporte de clasificación para el modelo Ridge:
```

```
              precision    recall  f1-score   support

           0       0.89      1.00      0.94        17
           1       1.00      0.97      0.99        70

    accuracy                           0.98
87 macro avg       0.95      0.99      0.96
87 weighted avg       0.98      0.98      0.98
                                           87
```

```
Reporte de clasificación para el modelo Lasso:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        17
           1       1.00      0.99      0.99        70

    accuracy                           0.99
87 macro avg       0.97      0.99      0.98
87 weighted avg       0.99      0.99      0.99
                                           87
```

[38]:
```
# Validación cruzada para el modelo
                Ridge
ridge_cv_scores = cross_val_score(mejor_modelo_ridge, X, y, cv=5)
print(f"Exactitud media del modelo Ridge con validación cruzada:␣
  ↪{ridge_cv_scores.mean():.2f}")


# Validación cruzada para el modelo Lasso
lasso_cv_scores = cross_val_score(mejor_modelo_lasso, X, y, cv=5)
print(f"Exactitud media del modelo Lasso con validación cruzada:␣
  ↪{lasso_cv_scores.mean():.2f}")
print('\n')

f1_scorer = make_scorer(f1_score)

# F1 score para el modelo Ridge
ridge_cv_scores_f1 = cross_val_score(mejor_modelo_ridge, X, y, cv=5,␣
  ↪scoring=f1_scorer)
print(f"F1-score medio del modelo Ridge con validación cruzada:␣
  ↪{ridge_cv_scores_f1.mean():.2f}")


# F1 score para el modelo Lasso
```

```
lasso_cv_scores_f1 = cross_val_score(mejor_modelo_lasso, X, y, cv=5,␣
  ↪scoring=f1_scorer)


print(f"F1-score medio del modelo Lasso con validación cruzada:␣
  ↪{lasso_cv_scores_f1.mean():.2f}")
```

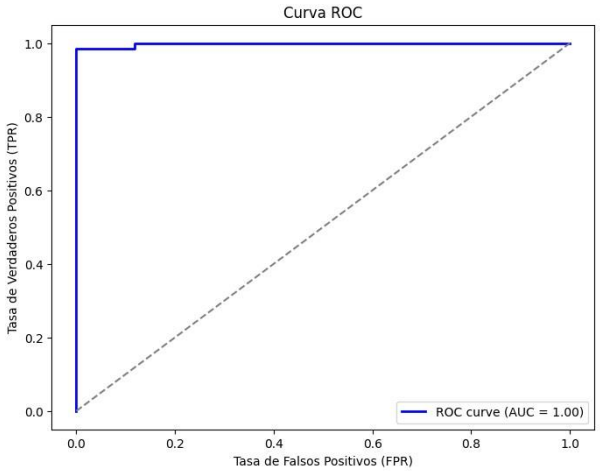
```
Exactitud media del modelo Ridge con validación cruzada: 0.91
Exactitud media del modelo Lasso con validación cruzada: 0.98



F1-score medio del modelo Ridge con validación cruzada: 0.94
F1-score medio del modelo Lasso con validación cruzada: 0.99
```

[39]:
```
# Evaluando predicciones
cm = confusion_matrix(y_test, predicciones_ridge)
print('Ridge') print(cm)
accuracy=accuracy_score(y_test,predicciones_
ridge) print("Exactitud: %.2f%%" % (accuracy
* 100.0)) print('\nLasso')
cm = confusion_matrix(y_test,
predicciones_lasso) print(cm)
accuracy=accuracy_score(y_test,predicciones_
lasso) print("Exactitud: %.2f%%" % (accuracy
* 100.0))
```

```
Ridge
[[17  0]
 [ 2 68]]
Exactitud:
97.70%

Lasso
[[17  0]
 [ 1 69]]
Exactitud:
98.85%
```

[40]:
```
# Probabilidad de la clase
        positiva
y_scores = mejor_modelo_ridge.predict_proba(X_test_esc_df)[:, 1]

# Calcular la curva ROC
fpr, tpr, _ = roc_curve(y_test, y_scores)
roc_auc = roc_auc_score(y_test, y_scores)
```

```
# Graficar la curva ROC
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC =
{roc_auc:.2f})') plt.plot([0, 1], [0, 1], color='gray', linestyle='--') #
Línea diagonal plt.xlabel('Tasa de Falsos Positivos (FPR)')
plt.ylabel('Tasa de Verdaderos Positivos (TPR)')
plt.title('Curva ROC')
plt.legend(loc='lower right')

plt.show()
```



4.1.1 Pruebas del modelo de regresión logística

```
[41]: # Crear un DataFrame con las columnas necesarias
columnas = [
    'd_semestres', 'p_desercion', 'p_graduados', 'c_docentes',
    'p_utilidad',␣
  C 'ingresos_totales','mod_en_linea', 'mod_hibrida', 'mod_presencial'
]
# Generar valores para realizar la prueba
data_prueba = np.array([
    [4, 10, 30, 8500, 8, 15000, 1, 0, 0], # Carrera
    1  [4,  10,  30,  8500,  8,  15000,  0,  1,  0], #
    Carrera 2 [5, 10, 28, 8500, 8, 15000, 0, 0,
    1], # Carrera 3
])

# Convertir a DataFrame antes de escalar
df_prueba = pd.DataFrame(data_prueba, columns=columnas)


# Cargar el scaler entrenado previamente
```

```
with open('scaler1.pkl', 'rb') as scaler_file:
    scaler1 = pickle.load(scaler_file)

# Escalar los datos con el scaler original
df_prueba_scaled = scaler1.transform(df_prueba)

# Convertir nuevamente a DataFrame para mantener las columnas
df_prueba_scaled = pd.DataFrame(df_prueba_scaled, columns=columnas)

# Realizar preddciones
y_pred_ridge = mejor_modelo_ridge.predict(df_prueba_scaled)

# Calcular probabilidades
y_prob = mejor_modelo_ridge.predict_proba(df_prueba_scaled)[:, 1]
print(y_prob)

print(pd.DataFrame({'Feature': columnas, 'Coef':
mejor_modelo_ridge.coef_[0]}))
print('\n')

# Mostrar resultados
for i, ridge_pred in enumerate(y_pred_ridge):
    print(f"Carrera {i+1}: Ridge: {'Rentable' if ridge_pred == 1 else
    'No␣
  ↪rentable'}")
```

```
[0.0022355  0.50866944
            0.51505663]
            Feature
                        Co
            ef
0       d_semestres   3.837381
1       p_desercion  -0.771293
2       p_graduados   0.445370
3        c_docentes   0.454344
4        p_utilidad  16.600200
5  ingresos_totales  -1.353418
6      mod_en_linea  -0.035372
7       mod_hibrida   1.365312
8    mod_presencial   0.541098

Carrera 1: Ridge: No
rentable Carrera 2:
Ridge: Rentable Carrera
3: Ridge: Rentable
```

4.2 Randon Forest

```
[42]: # Obtenemos la variabe dependiente y las
                independientes
 X = df_ranfor.drop(columns=['rentable'],axis=1) # Variables independientes
 y = df_ranfor['rentable'] # Variable dependiente

 print(X)
 print(y)
```

```
     d_semestres  p_desercion  p_graduados  c_docentes  p_utilidad  \
0              6        10.13         5.59    81263.87       52.56
1              6         6.46        12.91    91791.64       47.83
2              6         4.84        14.95    78673.22       53.64
3              6         7.22         8.06    77715.66       43.84
4              6         8.51        10.68    82553.70       48.35
..           …           …           …          …          …
283            4       100.00         0.00        0.00       65.91
284            6         0.00        94.44     7716.76       40.96
285            4        24.14         0.00    16213.50      -10.99
286            4        35.29         0.00     4832.45       50.35
287            4        21.60        36.00    12879.26       47.72

     ingresos_totales otros_costos
0           618706.69    248752.08
1           622513.19    264679.17
2           357951.45    137552.09
3           240916.44    104250.00
4           427880.01    187070.83
..                …           …
283             63.12       366.91
284          15853.78      6604.48
285          17635.32     10640.55
286          18325.48     12475.13
287          82159.34     45864.46

[288 rows x 7 columns]
0      1
1      1
2      1
3      1
4      1
      ..
283    1
284    1
285    0
286    1
287    1
```

```
Name: rentable, Length: 288, dtype: int64
```

[43]:
```
# Calcular valores de  Factor de Inflación de la Varianza (VIF - Variance
Inflation Factor)
# VIF < 5 -> Sin problemas de multicolinealidad.
# VIF entre 5 y 10 -> Multicolinealidad moderada (posible problema).
# VIF > 10 -> Multicolinealidad severa (es necesario corregir).

def calculate_vif(df):
        vif =
    pd.DataFrame()
    vif["variables"] = df_ranfor.columns
    vif["VIF"] = [variance_inflation_factor(df_ranfor.values, i) for i
    in
    range(df_ranfor.shape
        [1])]
    return vif

# Seleccionar las variables independientes (features)
vif = calculate_vif(X)
print(vif)
#df.head()
```

```
         variables       VIF
0       d_semestres  20.158523
1       p_desercion   2.020605
2       p_graduados   2.214336
3        c_docentes   7.967819
4        p_utilidad  12.760897
5          rentable  18.308114
6  ingresos_totales  47.863635
7      otros_costos  42.880662
```

[44]:
```
# Dividimos la data en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)


print("Distribución en y_train:", Counter(y_train))
print("Distribución en y_test:", Counter(y_test))
```

```
Distribución en y_train: Counter({1: 170, 0:
31}) Distribución en y_test: Counter({1: 70,
0: 17})
```

[45]:
```
# Utilizamos SMOTE para balancear las clases ya que tenemos pocos datos y queremoss crear más
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train,
y_train)

y_train_resampled = y_train_resampled.astype(int)

print("Antes de SMOTE:", Counter(y_train))
print("Después de SMOTE:", Counter(y_train_resampled))
```

```
Antes de SMOTE: Counter({1: 170, 0:
31}) Después de SMOTE: Counter({0: 170,
1: 170})
```

[46]:
```
# Escalar los datos numéricos
scaler2 = StandardScaler()

X_train_resam_df = pd.DataFrame(X_train_resampled, columns=X.columns)

# Escalar X_train
X_train_esc = scaler2.fit_transform(X_train_resam_df)
X_train_esc_df = pd.DataFrame(X_train_esc,
columns=X.columns)

# Guardar escalado original para las pruebas
with open('scaler2.pkl', 'wb') as scaler_file:
    pickle.dump(scaler2, scaler_file)

X_train_esc_df.head()
```

[46]:

	d_semestres	p_desercion	p_graduados	c_docentes	p_utilidad	ingresos_totales	otros_costos
0	-0.901364	0.223789	-0.594486	-0.892124	-1.356147	-1.159113	-1.177189
1	1.492884	-0.589426	0.844563	-0.699963	1.203883	-0.323886	-0.334163
2	0.295760	-0.637632	0.529788	0.731930	-0.532550	0.064350	-0.159701
3	0.295760	-0.395554	1.848146	0.281279	0.031053	-0.143969	-0.424159
4	0.295760	0.058212	-0.594486	-0.325492	0.787891	0.856032	1.461434

[47]:
```
# Escalar X_test
```

```
X_test_esc = scaler2.transform(X_test)
X_test_esc_df = pd.DataFrame(X_test_esc,
columns=X_test.columns) X_test_esc_df.head()
```

[47]:

	d_semestres	p_desercion	p_graduados	c_docentes	p_utilidad	\
0	1.492884	-0.364115	-0.264489	0.160126	1.012996	
1	1.492884	-0.834648	0.656459	-1.075987	1.670197	
2	1.492884	0.303434	-0.094326	0.912008	-1.100513	
3	1.492884	-0.688982	-0.180223	-0.079867	1.546741	
4	0.295760	-0.016193	-0.349843	-0.047204	-0.010211	

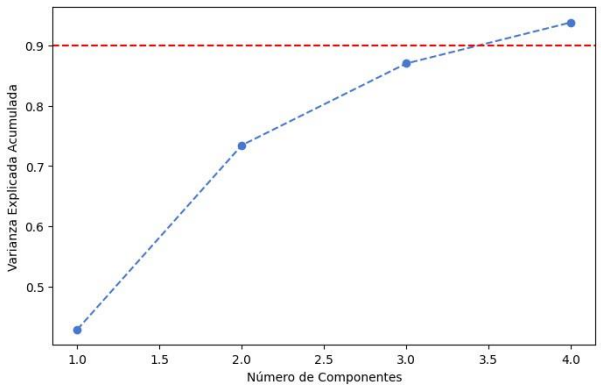
	ingresos_totales	otros_costos
0	0.579597	0.313256
1	-0.668181	-0.822439
2	-0.209710	-0.230681
3	1.228773	0.258956
4	0.241068	0.895887

[48]:
```
# Aplicar PCA
pca = PCA(n_components = 4)  # Ajustar el número de componentes según
sea␣
  ↪necesario
X_train_pca  = pca.fit_transform(X_train_esc_df)

with open('pca1.pkl', 'wb') as pca_file:
    pickle.dump(pca, pca_file)

# Imprimir la cantidad de varianza explicada por cada componente principal
# Toma el número mínimo de componentes que expliquen al menos 90% de la␣
  ↪varianza.
print("Varianza explicada por cada componente principal:", pca.
  ↪explained_variance_ratio_)
print("Varianza explicada total:", sum(pca.explained_variance_ratio_))
```

```
Varianza explicada por cada componente principal: [0.42908072 0.30527244
0.13547202 0.0679506 ]
Varianza explicada total: 0.9377757725358999
```

[49]:
```
# Calcular varianza explicada acumulada
explained_variance_ratio = np.cumsum(pca.explained_variance_ratio_)

# Graficar la varianza explicada acumulada
plt.figure(figsize=(8,5))
plt.plot(range(1, len(explained_variance_ratio)+1),
explained_variance_ratio,␣
```

```
  C marker='o', linestyle='--')
plt.xlabel('Número de Componentes')
plt.ylabel('Varianza Explicada
Acumulada')
#plt.title('Selección del Número Óptimo de Componentes')
plt.axhline(y=0.90, color='r', linestyle='--') # Línea de referencia en
95%
plt.show()
```



[50]:
```
# Crear y entrenar el modelo de Random
                Forest
modelo_ranfor = RandomForestClassifier(n_estimators=100, random_state=42)
modelo_ranfor.fit(X_train_pca, y_train_resampled)

# Aplicar PCA al conjunto de prueba escalado
X_test_pca = pca.transform(X_test_esc_df)

# 7. Predecir en el conjunto de prueba
y_pred_pca = modelo_ranfor.predict(X_test_pca)

# Evaluar el rendimiento del modelo
print("Exactitud en el conjunto de prueba:", accuracy_score(y_test,
y_pred_pca))
print("\nMatriz de Confusión:")
print(confusion_matrix(y_test,
y_pred_pca)) print("\nReporte de
Clasificación:")
print(classification_report(y_test,
y_pred_pca))
```

```
Exactitud en el conjunto de prueba: 0.9195402298850575
```

```
Matriz de
Confusión:
[[12  5]
 [ 2 68]]

Reporte de Clasificación:

              precision    recall  f1-score   support

           0       0.86      0.71      0.77        17
           1       0.93      0.97      0.95        70

    accuracy                           0.92
   87 macro avg       0.89      0.84      0.86
87 weighted avg       0.92      0.92      0.92
                                           87
```

[51]:
```
# Evaluación
print("Accuracy:", accuracy_score(y_test,
y_pred_pca)) print("Precision:",
precision_score(y_test, y_pred_pca))
print("Recall:", recall_score(y_test,
y_pred_pca)) print("F1-Score:", f1_score(y_test,
y_pred_pca))
```

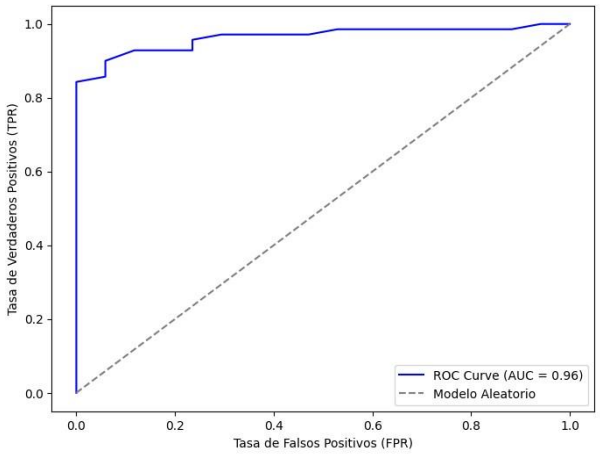
```
Accuracy: 0.9195402298850575
Precision: 0.9315068493150684
Recall: 0.9714285714285714
F1-Score: 0.951048951048951
```

[52]:
```
y_prob_pca =
modelo_ranfor.predict_proba(X_test_pca)[:, 1]

# Calcular ROC-AUC
roc_auc_general = roc_auc_score(y_test, y_prob_pca)
print(f"ROC-AUC General: {roc_auc_general:.4f}")
```

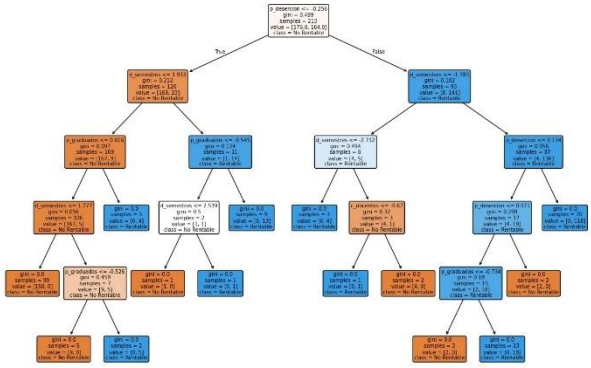
```
ROC-AUC General: 0.9639
```

[53]:
```
# Calcular la curva ROC
fpr, tpr, thresholds = roc_curve(y_test, y_prob_pca)
roc_auc_value = auc(fpr, tpr)  # Usar un nombre diferente para evitar
conflictos

# Graficar la Curva ROC
plt.figure(figsize=(8, 6))
```

```
plt.plot(fpr, tpr, color="blue", label=f'ROC Curve (AUC =
{roc_auc_value:.2f})')
plt.plot([0, 1], [0, 1], color="gray", linestyle="--", label="Modelo␣
    ↪Aleatorio") #
      Línea base
plt.xlabel("Tasa de Falsos Positivos (FPR)")
plt.ylabel("Tasa de Verdaderos Positivos (TPR)")
#plt.title("Curva ROC para el modelo Random Forest")
plt.legend(loc="lower right")
plt.show()
```



[54]:
```
# Visualizar el primer árbol del
    bosque
from sklearn.tree import plot_tree

plt.figure(figsize=(15, 10))
plot_tree(modelo_ranfor.estimators_[0], filled=True,
          feature_names=X_train_esc_df.columns, # Changed to use
          original␣
  ↪feature names
          class_names=['No Rentable', 'Rentable'], rounded=True) #
          Provide␣
  ↪class names
plt.show()
```



[90]:
```
# Crear un DataFrame con las columnas necesarias
columnas = [
    'd_semestres', 'p_desercion', 'p_graduados', 'c_docentes',␣
  ↪'p_utilidad','ingresos_totales','otros_costos'
]
# Generar valores para realizar la prueba
data_prueba = np.array([
    [5, 6.5, 30, 100, 10, 150, 120], # Carrera
    1  [4, 6.5, 30, 100, 30, 150, 120],   #
    Carrera 2 [4, 6.5, 35, 100, 21, 150, 120],
    # Carrera 3
])

# Convertir a DataFrame antes de escalar
df_prueba = pd.DataFrame(data_prueba, columns=columnas)

# Cargar el scaler entrenado previamente
with open('scaler2.pkl', 'rb') as scaler_file:
                scaler2 =
        pickle.load(scaler_file)

# Escalar los datos con el scaler original
df_prueba_scaled = scaler2.transform(df_prueba)

# Convertir nuevamente a DataFrame para mantener las columnas

df_prueba_scaled = pd.DataFrame(df_prueba_scaled, columns=columnas)

# Cargar el PCA entrenado previamente
with open('pca1.pkl', 'rb') as pca_file:
    pca_rf = pickle.load(pca_file)

# Aplicar la transformación PCA
nueva_carrera_pca = pca_rf.transform(df_prueba_scaled)
```

```
# Realizar predicciones
y_pred_rf = modelo_ranfor.predict(nueva_carrera_pca)

# Calcular probabilidades
y_prob = modelo_ranfor.predict_proba(nueva_carrera_pca)[:, 1]
print(y_prob)

umbral = 0.5
y_pred_rf = (y_prob >= umbral).astype(int)

# Mostrar resultados
for i, rf_pred in enumerate(y_pred_rf):
    print(f"Carrera {i+1}: {'Rentable' if rf_pred == 1 else 'No
    rentable'}")

importances = modelo_ranfor.feature_importances_
for feature, importance in zip(X.columns, importances):
    print(f"{feature}: {importance}")
```

```
[0.94 0.89
0.92] Carrera
1: Rentable
Carrera 2:
Rentable
Carrera 3:
Rentable
i_matricula_log:
0.10475323490076602
i_otros_log:
0.6365690155140227
c_docentes_log:
0.21186249333582766
c_adminis_log:
0.04681525624938368
```

4.3 Percepción multicapa (MLP)

[56]:
```
# Obtenemos la variabe dependiente y las independientes
X = df_mlp.drop(columns=['rentable'],axis=1) # Variables independientes
y = df_mlp['rentable'] # Variable dependiente
print(X)
print(y)
```

```
    d_semestres  p_desercion  p_graduados  c_docentes  c_adminis
    c_otros  \
```

```
0              6        10.13         5.59    81263.87  108641.85
140110.23
1              6         6.46        12.91    91791.64  115597.97
149081.20
2              6         4.84        14.95    78673.22   60075.53
77476.56
3              6         7.22         8.06    77715.66   45530.93
58719.07
4              6         8.51        10.68    82553.70   81702.72
105368.11
..           …           …           …           …         …         …
283            4       100.00         0.00        0.00     190.90
176.01
284            6         0.00        94.44     7716.76    3436.27
3168.21
285            4        24.14         0.00    16213.50    5536.22
5104.33
286            4        35.29         0.00     4832.45    6490.74
5984.39
287            4        21.60        36.00    12879.26   23863.02
22001.44

     p_utilidad  ingresos_totales
0         52.56         618706.69
1         47.83         622513.19
2         53.64         357951.45
3         43.84         240916.44
4         48.35         427880.01
..          …               …
283       65.91            63.12
284       40.96         15853.78
285     -10.99          17635.32
286       50.35         18325.48
287       47.72         82159.34

[288 rows x 8 columns]
0      1
1      1
2      1
3      1
4      1
      ..
283    1
284    1
285    0
286    1
287    1
Name: rentable, Length: 288, dtype: int64
```

[57]:
```
# Dividir el dataset en entrenamiento y
                  prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
  ↪random_state=42)
print(f"Tamaño del conjunto de entrenamiento: {X_train.shape[0]}
muestras")
print(f"Tamaño del conjunto de prueba: {X_test.shape[0]} muestras")
```

```
Tamaño del conjunto de entrenamiento: 201 muestras
Tamaño del conjunto de prueba: 87 muestras
```

[58]:
```
# Utilizamos SMOTE para balancear las clases ya que tenemos pocos datos
y␣
  ↪queremoss crear más
#smote = SMOTE(random_state=42)
#X_train_resampled, y_train_resampled = smote.fit_resample(X_train,
y_train)

#print("Antes de SMOTE:", Counter(y_train))
#print("Después de SMOTE:", Counter(y_train_resampled))
```

[59]:
```
from sklearn.preprocessing import
RobustScaler scaler3 = RobustScaler()

# Ajustar el escalador con los datos de entrenamiento y transformar
X_train = scaler3.fit_transform(X_train)

# Transformar el conjunto de prueba con el mismo escalador
X_test = scaler3.transform(X_test)
print(X_train)
```

```
[[-1.          1.04545455 -0.47022121 … -0.67341029 -2.22104916
      -
  0.697337
    16]
 [ 1.         -0.31118881  1.03119682 … -0.03658472  0.29660178
      -
  0.176878
    95]
 [ 0.         -0.39160839  0.70277935 … -0.17196171 -1.41108545
   0.06504
    424]
 …
```

```
[ 0.          0.70104895 -0.47022121 …  0.71870801  0.16628176
  0.33605
   493]
[-1.         -1.11188811  2.18150879 … -0.59740271  0.58396569
    -
 0.510095
   71]
[ 1.         -0.27447552  0.88428815 …  1.35203534 -1.12438139
  0.70846235]]
```

[60]:
```
# Definir la arquitectura del modelo
model = Sequential()
Input(shape=(X.shape[1],)),
model.add(Dense(16,
activation='relu'))
model.add(Dense(8,
activation='relu'))
model.add(Dense(1,
activation='sigmoid'))

# Compilar el modelo
model.compile(loss='binary_crossentropy', optimizer='adam',␣
  ↪metrics=['accuracy'])


# Entrenar el modelo
# class_weights = {0: 1.5, 1: 1.0}
history = model.fit(X_train, y_train, epochs=20, batch_size=10,␣
  ↪validation_data=(X_test, y_test))
```

```
Epoch 1/20
21/21  2s 16ms/step -
accuracy: 0.6664 - loss: 0.6802 - val_accuracy: 0.7126 - val_loss: 0.6601
Epoch 2/20
21/21  0s 7ms/step -
accuracy: 0.7372 - loss: 0.6254 - val_accuracy: 0.7816 - val_loss: 0.6214
Epoch 3/20
21/21  0s 7ms/step -
accuracy: 0.7892 - loss: 0.5861 - val_accuracy: 0.8161 - val_loss: 0.5852
Epoch 4/20
21/21  0s 6ms/step -
accuracy: 0.8403 - loss: 0.5296 - val_accuracy: 0.8046 - val_loss: 0.5511
Epoch 5/20
21/21  0s 7ms/step -
accuracy: 0.8375 - loss: 0.4833 - val_accuracy: 0.8046 - val_loss: 0.5162
Epoch 6/20
21/21  0s 6ms/step -
```

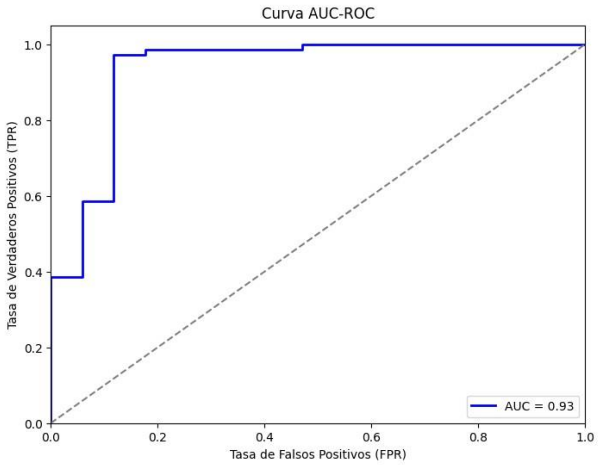
```
accuracy: 0.8556 - loss: 0.4386 - val_accuracy: 0.8506 - val_loss: 0.4768
Epoch 7/20
21/21  0s 7ms/step -
accuracy: 0.9119 - loss: 0.4038 - val_accuracy: 0.8621 - val_loss: 0.4360
Epoch 8/20
21/21  0s 6ms/step -
accuracy: 0.9171 - loss: 0.3488 - val_accuracy: 0.8621 - val_loss: 0.4003
Epoch 9/20
21/21  0s 7ms/step -
accuracy: 0.9319 - loss: 0.3043 - val_accuracy: 0.8851 - val_loss: 0.3683
Epoch 10/20
21/21  0s 6ms/step -
accuracy: 0.9364 - loss: 0.2879 - val_accuracy: 0.9310 - val_loss: 0.3310
Epoch 11/20
21/21  0s 6ms/step -
accuracy: 0.9603 - loss: 0.2282 - val_accuracy: 0.9425 - val_loss: 0.3069
Epoch 12/20
21/21  0s 7ms/step -
accuracy: 0.9721 - loss: 0.2249 - val_accuracy: 0.9425 - val_loss: 0.2894
Epoch 13/20
21/21  0s 5ms/step -
accuracy: 0.9570 - loss: 0.2049 - val_accuracy: 0.9310 - val_loss: 0.2746
Epoch 14/20
21/21  0s 9ms/step -
accuracy: 0.9535 - loss: 0.1778 - val_accuracy: 0.9310 - val_loss: 0.2638
Epoch 15/20
21/21  0s 9ms/step -
accuracy: 0.9642 - loss: 0.1529 - val_accuracy: 0.9425 - val_loss: 0.2513
Epoch 16/20
21/21  0s 9ms/step -
accuracy: 0.9682 - loss: 0.1383 - val_accuracy: 0.9425 - val_loss: 0.2437


    Epoch 17/20
    21/21  0s 9ms/step -
    accuracy: 0.9799 - loss: 0.1269 - val_accuracy: 0.9540 - val_loss: 0.2287
    Epoch 18/20
    21/21  0s 9ms/step -
    accuracy: 0.9708 - loss: 0.1228 - val_accuracy: 0.9540 - val_loss: 0.2252
    Epoch 19/20
    21/21  0s 9ms/step -
    accuracy: 0.9764 - loss: 0.1182 - val_accuracy: 0.9540 - val_loss: 0.2210
    Epoch 20/20
    21/21  0s 10ms/step -
    accuracy: 0.9748 - loss: 0.1139 - val_accuracy: 0.9540 - val_loss: 0.2160
```

```
[61]: # Evaluar el modelo con los datos de prueba
      loss, accuracy = model.evaluate(X_test,
      y_test) print(f'Precisión del modelo:
      {accuracy*100:.2f}%')

      3/3  0s 23ms/step -
      accuracy: 0.9536 - loss:
      0.2203
      Precisión del modelo: 95.40%

 [62]: # Realizar predicciones en el conjunto de
                        prueba
      y_pred_probs = model.predict(X_test) # Predicciones en probabilidades
      y_pred = (y_pred_probs > 0.5).astype("int32")

      # 1. Matriz de Confusión
      cm = confusion_matrix(y_test, y_pred)

      # 2. Clasificación: Precision, Recall, F1-score
      classification_rep = classification_report(y_test, y_pred,
      output_dict=True)

      # 3. AUC-ROC
      auc = roc_auc_score(y_test, y_pred_probs)

      # Crear un DataFrame para visualizar mejor los resultados
      eval_results = {
          "Métrica": ["Precisión", "Recall", "F1-score", "AUC-
          ROC"], "Clase 0": [
              classification_rep["0"]["precisio
              n"],
              classification_rep["0"]["recall
              "], classification_rep["0"]["f1-
              score"],
              auc  # AUC-ROC es un solo valor, no depende de clases individuales
          ],
          "Clase 1": [
              classification_rep["1"]["precisio
              n"],
              classification_rep["1"]["recall
              "], classification_rep["1"]["f1-
              score"],
              auc  # AUC-ROC es el mismo para ambas clases
          ]
      }

      df_eval_results = pd.DataFrame(eval_results)
```

```
# Mostrar el DataFrame
correctamente print("Evaluación
del Modelo:")
print(df_eval_results)

# Imprimir la matriz de
confusión print("\nMatriz de
Confusión:") print(cm)
```

```
3/3  0s 40ms/step
Evaluación del Modelo:
     Métrica   Clase 0   Clase 1
0  Precisión  0.882353  0.971429
1     Recall  0.882353  0.971429
2   F1-score  0.882353  0.971429
3    AUC-ROC  0.933613  0.933613

Matriz de
Confusión: [[15
2]
 [ 2 68]]
```

[63]:
```
# Calcular la curva ROC
fpr, tpr, _ = roc_curve(y_test, y_pred_probs)
roc_auc = roc_auc_score(y_test, y_pred_probs)
#roc_auc = auc(fpr, tpr)

# Graficar la curva AUC-ROC
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'AUC = {roc_auc:.2f}')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--') # Línea diagonal
de␣
  referencia
plt.xlim([0.0,
1.0])
plt.ylim([0.0,
1.05])
plt.xlabel('Tasa de Falsos Positivos (FPR)')
plt.ylabel('Tasa de Verdaderos Positivos (TPR)')
plt.title('Curva AUC-
ROC')
plt.legend(loc="lower
right") plt.show()
```



[64]:
```
# d_semestres, p_desercion, p_graduados, c_docentes, c_adminis,␣
  ↪c_otros        p_utilidad, ingresos_totales
nuevas_carreras = np.array([
    [4, 10, 40, 1000, 5000, 3000, 25, 1500], #
    Carrera 1 [4, 15, 18, 9500, 4500, 2000, 23,
    1500], #  Carrera 2 [4, 22, 10, 6500, 2500, 1200,
    8, 2400]  # Carrera 3
])
df_prueba = pd.DataFrame(nuevas_carreras,
columns=X.columns) X_example =
scaler3.transform(df_prueba)
# Realizar predicciones con el modelo
y_example_probs = model.predict(X_example)  # Probabilidades
y_example_preds = (y_example_probs > 0.5).astype("double")

# Mostrar los resultados
for i, (prob, pred) in enumerate(zip(y_example_probs, y_example_preds)):
    print(f"Ejemplo {i+1}: Probabilidad de ser rentable = {prob[0]:.2f}
                                                    ->␣
  ↪Predicción = {'Rentable (1)' if pred[0] == 1 else 'No Rentable (0)'}")
```

```
1/1  0s 56ms/step
Ejemplo 1: Probabilidad de ser rentable = 0.99 -> Predicción = Rentable
(1) Ejemplo 2: Probabilidad de ser rentable = 0.93 -> Predicción =
Rentable (1) Ejemplo 3: Probabilidad de ser rentable = 0.60 ->
Predicción = Rentable (1)
```

4.4 Regresión lineal multiple

[65]:
```
# Cargar datos
```

```
data = df.copy()

# Definir variables a evaluar para outliers antes de la transformación
variables_originales = ['i_matricula', 'i_otros', 'c_docentes', 'c_adminis',
    'c_otros']


# Método de eliminación de outliers usando IQR (Interquartile Range) en los
    datos originales
Q1 = data[variables_originales].quantile(0.25)
Q3 = data[variables_originales].quantile(0.75)
IQR = Q3 - Q1

# Filtrar los datos eliminando outliers ANTES de la transformación logarítmica
data_cleaned = data[~((data[variables_originales] < (Q1 - 1.5 *
                      IQR)) | (data[variables_originales] > (Q3 +
                      1.5 * IQR))).
    any(axis=1)].copy()


# Mostrar cantidad de datos antes y después de eliminar
outliers print(f"\nDatos antes de eliminar outliers:
{data.shape[0]}") print(f"Datos después de eliminar outliers:
{data_cleaned.shape[0]}")
```

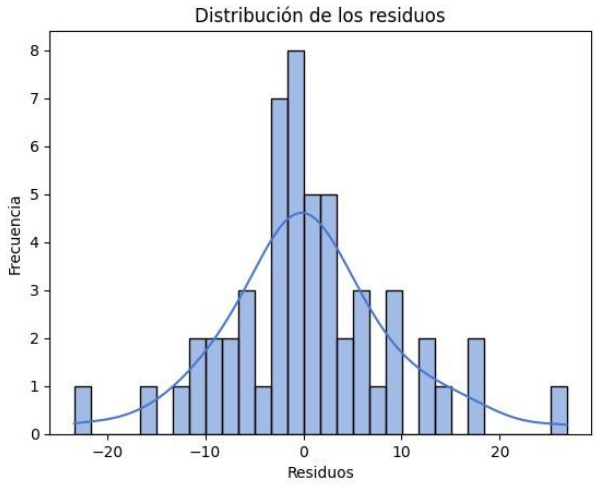
```
Datos antes de eliminar outliers: 288
Datos después de eliminar outliers: 262
```

[66]:
```
# Aplicar transformación logarítmica después de eliminar outliers
data_cleaned.loc[:, 'i_matricula_log'] =
np.log1p(data_cleaned['i_matricula']) data_cleaned.loc[:, 'i_otros_log']
= np.log1p(data_cleaned['i_otros']) data_cleaned.loc[:,
'c_docentes_log'] = np.log1p(data_cleaned['c_docentes'])
data_cleaned.loc[:, 'c_adminis_log'] =
np.log1p(data_cleaned['c_adminis']) data_cleaned.loc[:, 'c_otros_log']
= np.log1p(data_cleaned['c_otros'])

# Definir variables para el modelo
```

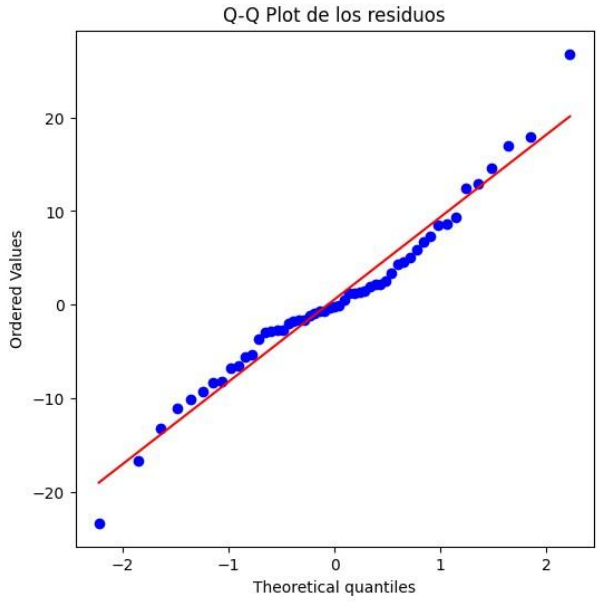
```
variables_transformadas = ['i_matricula_log', 'i_otros_log',
'c_docentes_log',␣
      'c_adminis_log',
        'c_otros_log']
X = data_cleaned[variables_transformadas]
y = data_cleaned['p_utilidad']

# Dividir en conjunto de entrenamiento y prueba



X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
  random_state=42)


# Ajustar el modelo de regresión
lineal model = LinearRegression()
model.fit(X_train, y_train)

# Predicciones
y_pred = model.predict(X_test)

# Evaluación del modelo
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

```
Mean Squared Error: 75.1766505436887
R-squared: 0.8931615542927687
```

[67]:
```
residuos = y_test - y_pred.flatten()

# Graficar el histograma de los
residuos sns.histplot(residuos,
bins=30, kde=True)
plt.xlabel("Residuos")
plt.ylabel("Frecuencia")
plt.title("Distribución de los
residuos") plt.show()
```



[68]:
```
import scipy.stats as stats

# Gráfico Q-Q plot (Quantile-Quantile)
plt.figure(figsize=(6, 6))
stats.probplot(residuos, dist="norm",
plot=plt) plt.title("Q-Q Plot de los
residuos")
plt.show()
```



[69]:
```
# Prueba de Shapiro-Wilk
stat, p = stats.shapiro(residuos)
print(f"Estadístico de Shapiro-Wilk: {stat:.4f}, p-valor: {p:.4f}")

# Interpretación del p-valor
```

```
    if p > 0.05:
        print("No se rechaza la hipótesis nula: los residuos parecen seguir
                                        una␣
      ↪distribución normal.")
    else:
        print("Se rechaza la hipótesis nula: los residuos NO siguen una␣
      ↪distribución normal.")
```

```
Estadístico de Shapiro-Wilk: 0.9732, p-valor: 0.2770
No se rechaza la hipótesis nula: los residuos parecen seguir una
distribución normal.
```

```
[70]: # Crear 3 nuevos registros con datos ficticios
    nuevos_datos = pd.DataFrame([
        [32000, 2800, 29000, 5900, 1800],   #
        Registro 1 [21000, 3000, 27000, 4500,
        2200], # Registro 2 [18000, 2500, 26000,
        4800, 2000], # Registro 3
    ], columns=['i_matricula', 'i_otros', 'c_docentes', 'c_adminis',
    'c_otros'])

    # Aplicar transformación logarítmica a los nuevos registros (como en el
    modelo) nuevos_datos['i_matricula_log'] =
    np.log1p(nuevos_datos['i_matricula']) nuevos_datos['i_otros_log'] =
    np.log1p(nuevos_datos['i_otros']) nuevos_datos['c_docentes_log'] =
    np.log1p(nuevos_datos['c_docentes']) nuevos_datos['c_adminis_log'] =
    np.log1p(nuevos_datos['c_adminis']) nuevos_datos['c_otros_log'] =
    np.log1p(nuevos_datos['c_otros'])

    # Seleccionar solo las columnas que usó el modelo
    X_nuevos = nuevos_datos[['i_matricula_log', 'i_otros_log',
    'c_docentes_log',␣
      ↪'c_adminis_log', 'c_otros_log']]

    # Hacer predicciones con el modelo entrenado (sin escalado)
    predicciones_nuevas = model.predict(X_nuevos)

    # Mostrar resultados
    print("Predicciones con LinearRegression")
    for i, pred in enumerate(predicciones_nuevas):
        print(f"Registro {i+1}: Predicción del porcentaje de utilidad =
        {pred:.2f}")
```

```
Predicciones con LinearRegression
Registro 1: Predicción del porcentaje de utilidad = 38.99
Registro 2: Predicción del porcentaje de utilidad = 28.13
Registro 3: Predicción del porcentaje de utilidad = 15.9
```

7.2. Acuerdo de confidencialidad

ACUERDO DE CONFIDENCIALIDAD

Este Acuerdo de Confidencialidad (en adelante, el "Acuerdo") es celebrado el 14 de marzo del 2025 entre:

INSTITUTO TECNOLÓGICO CORDILLERA, con domicilio en Avenida la Prensa y Logroño N45-268, representada por **Ms. David Andres Flores Torres**, en calidad de Rector, en adelante "**LA INSTITUCIÓN**";

Y

Jorge Armando Tatayo Chanaluisa, identificado con número de cedula **1715646897**, con domicilio en Gral. León de Febres Cordero N56-235 y José María Borrero, en adelante "**EL INVESTIGADOR**".

Ambas partes acuerdan lo siguiente:

1.OBJETO DEL ACUERDO

LA INSTITUCIÓN proporcionará acceso a ciertos datos e información (en adelante, **"INFORMACIÓN CONFIDENCIAL"**) a **EL INVESTIGADOR** con el propósito exclusivo de realizar estudios para el trabajo de titulación con el tema *Modelo predictivo para analizar la sostenibilidad financiera de las carreras de una institución de educación superior*, para la obtención del título de Magister en Sistemas de Información, mención en Data Science

2. DEFINICIÓN DE INFORMACIÓN CONFIDENCIAL

Se considerará **INFORMACIÓN CONFIDENCIAL** cualquier dato, documento, código, modelo, metodología, resultado de análisis o cualquier otra información relacionada con los registros académicos y financieros de **LA INSTITUCIÓN** a la que **EL INVESTIGADOR** tenga acceso en virtud de su investigación, salvo aquella que sea de dominio público.

Los datos específicos a los que **EL INVESTIGADOR** tendrá acceso incluyen registros de los periodos comprendidos entre los años 2015 y 2024, la información estará clasificada por carrera y por periodo académico, a continuación, se detalla los datos:

- Año.
- Periodo académico.
- Código carrera.
- Nombre carrera.
- Modalidad.
- Numero de semestres por carrera.
- Número de estudiantes matriculados.
- Porcentaje de deserción.
- Porcentaje de graduados.
- Valor total de ingresos por aranceles académicos.
- Valor total de otros ingresos como derechos, titulación, etc.
- Valor total del costo por concepto de docencia.
- Valor total del costo por concepto del personal administrativo y de servicios.
- Valor total de otros costos como servicios básicos, mantenimiento, etc.

3. OBLIGACIONES DE CONFIDENCIALIDAD

EL INVESTIGADOR se compromete a:

- No divulgar, compartir, copiar, reproducir ni utilizar la **INFORMACIÓN CONFIDENCIAL** para fines distintos a los autorizados por **LA INSTITUCIÓN**.
- Mantener la **INFORMACIÓN CONFIDENCIAL** en un entorno seguro y protegido contra accesos no autorizados.
- No transferir la **INFORMACIÓN CONFIDENCIAL** a terceros sin autorización expresa y por escrito de **LA INSTITUCIÓN**.

4. DURACIÓN

Este Acuerdo tendrá una duración indefinida y permanecerá en vigor incluso después de la finalización del trabajo de titulación de **EL INVESTIGADOR**.

5. CONSECUENCIAS DEL INCUMPLIMIENTO

En caso de incumplimiento de este Acuerdo, **EL INVESTIGADOR** será responsable por los daños y perjuicios ocasionados y podrá estar sujeto a acciones legales y/o disciplinarias por parte de **LA INSTITUCIÓN**.

6. LEGISLACIÓN APLICABLE

Este Acuerdo se regirá por las leyes del Ecuador y cualquier controversia será resuelta en los tribunales de Quito - Ecuador.

En señal de conformidad, las partes firman este Acuerdo en dos ejemplares el 13 de marzo del 2025.

INSTITUTO TECNOLÓGICO UNIVERSITARIO CORDILLERA

DAVID ANDRES FLORES TORRES

JORGE ARMANDO TATAYO CHANALUISA

Jorge Armando Tatayo Chanaluisa