Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática
MAESTRÍA EN INFORMÁTICA APLICADA

Implementación de un algoritmo de árbol de decisión para crear un sistema de predicción que determine las posibilidades de ingreso a la UdeG

Trabajo recepcional que para obtener el grado de
Maestro en Informática Aplicada

Presenta: **Luis Mario Morales Llamas**

Asesor: **Iván Esteban Villalón Turrubiates**

Tlaquepaque, Jalisco. 27 de junio de 2022.

AGRADECIMIENTOS

A mi esposa, Ale Arriola, que me ayudó a dar forma a este proyecto desde el inicio hasta el último punto y coma.

A mis padres, Irma y Leonardo, quienes siempre me alentaron a seguir adelante con mis planes.

A los profesores del ITESO que facilitaron el curso de esta maestría, por su guía y su compromiso.

A mi asesor, Iván Esteban Villalón, por su paciencia y ánimo para continuar creciendo y mejorando este trabajo y ayudarme a llevarlo a buen término.

Al Mtro. Ricardo Salas Mejía por sus atenciones y ayuda durante el curso de la maestría.

A todos quienes de alguna manera me ayudaron a finalizar este trabajo, compañeros y amigos, que cursamos los dos años de estudios durante un periodo que nos obligó a estar gran parte del tiempo en casa, tomando clases y trabajando virtualmente.

Al ITESO por otorgarme la beca de excelencia académica, que facilitó en gran medida el haber podido cursar esta maestría.

DEDICATORIA

A mi esposa, Ale Arriola, quien es mi compañera y amiga siempre, por estar conmigo en todo momento, escucharme y animarme a continuar con todo proyecto que emprenda. Por ser quien me da siempre la fuerza necesaria para todo.

RESUMEN

El presente trabajo presenta el desarrollo de un sistema de predicción que determina las posibilidades de ingreso a una universidad según datos estadísticos de promedios y demanda escolar de un periodo de diez años, utilizando un modelo de aprendizaje automático (*machine learning*) de árbol de decisión para realizar las predicciones.

El trabajo incluye el proceso de limpieza de datos así como el análisis inicial y preparación de los mismos. Se incluye una descripción del proceso a realizar y se ofrece una visión de trabajos similares o relacionados.

Además se describe todo el proceso de entrenamiento del modelo de aprendizaje automático, así como las pruebas realizadas para comprobar su funcionamiento.

Finalmente, se incluye el proceso de desarrollo de la interfaz gráfica para habilitar a los usuarios finales a interactuar con el modelo mediante una aplicación web. Se incluye un análisis de los resultados y directrices sobre el trabajo a futuro para mejorar o ampliar el proyecto o para generar nuevos proyectos en esta misma línea.

TABLA DE CONTENIDOS

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ACRÓNIMOS Y ABREVIATURAS

CLI	Interfaz de línea de comandos (*Command-line Interface*)
COVID-19	Enfermedad por coronavirus de 2019 causada por el virus SARS-CoV-2
CSS	Hojas de estilo en cascada (*Cascading Style Sheets*)
CSV	Valores separados por comas (*Comma Separated Values*)
CUCEI	Centro Universitario de Ciencias Exactas e Ingeniería
CUCS	Centro Universitario de Ciencias Sociales
EUA	Estados Unidos de América
HTML	Lenguaje de Marcas de Hipertexto (*HyperText Markup Language*)
HTTP	Protocolo de transferencia de hipertexto (*Hypertext Transfer Protocol*)
INEGI	Instituto Nacional de Estadística y Geografía
ITESO	Instituto Tecnológico y de Estudios Superiores de Occidente
NLP	Procesamiento de Lenguaje Natural (*Natural Language Processing*)
SEP	Secretaría de Educación Pública
SQL	Lenguaje de Consulta Estructurada (*Structured Query Language*)
UdeG	Universidad de Guadalajara
URL	Localizador uniforme de recursos (*Uniform Resource Locator*)
WSGI	Interfaz de enlace para servidor web (*Web Server Gateway Interface*)

1. INTRODUCCIÓN

Resumen: *En esta sección se presenta la información acerca del propósito de este trabajo, que es el de crear un sistema de predicción para encontrar las probabilidades de que un candidato sea admitido a la Universidad de Guadalajara. También se habla de los antecedentes existentes en esta área así como de la relevancia tecnológica de este tipo de análisis.*

1.1. Antecedentes

Este trabajo pretende desarrollar un sistema que determine la posibilidad de que algún candidato aspirante a la educación universitaria ingrese a la Universidad de Guadalajara (UdeG) en sus distintos centros universitarios, utilizando un modelo de aprendizaje automático conocido como árbol de decisión.

La UdeG proporciona en su sitio web datos estadísticos acerca de los alumnos que ingresan a la universidad, por lo que se pretenden utilizar esos datos para alimentar el algoritmo de aprendizaje automático. Los datos estadísticos de la UdeG pueden encontrarse en la siguiente liga: http://www.escolar.udg.mx/estadisticas.

Estas estadísticas son utilizadas por algunos medios para difundir, por ejemplo, las tasas de rechazo de la UdeG año con año, o los porcentajes de ingreso. Asimismo, son utilizadas por los centros de capacitación o nivelación para estudiantes que desean hacer el examen de admisión, para promover sus cursos entre los estudiantes interesados en prepararse para dicho examen. Sin embargo, estos datos no han sido utilizados en aplicaciones de predicción o de aprendizaje automático, por lo menos, no de manera pública o comercial. Del mismo modo, no son muchas las instituciones que ofrecen de manera pública datos estadísticos sobre educación más allá de algunos datos demográficos, por lo que esta fuente resulta de utilidad para el objetivo de este trabajo.

Por otra parte, parecen no existir modelos predictivos o de aprendizaje automático que se hayan aplicado para universidades mexicanas, aún a pesar de que México tiene algunas de las universidades más importantes de América Latina y exista, por tanto, una gran demanda de ingreso; en este contexto, muchas veces los alumnos que están por egresar del bachillerato no saben qué carrera elegir y terminan eligiendo entre las carreras más populares, en muchas ocasiones sin haber analizado mucho la situación.

Sí existen, sin embargo, aplicaciones de modelos predictivos en el ámbito local aplicados a otros temas, distintos del tema de interés de este trabajo, pero que por utilizar los mismos enfoques o similares, pueden resultar de interés, tales como:

- Modelo de predicción de casos de COVID-19 en México (https://www.sciencedirect.com/science/article/abs/pii/S0960077920303453). Trabajo que

presenta el modelado y predicción de casos de infecciones del virus COVID-19 en México a través de modelos matemáticos (Gompertz y logístico) y computacionales (redes neuronales artificiales).

- Plataforma de monitoreo y pronóstico de casos de COVID-19 en México (https://monitoreocovid.lasalle.mx/#mx,us). Plataforma desarrollada por la Universidad La Salle para el monitoreo y pronóstico de la evolución del COVID-19 en México utilizando algoritmos de inteligencia artificial para analizar la información publicada por la Secretaria de Salud predecir la evolución del número de casos confirmados.
- Pronóstico de precios de petróleo mediante redes neuronales (https://www.jstor.org/stable/26201481). Trabajo que muestra el uso de redes neuronales artificiales para el pronóstico de precios futuros de activos financieros.
- Predicción de tasas de desempleo en México mediante *big data* (https://www.jstor.org/stable/26863997), entre muchos otros. Trabajo que utiliza datos de *Google Trends* para pronosticar la tasa de desempleo en México aplicando algoritmos de aprendizaje automático.

Por otro lado, existen varios proyectos y publicaciones que utilizan modelos de predicción para explorar las posibilidades de ingresar a alguna universidad; sin embargo, la gran mayoría de estos proyectos se enfocan en analizar los datos correspondientes a universidades de los Estados Unidos de América. Algunos ejemplos de estos proyectos son los siguientes:

- https://github.com/amunde/university-admission. Proyecto en Python que analiza datos de universidades mediante algoritmos de regresión lineal.
- https://github.com/Manandedhia/Profile-Analysis. Repositorio que utiliza el lenguaje de programación R para analizar datos de universidades en EUA para encontrar patrones de admisión de estudiantes que desean ingresar a estudios de maestría.
- https://github.com/FilipTrocin/Acceptance_Predictor. Repositorio que utiliza algoritmos de regresión lineal para predecir la probabilidad de admisión a un curso de ingeniería en una universidad de EUA.

Por todo lo anterior es que se considera de interés la aplicación de este enfoque para la utilización de datos estadísticos en el contexto de los estudios universitarios en la ciudad de Guadalajara para determinar las probabilidades de ingreso de los aspirantes a la UdeG, y así, dar pie a nuevos estudios en esta misma área aplicados a otras universidades de la región o a otras localidades de México.

Para la realización de este trabajo se utilizará el lenguaje de programación Python, un algoritmo de aprendizaje automático conocido como árbol de decisión, librerías y herramientas de visualización de datos, así como datos estadísticos que abarcan un periodo de 10 años, publicados por la UdeG en su página de estadísticas. Para la interfaz gráfica se usa un *web framework* llamado Flask (https://flask.palletsprojects.com/en/2.1.x/).

El código completo puede consultarse en el siguiente repositorio de GitHub: https://github.com/doble-eme/modelo_ingreso_udg y la interfaz gráfica de la aplicación funcionando se puede consultar en la siguiente liga: https://predict-udg.herokuapp.com.

1.2. Justificación

En la mayoría de los casos de proyectos y publicaciones existentes, los datos y el enfoque de los estudios se refieren a universidades de Estados Unidos de América (EUA), o por lo menos, de habla inglesa. Uno de los objetivos de este trabajo es que la literatura y los estudios en torno al contexto latinoamericano se amplíe y sirva como precedente para que haya más estudios en este sentido.

Por otro lado, los estudiantes muchas veces no tienen muy claro, incluso ya muy cerca de graduarse de la preparatoria, qué es lo que quieren estudiar y mucho menos los factores que pueden influir en la posibilidad de ingresar a la universidad o a la carrera que desean estudiar. Este trabajo pretende ayudar también a que se genere una perspectiva más amplia de qué factores entran en juego al intentar ingresar a la universidad.

1.3. Problema

A pesar de la existencia de datos públicos que posibilitan estudios en el ámbito de la inteligencia artificial o aprendizaje automático en torno a la educación en México, estos no están siendo aprovechados más que por algunos medios y centros educativos, pero solo

como fuentes de información, y no tanto como objeto de estudio para generar más conocimientos.

Por otro lado, a pesar de las grandes cantidades de alumnos que hacen trámites para ingresar a alguna universidad en la región, no existen modelos todavía para determinar las posibilidades que tienen los mismos de ingresar a alguna de las carreras que se ofrecen por la UdeG.

Esto puede deberse en parte a que la utilización de estos modelos es relativamente reciente ya que hace apenas algunos pocos años que las computadoras de uso personal vienen equipadas con especificaciones capaces de procesar y analizar grandes cantidades de datos, requisito para que funcionen estos modelos de manera eficiente.

Además, anteriormente era muy complicado acceder a grandes cantidades de datos, almacenarlos y procesarlos. Sin embargo, el Internet ha permitido a casi cualquiera acceder a información desde sus casas o dispositivos móviles, y las velocidades de descarga ahora son mucho mayores que antes. Los equipos de cómputo ofrecen mayores cantidades de almacenamiento y mayor capacidad de procesamiento de datos.

Sin embargo, hoy por hoy es posible ya realizar este tipo de análisis de grandes cantidades de datos, y los modelos de inteligencia artificial y aprendizaje automático están siendo utilizados en casi todos los ámbitos de la industria y la educación, desde finanzas hasta moda y alimentos.

Como se ha mencionado antes, existen ya trabajos que utilizan estos modelos para estudiar otros temas en el ámbito mexicano, y por su parte, existen como tal trabajos que analizan este tema en específico pero enfocándose en universidades de EUA, por lo que no existe como tal una aplicación de estos modelos al ámbito local en este tema en específico.

Una de las dificultades más grandes para este tipo de análisis es la captura o la obtención de los datos. Sin embargo, como ya se mencionó, la misma UdeG cuenta con un medio para la descarga de grandes cantidades de datos que pueden ser utilizados para este tipo de análisis y sin embargo, hasta ahora solo están siendo utilizados de manera informativa o para dar a conocer el estado de cosas a la fecha.

Se considera que contar con modelos de predicción en torno a las posibilidades de ingreso a la universidad puede no solo ayudar a los estudiantes a tomar una decisión más informada sobre qué carrera elegir, sino que puede ayudar a ampliar los estudios que se generen en torno a este tema en otras universidades o en otras regiones del país.

Al día de hoy existen las condiciones tecnológicas y se cuenta con disponibilidad de acceso a algunas bases de datos para realizar un modelo de predicción sobre las posibilidades de ingreso a la UdeG; sin embargo, estos datos todavía no están siendo utilizados para este fin y se usan más bien para informar del estado de cosas actual. Los alumnos interesados en ingresar a alguna carrera universitaria podrían verse beneficiados de un modelo de predicción que les ayude a tomar mejores decisiones sobre qué carrera estudiar.

Es importante mencionar también que aunque existen algunas bases de datos disponibles, también hay grandes áreas de mejora en este sentido, y tanto las universidades como las instituciones de gobierno podrían participar más activamente en la liberación de datos estadísticos para generar más oportunidades de análisis e innovación.

1.4. Objetivos

A continuación se enlistan el objetivo general y los objetivos particulares de este trabajo.

Objetivo general

Analizar los datos publicados por la UdeG sobre las estadísticas de ingreso para generar un sistema de predicción que encuentre las probabilidades que un determinado estudiante tiene de ingresar a alguna carrera universitaria.

Objetivos particulares

Los siguientes son los objetivos particulares de este trabajo:

- Analizar y procesar los datos estadísticos de la UdeG para aplicar un modelo de aprendizaje automático adecuado para obtener predicciones de probabilidades de ingreso a la universidad.
- Aplicar los resultados del análisis y procesamiento de los datos en una interfaz gráfica que pueda ser utilizada para ver los resultados del análisis.

- Evaluar la eficacia del modelo para determinar si puede ser replicado en otros casos de estudio, tales como otras universidades de la región o de otras localidades del país.

1.5. Novedad científica, tecnológica o aportación

Como se mencionó antes, existen ya estudios en este sentido que utilizan modelos similares y las mismas tecnologías (Python o R, algoritmos de aprendizaje automático, etc.). Sin embargo, como se hizo mención también anteriormente, los estudios existentes se centran en universidades de los Estados Unidos de América y utilizan datos que corresponden a estadísticas de esa región. En este caso, el estudio utiliza datos reales de la región de Jalisco, y se centra en los parámetros disponibles de las estadísticas publicadas por la UdeG. Esto con la finalidad de aportar mayores estudios de esta naturaleza al entorno local y que eventualmente se amplíen y se generen más trabajos y análisis con datos de relevancia para nuestro país y nuestra región.

2. ESTADO DEL ARTE O DE LA TÉCNICA

Resumen: *En esta sección se hace un repaso de lo que significa la ciencia de datos, con qué otras áreas del conocimiento se relaciona, quiénes la realizan y en qué tipo de proyectos se aplica, así como el tipo de estudios que existen con respecto a los modelos de predicción basados en aprendizaje automático.*

2.1. Introducción a la ciencia de datos

El término ciencia de datos ha crecido en popularidad en los últimos años. Sin embargo, muchas veces es mal utilizado, o simplemente se utiliza sin saber realmente lo que significa o hasta dónde llega su alcance. Lo más común es que se le relacione con las grandes compañías tecnológicas y con otro término muy popular, que es el de *big data*.

A continuación, se revisan algunos de los antecedentes de la ciencia de datos, de dónde surge, con qué otras áreas del conocimiento se relaciona, y cuál es su alcance y las metodologías que se utilizan para aplicar la ciencia de datos a problemas reales. De este modo se puede delimitar y comprender mejor el alcance del presente trabajo.

2.2. Ciencia de datos y áreas del conocimiento con las que se relaciona

A pesar de que, por ser una disciplina con un origen relativamente reciente, no haya mucho consenso sobre qué es exactamente la ciencia de datos, lo que sí es claro para varios autores es que ésta debe su origen a otras disciplinas ya existentes, tales como la estadística, las matemáticas, la computación y la informática, entre otras.

Así define la ciencia de datos Longbing Cao (Cao, 2017) en el artículo "Data Science: Challenges and Directions": "La ciencia de datos es un campo interdisciplinario que se basa en y sintetiza otros campos del conocimiento, incluyendo la estadística, la informática, la computación, comunicaciones, administración y sociología, para estudiar datos siguiendo un «pensamiento de ciencia de datos»". La **Figura 1** muestra el diagrama que proporciona el autor para ilustrar cómo la ciencia de datos se interrelaciona con otras disciplinas:

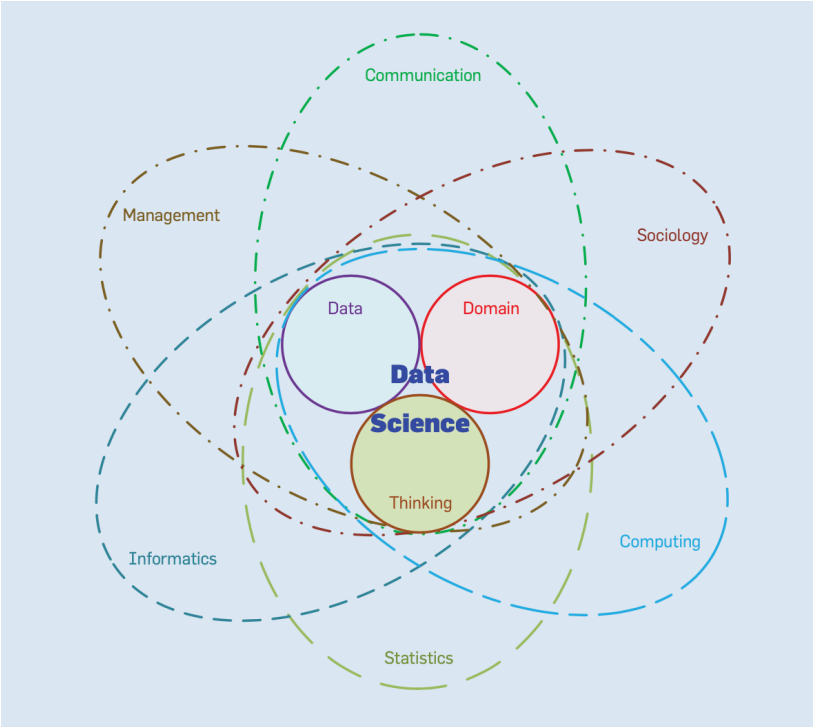

Figura 1. *La ciencia de datos en relación con otras áreas del conocimiento. Tomada de "Data Science: Challenges and Directions" por L. Cao, 2017, Communications of the ACM, vol. 60, no. 8, (p. 59). Derechos de autor 2017 por Longbing Cao.*

Si bien muchos pueden considerar a la ciencia de datos como una nueva forma de llamarle a la estadística o al análisis de datos, como podemos ver en la **Figura 1**, más que una nueva forma de llamarle a otras disciplinas, la ciencia de datos nace y se nutre de otras disciplinas para llegar a donde éstas no llegarían de forma independiente.

Así explica Field Cady (Cady, 2017) la ciencia de datos en su libro *The Data Science Handbook*: "La ciencia de datos significa hacer trabajo de análisis de datos que, por una o por otra razón, requiere una cantidad sustancial de habilidades de ingeniería de software". Y agrega: "Algunas veces, el entregable final es el tipo de trabajo que un estadista o analista de negocios entregaría, pero lograr su finalización demanda habilidades de software que el típico analista de datos simplemente no posee." (Cady, 2017).

Esto se debe a que en la actualidad los avances en los dispositivos tecnológicos que se utilizan día a día producen y a la vez son capaces de almacenar y procesar grandes cantidades de datos (*big data*) que antes simplemente no era posible. Hoy por hoy, cada que navegamos en la web, hacemos clic en algún enlace o subimos contenido a nuestras redes sociales, o utilizamos los servicios digitales de mapas para obtener direcciones y

recomendaciones de lugares, restaurantes o cualquier establecimiento, generamos grandes cantidades de información que son almacenadas y procesadas para su posterior análisis, mismo que se traduce en ganancias para las compañías que analizan estos datos y comercializan anuncios y recomendaciones pagados por terceros. Así lo explica Joel Grus (Grus, 2019) en *Data Science from Scratch*: "Vivimos en un mundo que está ahogándose en datos. Los sitios web rastrean cada clic de cada usuario. Tu *smartphone* lleva un registro de tu ubicación y velocidad cada segundo del día. Los *quantified selfers* (yo cuantificados) utilizan podómetros que están constantemente registrando sus latidos, hábitos de movilidad, dieta y patrones del sueño, los hogares inteligentes recolectan hábitos de vivienda y los comercializadores inteligentes recolectan hábitos de consumo". Y continúa explicando que, en general, el Internet es un generador gigantesco de datos, pues almacena datos en diversas formas y sobre prácticamente cualquier tema, desde las cosas más triviales hasta cuestiones relacionadas con gobiernos o instituciones.

En resumen, la ciencia de datos es una disciplina que surge de otras, principalmente de la estadística y las matemáticas, pero que se ayuda de la programación y herramientas informáticas para recolectar, almacenar, procesar y analizar información para generar nueva información útil para individuos o negocios.

2.3. Alcance y metodologías

Como se mencionó anteriormente, uno de los objetivos de la ciencia de datos es hacer que los datos sean transformados en **información útil**. Trabajar con algoritmos que ayuden a procesar información y a analizar grandes cantidades de datos puede no ser de mucha ayuda si esos resultados no son presentados de manera clara y organizada de forma que, personas con poco contexto de las matemáticas, informática o computación, puedan entenderlos.

Los dirigentes o ejecutivos de las empresas no quieren ver los datos por sí solos, sino que necesitan ver cómo esos datos afectan el negocio, o cómo se están comportando los productos en los que han invertido. De igual manera, los usuarios finales de productos en los que entran en juego algoritmos de ciencia de datos no sacarán mucho beneficio de

ver una serie de datos sino que necesitan ver esos datos en contexto y presentados de manera clara y bien organizada.

La misma disciplina de la ciencia de datos tiene aplicaciones para hacer que los datos que se analizan puedan ser visualizados y transformados en información útil. Es por eso que la primera aplicación que se menciona en este documento es la de la **visualización de datos**.

Primero, se hará una distinción entre los dos roles que juega la visualización de datos en la ciencia: (1) "la visualización como actividad utilizada durante el proceso de investigación para obtener una perspectiva de los datos aplicando el «análisis explicativo de los datos»" (Van Geenen & Wieringa, 2020) y (2) "la visualización de los datos como imágenes utilizadas como representaciones y resultado de la investigación, lo cual es comunicado públicamente" (Van Geenen & Wieringa, 2020).

En el primer caso la aplicación de la visualización de datos se refiere a su utilización por investigadores para analizar y manipular los datos de forma gráfica durante el proceso de investigación. En el segundo caso, se trata de herramientas que ayudan a los investigadores a presentar los resultados de sus investigaciones o análisis de una manera sencilla, llamativa y fácil de entender para diversas audiencias.

Como ejemplo, la **Figura 2** presenta los resultados de un análisis de comportamiento de los usuarios de la red social Twitter que utilizan la función de *reply* (respuesta) en el idioma neerlandés:

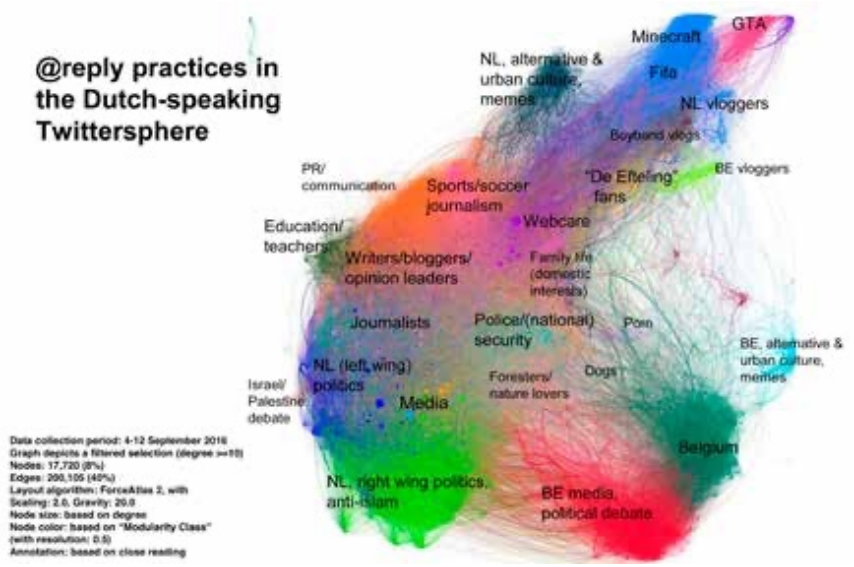

Figura 2. *Gráfica creada por Daniela van Geenen y Maranke Wieringa con Gephi y Photoshop para representar cómo distintos profesionales de diversas áreas utilizan la red social Twitter. Tomada de Van Geenen, D., & Wieringa, M. (2020). Approaching data visualizations as interfaces: An empirical demonstration of how data are imag(in)ed. En Engebretsen M. & Kennedy H. (Eds.), Data Visualization in Society (pp. 141-156). Amsterdam: Amsterdam University Press. doi:10.2307/j.ctvzgb8c7.15*

La **Figura 3** muestra otro ejemplo de visualización de datos, ésta correspondiente al campo del periodismo:

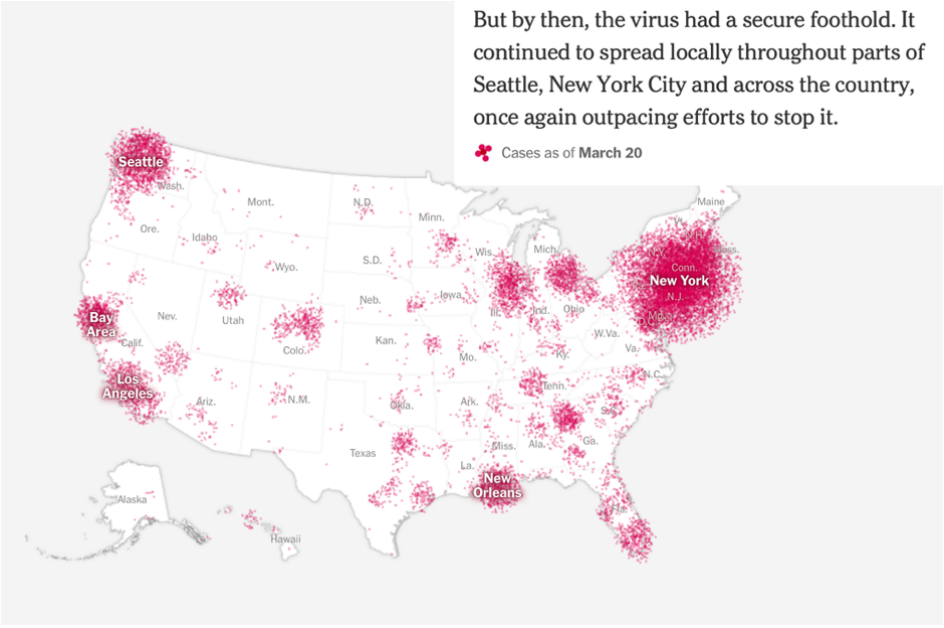

Figura 3. *Mapa interactivo en un reportaje del diario New York Times acerca del avance del virus Covid-19. Tomado de Wu, J., Cai, W., Watkins, D., & Glanz, J. (2021, January 12). How the Virus Got Out. The New York Times. Disponible en: https://www.nytimes.com/interactive/2020/03/22/world/coronavirus-spread.html. Consultado en 15/Mar/2021*

Para realizar este gráfico interactivo, el diario tomó datos tales como los patrones de viaje de las personas con información proveniente de herramientas que rastrean el movimiento de las personas a través del uso de teléfonos inteligentes, modelado de datos para estimar los posibles escenarios del brote del virus, y estimaciones según el número de personas que habían salido de Wuhan durante la etapa inicial del brote y el comportamiento del virus en esa región.

En este ejemplo se observa que la representación de los datos no se limita a presentar información de forma gráfica, sino que lo hace en un contexto y en conjunto con otros datos de manera que la información no solo es clara y precisa, sino que además proporciona una perspectiva de cómo un fenómeno evoluciona en el tiempo y cómo puede afectar a un entorno particular. Explica Wibke Weber en el artículo "Exploring narrativity in data visualization in journalism": "Un gráfico circular, un diagrama de red o un diagrama de árbol no representan un historia por sí solos, presentan hechos. Por otra parte, un gráfico lineal que muestra cómo ciertos valores han cambiado a lo largo de un periodo de tiempo, puede contar una historia." (Weber, 2020).

Con esto puede verse que **el alcance** de la ciencia de datos prácticamente llega a cualquier ámbito que se quiera aplicar, mientras existan datos suficientes para analizar y procesar, lo cual en la actualidad es fácilmente hecho posible por las interacciones en redes sociales, la localización registrada por los teléfonos celulares o los dispositivos inteligentes en casa; y mientras esos datos sean analizados y presentados de una forma útil y visualmente fácil de entender a los usuarios finales, usando una narrativa lógica.

Para lograr esto existen diferentes **metodologías** y herramientas que intervienen en los procesos de análisis y visualización de los datos. Como ya se mencionó antes, uno de los pilares de la ciencia de datos es el conocimiento de lenguajes de programación. En la actualidad, los lenguajes de programación más utilizados para trabajos de ciencia de datos son **Python** y **R**. Nos dice Field Cady (Cady, 2017) en su libro: "Python es el lenguaje más popular entre los científicos de datos. R es su único mayor competidor, por lo menos en cuanto a herramientas gratuitas. He utilizado ambos y pienso que Python es mucho mejor".

Python ofrece además numerosas librerías que ya contienen algoritmos enfocados específicamente a la ciencia de datos, además de librerías específicas para la visualización de datos, con lo cual se cumplen los requisitos mencionados anteriormente (análisis, procesamiento y visualización de datos). En cuanto a almacenamiento de datos, las bases de datos más comunes son las basadas en SQL, pero existen también herramientas para el almacenamiento de datos en bases de datos no relacionales (NoSQL) tales como MongoDB (https://www.mongodb.com/). (Cady, 2017)

Finalmente, se hablará de otra de las disciplinas que complementan a la ciencia de datos: los modelos de aprendizaje automático. Estos modelos son algoritmos que ayudan a los investigadores a realizar lo que de manera tradicional implicaría cantidades enormes de trabajo de procesamiento y análisis de datos. Estos modelos pueden ser implementados en lenguajes de programación como Python mediante librerías de libre acceso.

Giuseppe Bonaccorso (Bonaccorso, 2018) menciona algunas de las aplicaciones de los modelos de aprendizaje automático en su libro *Machine Learning Algorithms*: “Filtros de *spam*, Procesamiento de Lenguaje Natural (NLP por sus siglas en inglés), rastreo visual mediante cámaras web o *smartphones* o el análisis predictivo, son sólo algunas de las aplicaciones que revolucionaron las interacciones humano-máquinas e incrementaron nuestras expectativas”.

En las secciones posteriores se analizan con más detalle los algoritmos específicos que se utilizan en aprendizaje automático, por ahora solo se puede mencionar que éstos se dividen en modelos supervisados, modelos no supervisados, semi-supervisados, aprendizaje por refuerzo y neurociencia computacional (Bonaccorso, 2018).

A continuación se analizan algunas aplicaciones de modelos de aprendizaje automático aplicados a casos de predicción de resultados para tener mayor contexto de lo que se ha hecho hasta ahora en este caso específico de la ciencia de datos, que es la que atañe a este trabajo.

2.4. ¿Qué se ha hecho y cuáles han sido los resultados en cuanto a modelos predictivos?

Como se ha visto hasta ahora, en el campo de la ciencia de datos existen muy diversas aplicaciones, las cuales van desde el periodismo hasta el *marketing* y prácticamente cualquier campo que disponga de suficientes datos para analizar mediante modelos de aprendizaje automático o alguna otra metodología según sea conveniente para el caso específico.

El enfoque de este trabajo es revisar algunas de las aplicaciones más recientes que se han hecho en el campo de la ciencia de datos, específicamente en el ámbito de los modelos de predicción, pues son los que interesan para este trabajo para analizar qué se ha hecho y dónde se encuentra el conocimiento en esta área en específico, además para ponderar qué queda por hacer para completar el presente trabajo.

Se presenta un primer ejemplo de análisis predictivo en el ámbito educativo: el trabajo de Wenjie Xing y Ming Guan (Xing & Guan, 2017) en el que analizan las posibles desigualdades raciales contra asiáticos en las admisiones a universidades de EUA utilizando un modelo de regresión. Los autores afirman que los asiático-americanos tienen menores probabilidades de ingresar a las universidades de élite estadounidenses en comparación con otras minorías. Sin embargo, mencionan que una gran limitante para el estudio que realizan es que los datos utilizados corresponden solo a los de universidades estatales, puesto que las universidades privadas no están obligadas a hacer públicos sus registros de datos sobre admisiones. Por otra parte, mencionan también como una limitante que el modelo utilizado, de regresión lineal, no proporciona un nivel de confianza tan elevado al evaluar solamente una variable y deja fuera otras que sí son tomadas en cuenta por los comités de admisión, como las actividades extracurriculares o la carta de motivos personales de los estudiantes.

Otro ejemplo de la aplicación de modelos de aprendizaje automático en el ámbito educativo es un trabajo por Joseph Jamison (Jamison, 2017) en el que analiza y predice las tasas de admisión de la universidad privada Davidson College mediante un algoritmo de clasificación de “bosques aleatorios” (*random forest classifier*). Como parte de ese trabajo creó una interfaz gráfica para que la oficina de admisiones de la universidad pudiera utilizar

el modelo fácilmente para centrar sus recursos en los estudiantes que más probablemente estén dispuestos a aceptar una oferta de la universidad, evitando así malgastar recursos en estudiantes que probablemente no vayan a aceptar la oferta de todos modos.

Así como estos ejemplos, la gran mayoría de los estudios que se han realizado corresponden a datos de universidades de los EUA y se han realizado con datos correspondientes al contexto de ese país. Existen también varios estudios que centran su atención en otras regiones, pero son los menos. En este trabajo se busca aportar al análisis de los datos locales y propiciar que se continúen trabajos de esta naturaleza en otras regiones de México. En la siguiente sección se explorarán los algoritmos de aprendizaje automático más a detalle y se determinará cuál será la metodología a seguir para este trabajo en particular.

3. MARCO TEÓRICO/CONCEPTUAL

Resumen: *En esta sección se explica con más detalle en qué consiste el proceso de análisis de datos y cuáles son algunas de las técnicas y algoritmos específicos que se utilizan en el campo del aprendizaje automático.*

3.1. El proceso de análisis de datos

Para describir el proceso de análisis de datos primero se puede explicar el proceso en su forma más simple y luego se detallará una serie de pasos más específicos que conforman el proceso en sus aplicaciones para los modelos de aprendizaje automático.

Brett Lantz (Lantz, 2013) explica en su libro *Machine Learning with R* el proceso básico de aprendizaje, sin importar si se trata de un humano o una máquina: "El proceso se puede dividir en tres componentes: (1) **Entrada de datos**, (2) **Abstracción** y (3) **Generalización**."

El autor también menciona que estos elementos no son absolutos y que además están tan interrelacionados que es difícil separarlos por completo. Por otra parte, en cada uno de esos elementos intervienen también otros elementos que los complementan.

Por ejemplo, en **la entrada de datos** está involucrada la observación, el almacenamiento de los datos, y la capacidad de recordar los datos almacenados para razonamientos futuros. En el caso de la **abstracción** interviene el proceso de procesar esos datos en representaciones más amplias (darle significado a los datos); y en la **generalización**, se utilizan los datos abstraídos para crear las bases para una toma de acción posterior.

Más adelante, el mismo autor refiere que en el proceso de abstracción interviene un proceso conocido como **representación del conocimiento**, que se refiere a la capacidad de darle un significado a la información o a alguna representación de esta información. En el caso de las computadoras, durante este proceso es que se condensan los datos todavía sin procesar dentro de un modelo para procesar los datos. El autor señala que la selección del modelo es algo que no se deja a decisión de la computadora, sino que este es un paso que debe ser realizado por el investigador según su criterio y su conocimiento de los datos. El proceso de seleccionar un modelo específico y aplicarlo a un conjunto de datos se conoce como **entrenamiento**. Finalmente, en el proceso de **generalización**, el objetivo es encontrar los elementos más significativos que forman parte del proceso para utilizarlos para acciones futuras.

El procedimiento descrito anteriormente detalla las partes del proceso muy a grandes rasgos. A continuación se presentan dos diferentes propuestas de pasos específicos para el análisis de datos.

Estos son los pasos propuestos por Jared Dean (Dean, 2014) en el libro *Big Data, Data Mining, and Machine Learning*:

1. Preparación de los datos
2. Exploración y análisis de los datos
3. Construcción de un primer modelo
4. Aplicación y comparación de distintos modelos iterativamente (selección del mejor)

Estos son los pasos propuestos por Brett Lantz (Lantz, 2013) para aplicar el aprendizaje automático a un conjunto de datos:

1. Recolección de los datos
2. Exploración y preparación de los datos
3. Entrenamiento de modelo(s)
4. Evaluación del desempeño de modelo(s)
5. Mejora del desempeño de modelo(s), si es necesario

Se puede ver que ambos autores coinciden en la mayoría de los pasos, con la diferencia de que Lantz incluye el paso de recolección de datos como el primer paso, mezcla en el segundo paso el proceso de exploración y preparación de los datos, y agrega un paso al final para mejorar el desempeño del modelo en caso de ser necesario.

3.2. Técnicas y tipos de algoritmos de aprendizaje automático

A continuación se describen algunas de las técnicas y algoritmos utilizados en aprendizaje automático. Existen principalmente dos tipos de de modelos de aprendizaje automático:

1. **Predictivos**, que pretenden estimar valores futuros de variables de interés (variable dependiente) usando otras variables (variables independientes o predictivas).
2. **Descriptivos**, que pretenden identificar patrones que expliquen o resuman los datos.

Brett Lantz (Lantz, 2013) explica que "como a los modelos predictivos generalmente se les dan instrucciones de lo que necesitan aprender y cómo deben

aprenderlo, al proceso de entrenar un modelo predictivo se le conoce como modelo de **aprendizaje supervisado**".

Por su parte, el mismo autor explica más adelante que "Un modelo descriptivo, al contrario que uno predictivo, no posee un elemento que sea de mayor interés que el resto. De hecho, como no hay un objetivo de aprendizaje, el proceso de entrenamiento de un modelo descriptivo se conoce como modelo de **aprendizaje no supervisado**".

Estos son algunos de los algoritmos de aprendizaje automático más comunes, según presentados en el libro de Kevin Jolly (Jolly, 2018), *Machine Learning with Scikit-learn Quick Start Guide*:

Modelos de aprendizaje supervisado

Modelos que pueden ser utilizados tanto para resolver problemas de clasificación como de regresión.

- **Regresión lineal**
- **Regresión logística**
- ***k-Nearest Neighbors* o k-vecinos más próximos**
- ***Support vector machines* o máquinas de vectores de soporte**
- ***Tree-based algorithms* o algoritmos de árbol (árboles de decision, random forests, boosted trees)**
- ***Naive Bayes* o clasificador bayesiano ingenuo**

Modelos de aprendizaje no supervisado

Modelos usados típicamente para agrupar puntos de datos según su distancia.

- **k-means**

Como este trabajo se enfoca en hacer un análisis predictivo (posibilidad de ingreso a la UdeG), los modelos con los que se trabajará serán entonces modelos de **aprendizaje supervisado**.

3.3. Selección de modelos

Una de las partes fundamentales en el proceso es la selección del "mejor" modelo o modelos para el conjunto de datos a analizar. Dependiendo de la naturaleza de los datos y el

objetivo de la investigación, en algunos casos será más fácil o más complicado escoger el modelo o modelos que mejor funcionen, pero una de las recomendaciones mencionadas por Lantz (Lantz, 2013) es "considerar las distinciones entre los distintos algoritmos (…) y, cuando la precisión predictiva sea un aspecto importante, puede que sea necesario probar con varios algoritmos y escoger el que mejor se desempeñe".

En este sentido Jared Dean (Dean, 2014) dice: "Una consideración en el proceso de construcción de modelos de predicción es determinar cuál modelo es el mejor. (…) El gran número de opciones y combinaciones hacen que un método de «probarlo todo» sea inviable para cualquier problema de minería de datos. Así que el proceso de evaluación de los modelos juega un rol importante en el proceso". Pero advierte que la dificultad está justamente en determinar cuál es el "mejor" modelo, puesto que para determinar eso hay que tomar muchos factores en consideración.

También es importante considerar la naturaleza de los datos con los que se estará trabajando. En este caso, al tratarse de un conjunto de datos bastante amplio con diversas opciones a evaluar (carreras ofertadas por la UdeG), del cual se pretende obtener una decisión final (posibilidad de entrar o no a determinada carrera), hemos decidido utilizar el **algoritmo de árbol de decisión**.

3.4. Árboles de decisión

Los árboles de decisión son un tipo de algoritmo de árbol de clasificación. Estos algoritmos son "utilizados para predecir una categoría o una clase" (Jolly, 2018). Un árbol de decisión "en términos simples, es un conjunto de reglas para clasificar observaciones en distintos grupos" (Jolly, 2018).

Así pues, el árbol de decisión clasifica elementos en distintos grupos tantas veces como sea necesario, formando una estructura similar a la de un árbol con distintas ramificaciones. Al final, visualmente puede tomar la forma de un árbol (invertido), como muestra la **Figura 4**:

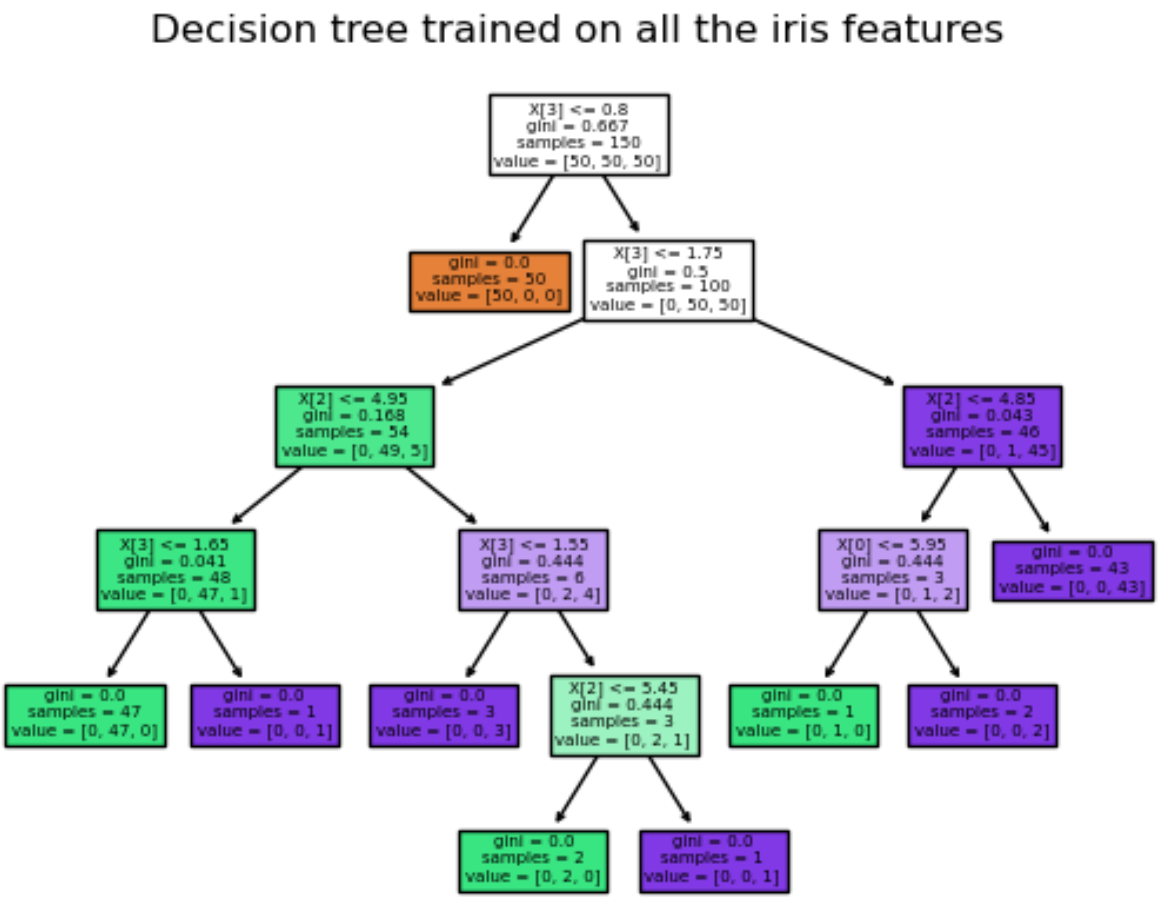

Figura 4. *Ejemplo de un árbol de decisión tomado de la documentación de la librería para Python de aprendizaje automático, Scikit-learn: https://scikit-learn.org/stable/auto_examples/tree/plot_iris_dtc.html (1.10. Decision Trees, 2007)*

El nodo principal es llamado **nodo raíz** o ***root*** y es "el atributo más importante del árbol en cuanto a la decisión de cómo se agruparán las observaciones" (Jolly, 2018). Los nodos debajo del nodo raíz son los **nodos hijos** (***children***) y los nodos de la última línea o los de la parte más baja del árbol son conocidos como **nodos hoja** o ***leaf nodes***.

Uno de los aspectos más importantes es determinar qué característica es la que mejor puede funcionar como nodo raíz o la que "ofrece la mejor división posible" (Jolly, 2018), a partir de la cual se hará la primera gran clasificación. A partir de esa primera clasificación el modelo irá clasificando el resto de las características "en distintos grupos, según la cantidad de clases o categorías que existan en el conjunto de datos" (Jolly, 2018) hasta que "(1) ya no existen datos para categorizar, (2) continuar subdividiendo los datos ya

no mejorará el estado actual del modelo, o (3) el árbol ha alcanzado el número máximo de divisiones especificadas" (Dean, 2014).

Una vez que se ha completado el proceso, el árbol resultante casi siempre tiene una sobresaturación o sobreajuste, por lo que es necesario realizar un proceso de "podado" o *pruning*. Generalmente este proceso se realiza utilizando "un conjunto de datos distinto al utilizado para entrenar el modelo para validar el comportamiento del modelo con datos generales, distintos a los usados para crear las categorías de separación" (Dean, 2014).

Existen varios métodos para determinar cómo se realizará la división de las categorías o qué tan acertadas serán esas divisiones; entre ellas: la **entropía** y el índice **Gini**. "La entropía utiliza la información de cada separación: el indicador de que la separación es 'buena' es la ganancia de información (***information gain***) o disminución de la entropía" (Dean, 2014). El índice Gini funciona de manera similar, tomando como referencia la ganancia de información con cada separación.

La fórmula matemática que representa la medición de la entropía es la siguiente:

$$E = \sum_{r}^{R} \frac{N_r}{N_R} \sum_{t}^{T} \frac{N_t}{N_r} log_2 \left(\frac{N_t}{N_r} \right)$$

Figura 5. *Fórmula matemática que representa la medición de la entropía.*

Donde se calcula la entropía para un conjunto de N_t observaciones que poseen un valor *t* como *target* u objetivo.

De manera similar, la fórmula para el índice Gini se representa de la siguiente forma:

$$G = \sum_{r}^{R} \frac{N_r}{N_R} \sum_{t}^{T} \frac{N_t}{N_r} \left(1 - \frac{N_t}{N_r} \right)$$

Figura 6. *Fórmula matemática que representa el índice Gini.*

(Dean, 2014)

En las siguientes secciones se verá a detalle la implementación del modelo al conjunto de datos correspondiente a nuestro análisis, desde la obtención de los datos, pasando por el entrenamiento del modelo, hasta la obtención de los resultados relevantes.

4. DESARROLLO METODOLÓGICO

Resumen: *En esta sección se explica el proceso llevado a cabo para la implementación del modelo de árbol de decisión para determinar las posibilidades de ingreso a distintas carreras de la UdeG.*

4.1. Esquema de trabajo

El proceso incluye las siguientes fases:

I. Obtención de los datos desde la página web de Estadísticas de la UdeG
II. Preparación de los datos
    i. Análisis inicial de los datos y su contenido
    ii. Descripción de los datos
    iii. Limpieza de los datos
III. Análisis de los datos
    i. Análisis de estadística descriptiva
    ii. Establecer variables de interés
    iii. Análisis de los datos con respecto a las variables de interés
IV. Entrenamiento de algoritmos y análisis de los resultados
    i. Entrenamiento del modelo
    ii. Comprobar el funcionamiento del modelo
V. Desarrollo de interfaz gráfica

4.2. Obtención de los datos

En el proceso de obtención de datos se consultaron varias fuentes públicas de información acerca de la educación en México, tales como el Instituto Nacional de Estadística y Geografía (INEGI) o la Secretaría de Educación Pública (SEP), sin embargo, los datos disponibles en estas fuentes no correspondían a los objetivos de la investigación, no estaban lo suficientemente actualizados o eran muy generales.

Conforme el objetivo de la investigación se fue definiendo más y el alcance de la investigación se hizo más preciso, encontramos que la información más adecuada era la que está disponible en la página de **Estadísticas** de la UdeG: http://escolar.udg.mx/estadisticas.

4.3. Preparación de los datos

En esta sección se describe primero el contenido de los datos y luego el proceso llevado a cabo para la limpieza y preparación de los datos para poder utilizarlos en el modelo de aprendizaje automático. El proceso consistió básicamente en renombrar los archivos originales, renombrar los títulos de las columnas que conforman los archivos y crear *data frames* utilizando la librería Pandas para crear un conjunto de datos relevante y que sirva para entrenar el modelo.

Un *data frame* es "una tabla bi-dimensional, de tamaño variable, con datos potencialmente heterogéneos" (*Pandas.DataFrame - pandas 1.4.2 Documentation*, 2008).

4.3.1. Análisis inicial de los datos y su contenido

Los datos disponibles en esta fuente contienen información acerca de los estudiantes admitidos en los últimos 10 años: promedios mínimos por carrera y por centro universitario, número total de admitidos (hombres y mujeres) y número de admitidos por escuela de procedencia (públicas y privadas).

Esta información se encuentra en archivos Excel (.xlsx). El proceso de descarga consistió en descargar cada archivo de Excel individualmente, con lo que se obtuvo un total de 40 archivos `.xlsx`.

Estos 40 archivos de Excel pertenecen a dos categorías según el tipo de información que contienen:

- `Admitidos_por_procedencia_escolar_{año}.xlsx`
- `Puntajes_mininos_CUs_{año}.xlsx`

En los archivos de **Admitidos por procedencia escolar** se encuentra la información del total de admitidos por cada centro universitario y por cada carrera, divididos en el número de mujeres y hombres que provienen de escuelas públicas o privadas, pertenecientes a la UdeG o incorporadas, etcétera. A continuación, en la **Figura 7** se presenta una imagen de uno de los archivos Excel:

			ADMISION DE MUJERES POR ESCUELA DE PROCEDENCIA										ADMISION DE HOMBRES POR ESCUELA DE PROCEDENCIA									
DESCRIPCION DE CAMPUS	CARRERA	Total admitidos primer ingreso	Extranjeros	Incorporadas UDG	Oficial UDG	Particulares Jalisco	Particulares Zona Metropolitana de Guadalajara	Particulares Otros Estados	Públicas Jalisco	Públicas Zona Metropolitana de Guadalajara	Públicas Otros Estados	Sin Referencia	Extranjeros	Incorporadas UDG	Oficial UDG	Particulares Jalisco	Particulares Zona Metropolitana de Guadalajara	Particulares Otros Estados	Públicas Jalisco	Públic Zona Metro de Guad
CUCEI-C. U.	LICENCIATURA EN INGENIERIA TOPOGRAFICA	45	0	0	2	0	0	0	0	1	0	0	0	1	27	0	2	0	3	
CUCEI-C. U.	LICENCIATURA EN INGENIERIA EN COMPUTACION	170	0	2	19	0	0	0	0	3	0	0	1	9	105	0	7	4	3	
CUCEI-C. U.	LICENCIATURA EN INGENIERIA EN COMUNICACIONES Y ELECTRONICA	270	0	1	13	0	3	1	0	2	2	0	0	14	183	1	9	8	3	
CUCEI-C. U.	LICENCIATURA EN INGENIERIA BIOMEDICA	44	1	2	6	0	1	2	0	0	4	0	0	3	17	0	2	2	2	
CUCEI-C. U.	LICENCIATURA EN MATEMATICAS	60	1	0	18	0	0	1	2	2	4	0	0	2	19	0	5	0	3	
CUCEI-C. U.	LICENCIATURA EN QUIMICA	88	0	4	28	0	0	0	0	4	3	0	0	2	31	0	3	1	3	
CUCEI-C. U.	LICENCIATURA EN QUIMICO FARMACOBIOLOGO	150	0	6	62	0	3	4	3	7	8	0	0	3	42	0	1	5	4	
CUCEI-C. U.	LICENCIATURA EN INGENIERIA QUIMICA	164	0	8	42	0	4	2	1	6	4	0	0	9	63	0	3	3	5	
CUCEI-C. U.	LICENCIATURA EN INGENIERIA CIVIL	124	0	2	9	0	1	1	1	1	0	0	0	8	60	3	8	5	6	
CUCEI-C. U.	LICENCIATURA EN INGENIERIA INDUSTRIAL	170	0	1	25	0	8	1	0	3	2	0	0	5	89	0	9	3	5	
CUCEI-C. U.	LICENCIATURA EN INFORMATICA	170	0	5	25	0	3	0	1	9	2	0	0	9	94	0	5	0	2	
CUCEI-C. U.	LICENCIATURA EN INGENIERIA MECANICA ELECTRICA	185	0	0	3	0	0	0	1	2	1	0	0	6	125	0	8	4	10	
CUCEI-C. U.	LICENCIATURA EN FISICA	45	0	1	4	0	3	2	0	0	1	0	0	0	20	0	8	2	0	

***Figura 7**. Tabla con el número de mujeres y hombres admitidos según su procedencia escolar.*

Por su parte, en los archivos de **Puntajes mínimos por Centros Universitarios** se encuentra la información relativa al puntaje mínimo requerido en cada periodo para ingresar a cada carrera de cada centro universitario. Contiene otros datos tales como el porcentaje de admisión o el cupo disponible en cada periodo. A continuación, la **Figura 8** muestra una imagen con el contenido de uno de estos archivos:

PUNTAJES MINIMOS NIVEL SUPERIOR CAL. 2013"A"

CENTRO	CARRERA	ASPIRANTES	ADMITIDOS	NO ADMITIDOS	CUPO	CUPO DISPONIBLE	% ADMISIÓN	PUNTAJE MINIMO
CUAAD	LICENCIATURA EN URBANISTICA Y MEDIO AMBIENTE	75	50	25	50	0	66.67%	139.4222
	LICENCIATURA EN DISEÑO PARA LA COMUNICACION GRAFICA	569	135	434	135	0	23.73%	154.3200
	LICENCIATURA EN DISEÑO INDUSTRIAL	334	75	259	75	0	22.46%	160.5300
	LICENCIATURA EN ARQUITECTURA	743	135	608	135	0	18.17%	163.1700
	LICENCIATURA EN DISEÑO DE INTERIORES Y AMBIENTACION	376	60	316	60	0	15.96%	160.2589
	LICENCIATURA EN DISEÑO DE MODAS	82	40	42	40	0	48.78%	153.6778
	LICENCIATURA EN ARTES AUDIOVISUALES	28	15	13	15	0	53.57%	163.8889
	LICENCIATURA EN ARTES ESCENICAS PARA LA EXPRESION DANCISTICA	37	35	2	35	0	94.59%	106.9444
	LICENCIATURA EN ARTES ESCENICAS PARA LA EXPRESION TEATRAL	25	25	0	25	0	100.00%	123.7778
	LICENCIATURA EN ARTES VISUALES PARA LA EXPRESION FOTOGRAFICA	68	68	0	70	0	100.00%	118.6667
	LICENCIATURA EN ARTES VISUALES PARA LA EXPRESION PLASTICA	77	77	0	80	0	100.00%	109.5000
CUCBA	LICENCIATURA EN MEDICINA VETERINARIA Y ZOOTECNIA	583	185	398	185	0	31.73%	141.3044
	INGENIERO AGRONOMO	207	160	47	160	0	77.29%	115.7778
	LICENCIATURA EN BIOLOGIA	188	181	7	181	0	96.28%	110.8000
	LICENCIATURA EN CIENCIA DE LOS ALIMENTOS	61	36	25	36	0	59.02%	134.2211
	LICENCIATURA EN AGRONEGOCIOS	38	35	3	35	0	92.11%	107.2033
CUCEA	LICENCIATURA EN NEGOCIOS INTERNACIONALES	667	315	352	315	0	47.23%	142.3944
	LICENCIATURA EN ECONOMIA	99	99	0	100	0	100.00%	96.0556
	LICENCIATURA EN ADMINISTRACION	766	350	416	350	0	45.69%	140.8300
	LIC. EN ADMINISTRACION GUBERNAMENTAL Y POLITICAS PUBLICAS	128	80	48	80	0	62.50%	128.0000
	LICENCIATURA EN MERCADOTECNIA	492	290	202	290	0	58.94%	135.1478
	LICENCIATURA EN RECURSOS HUMANOS	334	100	234	100	0	29.94%	140.0522
	LICENCIATURA EN CONTADURIA PUBLICA	665	406	259	406	0	61.05%	134.8333
	LICENCIATURA EN ADMINISTRACION FINANCIERA Y SISTEMAS	274	120	154	120	0	43.80%	150.3056
	LICENCIATURA EN GESTION Y ECONOMIA AMBIENTAL	70	70	0	70	0	100.00%	104.0000
	LICENCIATURA EN TURISMO	522	285	237	285	0	54.60%	133.4189
	LICENCIATURA EN TECNOLOGIAS DE LA INFORMACION	147	86	61	86	0	58.50%	137.3889
CUCEI	LICENCIATURA EN QUIMICA	85	74	11	88	0	87.06%	126.2567
CUCEI	LICENCIATURA EN INGENIERIA EN COMPUTACION	409	196	213	196	0	47.92%	145.8889

***Figura 8**. Tabla con los puntajes mínimos requeridos para cada carrera de cada centro universitario en el período 2013 A.*

Para este trabajo se utilizaron solo los datos en los archivos de **puntajes mínimos**. Por lo que se procederá a describir los datos en estos archivos.

4.3.2. Descripción de los datos

Como se mencionó previamente, los archivos que se utilizarán en este trabajo son los correspondientes a puntajes mínimos que abarcan un periodo de 10 años que van del calendario escolar 2011A al 2019B. La **Tabla 1** muestra el contenido de los archivos `.xlsx`, que consisten, con algunas diferencias entre archivos, en los siguientes datos:

Variable	Tipo de dato	Descripción
Centro	String	El centro universitario en siglas (P. Ej: CUAAD, CUCSH, etc.)
Carrera	String	Nombre de la carrera
Aspirantes	Int	Número de aspirantes en ese calendario escolar
Admitidos	Int	Número de admitidos por carrera en el calendario escolar
No admitidos	Int	Número de aspirantes que no fueron admitidos a la carrera en ese calendario escolar
Cupo	Int	Número de espacios disponibles en total para esa carrera en ese calendario escolar, antes de admitir estudiantes
Cupo disponible	Int	Número de espacios restantes que quedan disponibles en esa carrera en ese calendario escolar luego de haber admitido a los estudiantes aspirantes
% Admisión	float	Porcentaje de admisión según el cupo y el número de aspirantes para cada carrera en el calendario escolar
Puntaje Mínimo	float	Puntaje mínimo requerido para ingresar a la carrera en ese calendario escolar. Se conforma del promedio de bachillerato más el puntaje obtenido en el examen de admisión.

Tabla 1. *Descripción y tipos de datos en los archivos originales.*

4.3.3. Limpieza de los datos

Una de las primeras cosas que se identifican en los archivos extraídos es que tenían diferencias en la nomenclatura de los archivos, así como diferencias en la nomenclatura de algunas columnas que identifican los datos. Para dar consistencia a los archivos y poderlos manipular más fácilmente, inicialmente se homologaron todos los archivos para que siguieran el siguiente formato: `Puntajes_mininos_CUs_{año}.xlsx`

Además, como se muestra en la **Tabla 2**, se renombraron las columnas para que tuvieran la siguiente nueva nomenclatura:

Nombre original	Nuevo nombre
CENTRO	CENTRO
CARRERA	CARRERA

ASPIRANTES	ASPIRANTES
ADMITIDOS	ADMITIDOS
NO ADMITIDOS	NO_ADMITIDOS
CUPO	CUPO
CUPO DISPONIBLE	CUPO_DISPONIBLE
% ADMISIÓN	PORCENTAJE_ADMISION
PUNTAJE MÍNIMO	PUNTAJE_MINIMO

***Tabla 2**. Nueva nomenclatura de las columnas.*

Una vez que se homogeneizaron las nomenclaturas de los archivos y las columnas de los datos, se procedió a importar los datos a un Jupyter Notebook (https://jupyter.org) para poder iniciar la limpieza de los datos antes de comenzar a entrenar el modelo.

4.4. Análisis de los datos

En esta sección se analizan los datos obtenidos para entender mejor la información con la que se cuenta antes de realizar el entrenamiento del modelo. Se hará un análisis de estadística descriptiva, se analizará cuáles son las variables de interés para el modelo y se explorará qué se puede comenzar a intuir a través de este análisis preliminar al entrenamiento y aplicación del modelo.

4.4.1. Análisis de estadística descriptiva

A continuación se analizan los datos que extrajimos inicialmente del sitio de la UdeG con respecto a los promedios mínimos de ingreso a las diferentes carreras de nivel superior ofrecidas en las diferentes sedes y centros universitarios.

Los objetivos específicos de este análisis son los siguientes:

- Analizar y comprender a detalle los datos estadísticos publicados por la UdeG
- Observar el comportamiento de los datos y establecer relaciones entre las distintas variables para determinar si existen correlaciones
- Determinar cuáles son los factores más relevantes para determinar las probabilidades de ingreso a las distintas carreras de la UdeG

4.4.1.1. Población y marco muestral

La población corresponde a los datos de los puntajes mínimos, aspirantes y número de admitidos a la UdeG en todas las carreras desde el calendario 2014 A hasta el calendario 2021 A.

Para este trabajo se extrajeron algunos datos de esa población para trabajar con la muestra correspondiente a las 10 carreras más demandadas de cada calendario en cuanto a número de aspirantes y número de admitidos para dar un total de 150 datos para cada campo y se tomaron todos los datos correspondientes al puntaje mínimo de todos los calendarios (cada muestra de cada calendario con alrededor de 222 datos, para dar un total de tres mil 343 datos de promedios mínimos).

4.4.1.2. Tamaño de la muestra

Para trabajar con un tamaño de muestra de 222 datos aproximadamente por calendario escolar y un nivel de confianza estándar de 95% cuando el tamaño de la población (número total de aspirantes por calendario) es en promedio de 40 mil 655, tenemos un margen de error de 6.57%.

4.4.2. Establecer variables de interés

Las siguientes tablas muestran el resumen de los datos muestrales de las variables más importantes, mismas que ayudarán a establecer las variables de interés para el trabajo:

Carrera	Frecuencia	F. Relativa	F. Porcentual	Aspirantes Promedio	Admiditos Promedio	Porcentaje Admisión Promedio	Puntaje Mínimo Promedio
	15	0.10	10%				
MEDICO CIRUJANO Y PARTERO **(CUCS)**	15	0.10	10%	**3032**	**350**	**12%**	**177.33**
ABOGADO **(CUCSH)**	15	0.10	10%	**1374**	**416**	**31%**	**148.71**
LIC. EN ENFERMERIA **(CUCS)**	15	0.10	10%	**1252**	**233**	**19%**	**149.97**
LIC. EN NEGOCIOS INTERNACIONALES **(CUCEA)**	15	0.10	10%	**1127**	**415**	**38%**	**143.14**
LIC. EN PSICOLOGIA **(CUCS)**	14	0.09	9%	**1155**	**209**	**19%**	**159.07**
LIC. EN CIRUJANO DENTISTA **(CUCS)**	14	0.09	9%	**996**	**150**	**15%**	**163.69**
LIC. EN CONTADURIA PUBLICA **(CUCEA)**	13	0.09	9%	**992**	**449**	**46%**	**137.49**
LIC. EN ARQUITECTURA **(CUAAD)**	12	0.08	8%	**1059**	**209**	**20%**	**161.21**
MEDICO CIRUJANO Y PARTERO **(CUTONALA)**	7	0.05	5%	**997**	**101**	**10%**	**173.88**
LIC. EN ADMINISTRACION **(CUCEA)**	7	0.05	5%	**900**	**377**	**43%**	**141.38**
LIC. EN CULTURA FISICA Y DEPORTES **(CUCS)**	4	0.03	3%	**858**	**175**	**20%**	**144.36**

Tabla 3. *Frecuencia con la cual distintas carreras de la UdeG han aparecido en los últimos 15 calendarios entre las más demandadas, con el promedio de aspirantes, promedio de estudiantes admitidos, porcentaje de admisión y puntaje mínimo promedio.*

En la **Tabla 3** se muestran las carreras más demandadas en los últimos años, así como el número de aspirantes en promedio, número de admitidos, porcentaje de admisión y puntaje mínimo. Si se obtiene la media del porcentaje de admisión promedio de estas carreras más demandadas es de 25%, posteriormente se usará este porcentaje para determinar la variable dicotómica `PROB_ALTA` bajo la siguiente lógica:

- Si la carrera a analizar tiene un porcentaje de admisión menor que 25%, la probabilidad se determina como probabilidad “baja” = 0.
- Si la carrera tiene un porcentaje de admisión mayor a 25%, la probabilidad puede ser considerada “alta” = 1.

Distribución de Frecuencias Promedios Estudiantes UDG						
Promedio		Frecuencia	F. Acumulada	F. Relativa	F. Porcentual	F. Relativa Acum.
60	64	10	10	0.0431	4%	0.0431
64	68	28	38	0.1207	12%	0.1638
68	72	50	88	0.2155	22%	0.3793
72	76	27	115	0.1164	12%	0.4957
76	80	19	134	0.0819	8%	0.5776
80	84	20	154	0.0862	9%	0.6638
84	88	27	181	0.1164	12%	0.7802
88	92	31	212	0.1336	13%	0.9138
92	96	13	225	0.0560	6%	0.9698
96	100	7	232	0.0302	3%	1.0000

Tabla 4. *Distribución de frecuencias de los promedios de preparatoria de los estudiantes que ingresan a la UdeG (muestra corresponde al calendario 2021 A, en el que solo se muestran promedios de preparatoria puesto que no se aplicó examen de admisión).*

La **Tabla 4** muestra la distribución de frecuencias de los promedios de preparatoria de estudiantes aspirantes a la UdeG. Se observa que los promedios que con más frecuencia aparecen son los que se encuentran entre 68 y 72, seguido de los promedios entre 88 y 92. Es decir, tenemos una distribución bimodal, con dos picos de promedios; uno entre los puntajes "bajos" y uno de puntajes "altos", como lo demuestra el histograma que se muestra en la **Figura 9**:

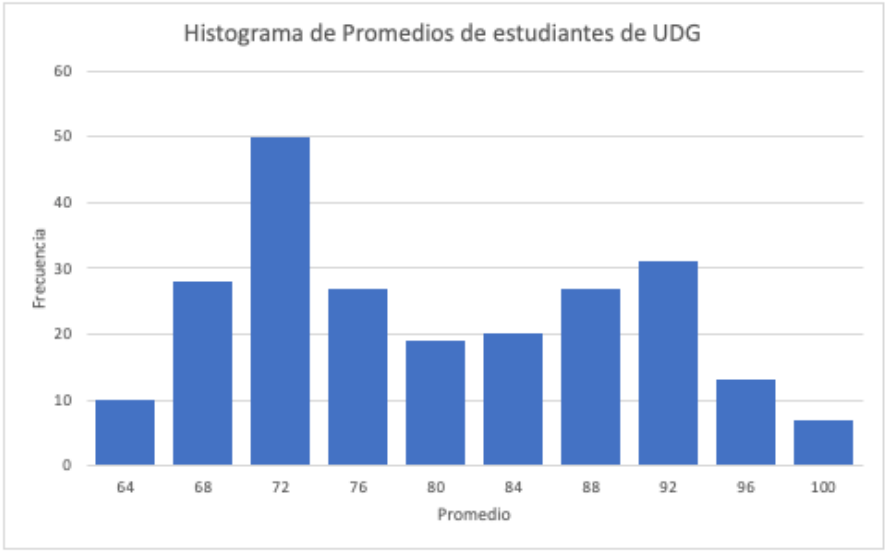

Figura 9. *Histograma de promedios de preparatoria de los aspirantes a la UdeG.*

Se utiliza como referencia el porcentaje de admisión promedio de las carreras más demandadas ya que representan un alto porcentaje de los aspirantes a la universidad. Como se muestra en la **Figura 10**, las 10 carreras con mayor demanda concentran 36% del total de aspirantes a la UdeG, mientras que el resto de aspirantes se dividen entre toda la oferta de carreras y centros universitarios restantes. Es por eso que el porcentaje de admisión promedio de estas carreras es el que determina nuestra variable dicotómica `PROB_ALTA`, misma que se utilizará más tarde en el modelo de aprendizaje automático.

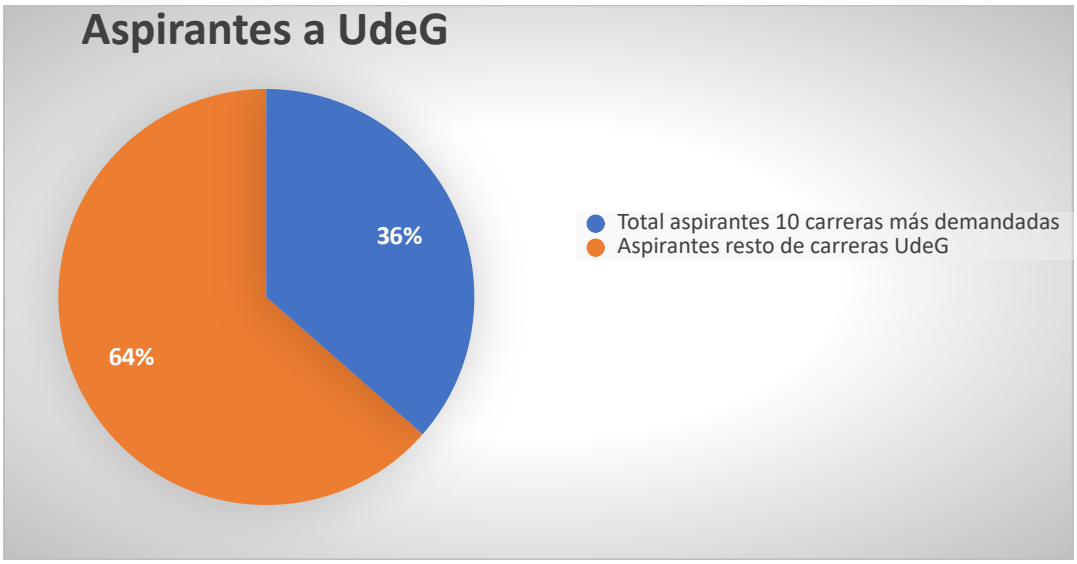

Figura 10. *Las 10 carreras más demandadas de la UdeG concentran 36% de la demanda general a la Universidad, mientras que 64% restante se divide entre todo el resto de la oferta universitaria.*

Finalmente, se observan las tendencias de promedios según el calendario escolar en el que se hagan trámites a la UdeG, esto con el fin de confirmar si es que existe una diferencia entre el calendario "A" y calendario "B" con respecto a la dificultad de ingresar según el promedio. La **Tabla 5** muestra la media de puntajes de examen de admisión más promedio de preparatoria según cada calendario:

Puntajes según calendario	
2014 A	115.19
2014 B	123.71
2015 A	115.46
2015 B	121.50
2016 A	115.13
2016 B	121.95
2017 A	117.60
2017 B	122.47
2018 A	117.56
2018 B	122.98
2019 A	117.98
2019 B	125.42

Tabla 5. *Puntajes de examen más promedio de preparatoria para distintos calendarios (A o B).*

Lo anterior deja ver que efectivamente hay una tendencia a que el calendario "B" presente promedios más altos que el calendario "A". En promedio, los puntajes del calendario "A" para los años mostrados en la tabla es de 116.48, mientras que el promedio del calendario "B" se coloca en los 123 puntos.

4.4.3. Análisis de los datos con respecto a las variables de interés

A continuación se explica el proceso para analizar los datos teniendo ya en cuenta la principal variable de interés (`PROB_ALTA`), así como el resto de variables de interés como promedios y calendarios escolares. Antes de iniciar, se importaron en un Jupyter Notebook las librerías necesarias para esta actividad, como se ilustra en la **Figura 11**:

```
#importar librerías necesarias
import numpy as np
import pandas as pd
import glob
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
%matplotlib inline
```

Figura 11. *Importación de librerías necesarias.*

Primero, se crean los *data frames* iniciales. Se crea un *data frame* donde se van a unir todos los archivos de puntajes mínimos correspondientes al calendario 'A', y asimismo, se creará un *data frame* para unir los archivos de puntajes mínimos correspondientes al calendario 'B'.

Se identifica cada calendario agregando una columna nueva asignando el valor '1' para los datos correspondientes al calendario 'A', y el valor '2' para los datos correspondientes al calendario 'B'. Esto para poder trabajar más fácilmente con números enteros en lugar de cadenas de texto o *strings*.

```
datos_udg_admision_df_A = pd.DataFrame()

#Leer datos de los archivos excel originales y crear un DataFrame para el calendario A
for f in glob.glob('modelo_ingreso_udg/datos/Puntajes_minimos_CUs_20*A.xlsx'):
    df = pd.read_excel(f, header=1, usecols=[0,1,2,3,4,5,7,8])
    df.set_index(['CENTRO'], inplace=True)
    datos_udg_admision_df_A = datos_udg_admision_df_A.append(df)

#Agregar identificador 1 para calendario A en una nueva columna
datos_udg_admision_df_A['CALENDARIO'] = '1'

datos_udg_admision_df_B = pd.DataFrame()

#Leer datos de los archivos excel originales y crear un DataFrame para el calendario B
for f in glob.glob('modelo_ingreso_udg/datos/Puntajes_minimos_CUs_20*B.xlsx'):
    df = pd.read_excel(f, header=1, usecols=[0,1,2,3,4,5,7,8])
    df.set_index(['CENTRO'], inplace=True)
    datos_udg_admision_df_B = datos_udg_admision_df_B.append(df)

#Agregar identificador 1 para calendario B en una nueva columna
datos_udg_admision_df_B['CALENDARIO'] = '2'
```

Figura 12. *Data frames iniciales.*

A continuación, se unen los dos *data frames* iniciales en uno solo y se visualiza el *data frame* para asegurar que las operaciones anteriores se realizaron con éxito, es decir, se busca comprobar que se agregó la columna con el indicador del calendario, como se demuestra en la **Figura 13**.

```
datos_udg_admision_df = datos_udg_admision_df_A.append(datos_udg_admision_df_B)

datos_udg_admision_df.head()
```

CENTRO	CARRERA	ASPIRANTES	ADMITIDOS	NO_ADMITIDOS	CUPO	PORCENTAJE_ADMISION	PUNTAJE_MINIMO	CALENDARIO
CUAAD	LICENCIATURA EN ARQUITECTURA	876	211	665	211	0.240868	159.053	1
CUAAD	LICENCIATURA EN DISEÑO DE INTERIORES Y AMBIENT...	250	70	180	70	0.280000	148.819	1
CUAAD	LICENCIATURA EN DISEÑO DE MODAS	69	40	29	40	0.579710	143.556	1
CUAAD	LICENCIATURA EN DISEÑO INDUSTRIAL	238	75	163	75	0.315126	156.847	1
CUAAD	LICENCIATURA EN DISEÑO PARA LA COMUNICACION GR...	310	167	143	167	0.538710	138.177	1

Figura 13. *Data frame conteniendo todos los datos importados así como la nueva columna de 'Calendario'.*

Ahora se revisa la información del *data frame* para comprobar el tipo de información con el que se estará trabajando.

```
#Revisamos la información del Data Frame para ver el tipo de datos
datos_udg_admision_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 4205 entries, CUAAD to nan
Data columns (total 8 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   CARRERA              4203 non-null   object
 1   ASPIRANTES           4205 non-null   int64
 2   ADMITIDOS            4205 non-null   int64
 3   NO_ADMITIDOS         4205 non-null   int64
 4   CUPO                 4205 non-null   int64
 5   PORCENTAJE_ADMISION  4205 non-null   float64
 6   PUNTAJE_MINIMO       4187 non-null   object
 7   CALENDARIO           4205 non-null   object
dtypes: float64(1), int64(4), object(3)
memory usage: 295.7+ KB
```

Figura 14. *Información del data frame.*

Como se puede ver en la **Figura 14**, los últimos campos: '`PUNTAJE_MINIMO`' y '`CALENDARIO`', aparecen como tipo `object`. Esto puede causar problemas más adelante, porque se desea que estos datos sean considerados como datos numéricos. Para solucionar este problema, se utiliza la librería Pandas para forzar a estos datos a ser considerados como datos numéricos:

```
datos_udg_admision_df['PUNTAJE_MINIMO'] = pd.to_numeric(datos_udg_admision_df['PUNTAJE_MINIMO'],errors = 'coerce')
datos_udg_admision_df['CALENDARIO'] = pd.to_numeric(datos_udg_admision_df['CALENDARIO'],errors = 'coerce')
```

Figura 15. *Conversión de tipo de datos 'object' a tipo de datos numéricos.*

Ahora se agrega una nueva columna, reemplazando la variable del porcentaje de admisión por una variable de categoría dicotómica ('1' o '0'). Esto es porque se va a

utilizar esta variable como nuestra variable principal una vez que se entrene el modelo. Para ejecutar este reemplazo, se asigna el valor de '1' a los valores cuyo porcentaje de admisión sea mayor a 25%, y el valor '0' a los valores cuyo porcentaje de admisión sea menor a 25%.

Es decir, la nueva variable, a la que se llamará '`PROB_ALTA`' va a considerar como altas ('1') las probabilidades de ingreso a una carrera con un porcentaje de admisión mayor a 25%, asimismo, considerará como probabilidad baja ('0') las carreras cuyo porcentaje de admisión sea menor a 25%. La decisión de utilizar 25% como un valor determinante se explica en detalle en el apartado **4.4.1. Análisis de estadística descriptiva**. Se comprueba nuevamente la información en el *data frame* para asegurar que se realizó el reemplazo de la variable y que se cambiaron los tipos de datos de '`object`' a '`int64`', es decir, datos numéricos.

```
#reemplazar variable de porcentaje de admisión por categoría dicotómica
#probabilidad alta = 1, cuando el % de admisión es mayor al 25%
#probabilidad baja = 0, cuando el % de admisión es menor a 25%
datos_udg_admision_df['PROB_ALTA'] = np.where(datos_udg_admision_df.PORCENTAJE_ADMISION < .25, 0, 1)
datos_udg_admision_df = datos_udg_admision_df.drop(columns = 'PORCENTAJE_ADMISION')
```

```
datos_udg_admision_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 4205 entries, CUAAD to nan
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   CARRERA         4203 non-null   object
 1   ASPIRANTES      4205 non-null   int64
 2   ADMITIDOS       4205 non-null   int64
 3   NO_ADMITIDOS    4205 non-null   int64
 4   CUPO            4205 non-null   int64
 5   PUNTAJE_MINIMO  4186 non-null   float64
 6   CALENDARIO      4205 non-null   int64
 7   PROB_ALTA       4205 non-null   int64
dtypes: float64(1), int64(6), object(1)
memory usage: 295.7+ KB
```

Figura 16. *Reemplazo de la variable PORCENTAJE_ADMISION por la variable PROB_ALTA y confirmación de conversión del tipo de datos por datos numéricos.*

A continuación se analiza el *data frame* para ver si contiene datos nulos, y de ser así, eliminar las filas de datos con datos nulos, para así asegurarse de que al entrenar el modelo solo se utilicen los elementos que contengan los datos completos, de manera que el modelo sea entrenado con datos reales y relevantes, además de que se evitan errores al procesar los datos. Se comprueba además que el nuevo *data frame* ya contiene la nueva variable dicotómica '`PROB_ALTA`'.

```
datos_udg_admision_df.isnull().values.any()
```

```
True
```

```
df = datos_udg_admision_df.dropna()
df.isnull().values.any()
```

```
False
```

```
#visualización del nuevo dataframe con la nueva variable de probabilidad
df.head()
```

CENTRO	CARRERA	ASPIRANTES	ADMITIDOS	NO_ADMITIDOS	CUPO	PUNTAJE_MINIMO	CALENDARIO	PROB_ALTA
CUAAD	LICENCIATURA EN ARQUITECTURA	876	211	665	211	159.053	1	0
CUAAD	LICENCIATURA EN DISEÑO DE INTERIORES Y AMBIENT...	250	70	180	70	148.819	1	1
CUAAD	LICENCIATURA EN DISEÑO DE MODAS	69	40	29	40	143.556	1	1
CUAAD	LICENCIATURA EN DISEÑO INDUSTRIAL	238	75	163	75	156.847	1	1
CUAAD	LICENCIATURA EN DISEÑO PARA LA COMUNICACION GR...	310	167	143	167	138.177	1	1

Figura 17. *Identificar si existen valores nulos, eliminarlos y visualizar el nuevo data frame ya con la variable dicotómica de PROB_ALTA.*

Una vez que se eliminan los datos nulos y se tienen ya las nuevas columnas con las variables de interés en el *data frame* junto con el resto de variables, se puede comenzar con la fase de entrenamiento del modelo de árbol de decisión. Pero antes se realiza la visualización de datos para entender mejor el conjunto de datos. La **Figura 18** muestra el total de carreras para cada calendario que son consideradas, antes del proceso de entrenamiento, como probabilidad alta y baja, según corresponda:

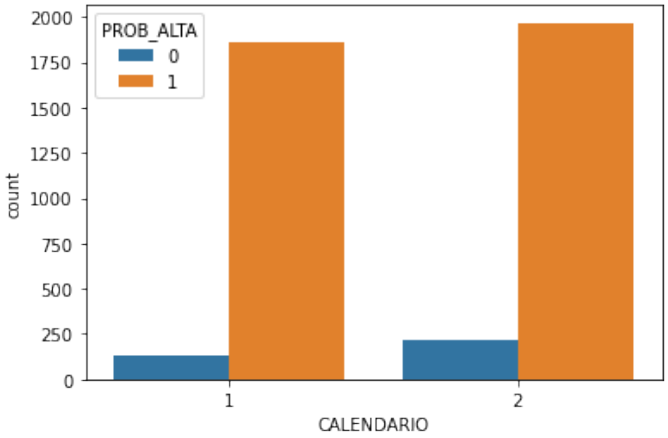

Figura 18. *Gráfica con el total de carreras según se hayan clasificado como probabilidad alta o baja para cada calendario.*

Una vez que el modelo sea entrenado tomará en cuenta este nodo raíz pero también tomará en cuenta el resto de las variables presentes en el *data frame* para continuar

clasificando cada carrera según los datos específicos del calendario (promedio mínimo de ese calendario, así como total de aspirantes, admitidos, etc.) y seguirá clasificando cada carrera según sus variables hasta formar un árbol con sus nodos dividiendo subsecuentemente cada elemento en la categoría que le corresponda (1 y 0, o probabilidad alta y probabilidad baja).

Entre los datos más relevantes se encuentran, por ejemplo, el número de admitidos *vs* el número de aspirantes. La **Figura 19** muestra la relación entre los alumnos que son admitidos y los que son rechazados, usando el *data frame* de los datos de los últimos 10 años:

```
labels = 'ASPIRANTES', 'ADMITIDOS'
sizes = [188.194458, 75.307692]
explode = (0.1, 0)

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels,
        autopct='%1.1f%%', shadow=True, startangle=90)
ax1.axis('equal')

plt.show()
```

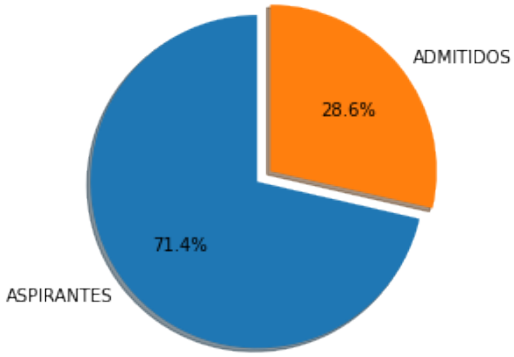

Figura 19. *Gráfica con el total de aspirantes y el total de admitidos para todas las carreras evaluadas.*

Según la gráfica, solo 28.6% de los alumnos que hacen trámite a la UdeG son admitidos, mientras que el 71.4% restante son rechazados.

Se procederá entonces a utilizar el *data frame* que se ha estado analizando para entrenar el modelo de aprendizaje automático de árbol de decisión y poder comenzar a ver los resultados.

4.5. Entrenamiento de algoritmos y análisis de los resultados

En esta sección se explica el procedimiento para entrenar el modelo de árbol de decisión y se observarán los resultados iniciales obtenidos con el modelo entrenado.

4.5.1. Entrenamiento del modelo

Como se menciona en la sección **3.4. Árboles de decisión**, "los árboles de decisión son un tipo de algoritmo de árbol de clasificación, (…) «utilizados para predecir una categoría o una clase»". Un árbol de decisión ideal divide las características del conjunto de datos en dos grupos claramente distintos (Jolly, 2018).

Como lo que se desea predecir es si una carrera posee "probabilidad alta" de ingreso o "probabilidad baja", una vez que se inicie el entrenamiento del modelo, esta debería ser la variable que el modelo escoja como nodo raíz.

A partir de esa división inicial, el modelo irá subdividiendo todos los demás elementos en el conjunto de datos en distintas categorías o subnodos, llamados nodos hijos (***children***) y **nodos hoja** (***leaf nodes***).

Primero se separa el conjunto de datos en variables dependientes y variables independientes: es decir, las variables que se tomarán en cuenta como los nodos para realizar la primera división y las divisiones subsiguientes con el modelo de aprendizaje automático, como se muestra en la **Figura 20**:

```
#Separar el conjunto de datos en variables dependientes e independientes
feature_cols = ['CALENDARIO','PUNTAJE_MINIMO','CUPO','ADMITIDOS','ASPIRANTES']
X = df[feature_cols]
y = df.PROB_ALTA
```

Figura 20. *Asignar las variables dependientes e independientes en el Jupyter Notebook.*

A continuación, se separan los datos en datos de entrenamiento y datos de prueba. Esto para que el algoritmo pueda utilizar un conjunto de datos para realizar el entrenamiento del modelo y otro conjunto de datos del mismo conjunto inicial para realizar pruebas y comprobar el nivel de eficacia del entrenamiento del modelo:

```
#separar los datos en datos de entrenamiento y datos de prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

Figura 21. *Separar el conjunto de datos en datos de entrenamiento y datos de prueba.*

Una vez hecho esto, se utiliza la librería de **Scikit Learn** para entrenar el modelo. Esta librería es un *software open source* o de uso libre que ofrece funciones para entrenar modelos de aprendizaje con algoritmos funcionales listos para utilizar en aplicaciones de ciencia de datos o proyectos de aprendizaje automático.

En específico, se usará primero la función `DecisionTreeClassifier()`, la cual se encarga de entrenar un modelo con el conjunto de datos proporcionado. La función puede recibir como parámetros el criterio de división (si no se especifica, por defecto se usa el criterio *gini*). Se hace también un ajuste del modelo usando el método `fit()`, que se encarga de construir el árbol de decisión de clasificación a partir del conjunto de datos proporcionado, y finalmente, el método `predict()`, que es el que proporcionará el valor de las predicciones de las clases del conjunto de datos de prueba. Adicionalmente, se mide la exactitud del modelo en esta primera iteración:

```
#llamar la clase de sklearn para crear la clasificación del árbol de decisión
clf = DecisionTreeClassifier()
#hacer el ajuste del modelo
clf = clf.fit(X_train,y_train)
#usar el modelo entrenado para predecir las clases del conjunto de datos de prueba
y_pred = clf.predict(X_test)
```

```
#revisar porcentaje de precisión del modelo
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9681528662420382
```

Figura 22. *Entrenamiento del modelo con las funciones de la librería de Python Scikit Learn.*

El modelo inicial arroja un nivel de exactitud de 96%, el cual es bastante alto y, por tanto, confiable.

A continuación se usará una herramienta de representación visual para poder obtener una vista de cómo luce inicialmente el árbol de decisión con los datos proporcionados. Para graficar el árbol se usará la librería llamada **Graphviz**:

```
#visualización del árbol
import graphviz
from sklearn import tree

dot_data = tree.export_graphviz(clf, out_file=None,
                    feature_names = feature_cols,
                    class_names=['0','1'],
                    filled=True, rounded=True,
                    special_characters=True)

graph = graphviz.Source(dot_data)
graph
```

Figura 23. *Código para graficar el modelo mediante la librería Graphviz.*

La representación visual del modelo generada por Graphviz es la siguiente:

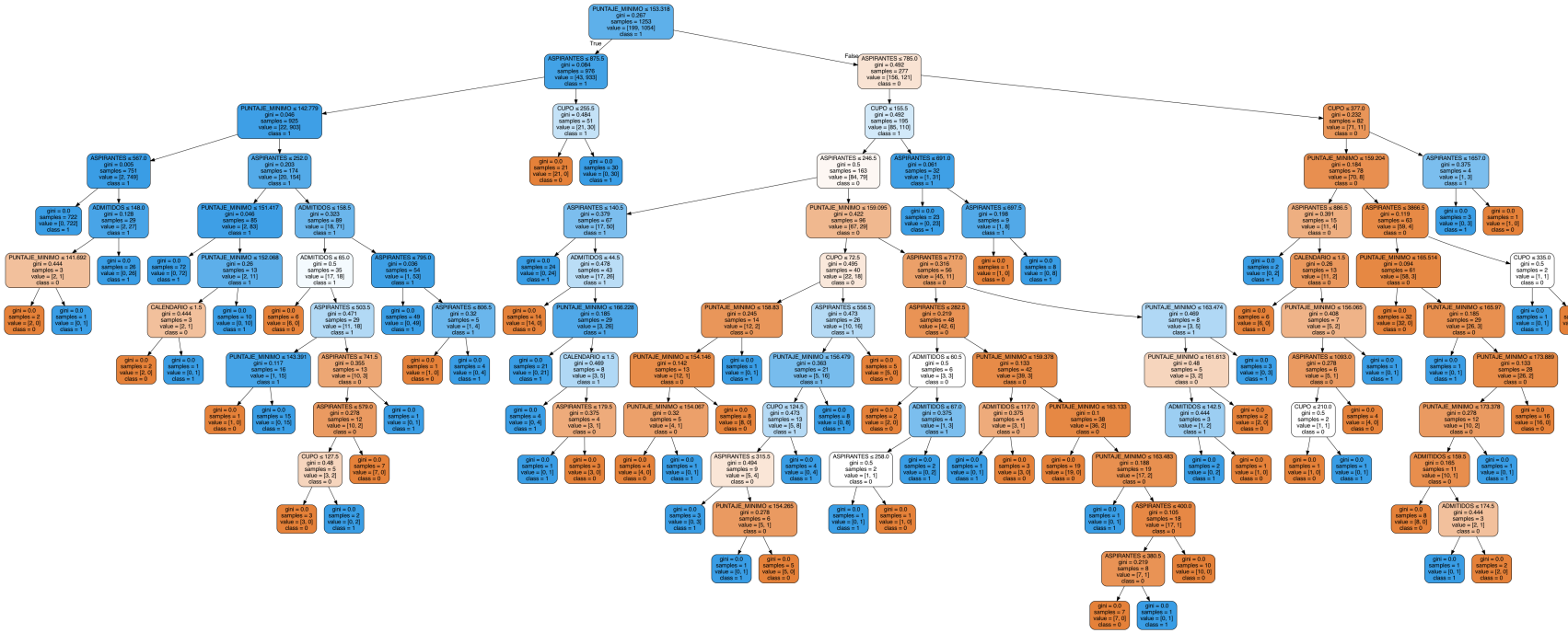

Figura 24. *Gráfico del árbol de decisión con las diferentes ramificaciones o clasificaciones creadas al entrenar el modelo.*

A pesar de que ya en la representación del árbol de decisión que se muestra en la **Figura 24** se pueden ver las diferentes clasificaciones creadas por el algoritmo, es complicado ver claramente las diferentes categorías en que fue separado el conjunto de datos.

Para realizar el proceso de "podado" del árbol de decisión, se utilizan las mismas funciones de Scikit Learn que se utilizaron previamente, pero especificando en los parámetros de la función `DecisionTreeClassifier()` el método de división a utilizar (índice gini), así como un máximo nivel de subcategorías ('4'). Asimismo, se consulta el nivel de precisión del modelo y se vuelve a graficar.

```
#volvemos a entrenar el modelo con parámetros para reducir el tamaño del árbol
clf = DecisionTreeClassifier(criterion="gini", max_depth=4)
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9348230912476723
```

```
#visualización del nuevo árbol
dot_data = tree.export_graphviz(clf, out_file=None,
                    feature_names = feature_cols,
                    class_names=['0','1'],
                    filled=True, rounded=True,
                    special_characters=True)

graph = graphviz.Source(dot_data)
graph
```

Figura 25. *Entrenamiento del modelo con parámetros específicos, consulta de precisión del modelo y graficado del nuevo modelo.*

Al forzar al algoritmo a un número específico de subcategorías posibles, se pierde un poco de precisión del modelo y se pasa de 96% de precisión del entrenamiento anterior a 93% de precisión con el entrenamiento del modelo con los nuevos parámetros. Si bien se pierde un poco de precisión, el árbol de decisión es ahora mucho más legible para nosotros, resultando en la representación del árbol siguiente:

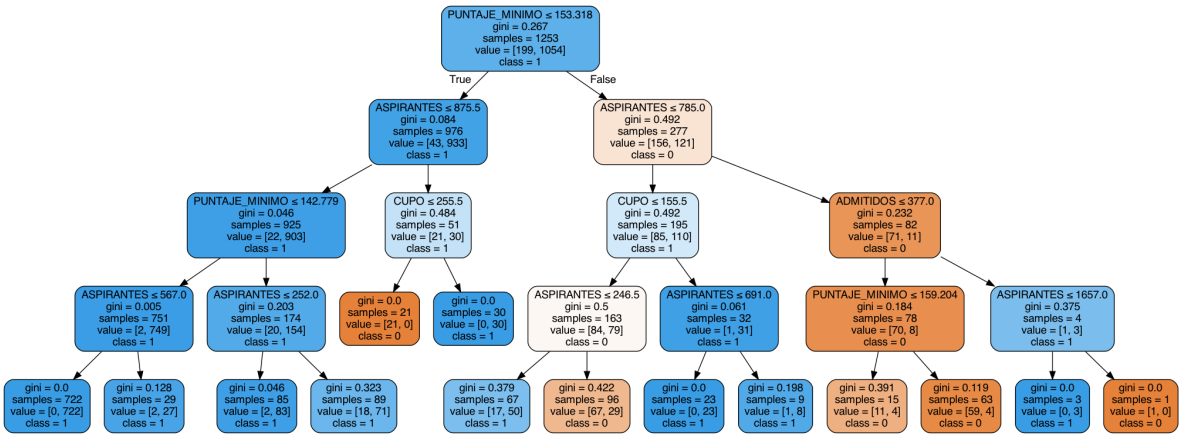

Figura 26. *Representación del árbol de decisión "podado" con parámetros específicos para una visualización más clara de la clasificación de los datos.*

En esta nueva representación del árbol de decisión, ilustrado en la **Figura 26**, se puede ver mucho más claramente cómo se han clasificado los datos y las distintas categorías en las que se ha subdividido el conjunto de datos.

Ahora el modelo cuenta con una clasificación precisa de las distintas carreras existentes en la oferta de la UdeG y puede hacer predicciones precisas con datos reales para determinar si una carrera es considerada con probabilidad alta o baja de ingreso.

En las siguientes secciones se explica el proceso para probar el modelo entrenado con datos reales y comprobar que funciona realmente y que sea posible utilizarlo para predicciones con más casos y en aplicaciones con usuarios externos que desean obtener un resultado de predicción según sus intereses particulares.

4.5.2. Comprobar el funcionamiento del modelo

Para comprobar que el modelo entrenado en las secciones anteriores funciona correctamente y es capaz de arrojar predicciones acertadas, se procederá a llamar al modelo para que proporcione una predicción con datos externos, es decir, que no fueron utilizados durante el proceso de entrenamiento y por lo tanto el modelo no cuenta con información sobre la carrera de la que se espera obtener el resultado de la predicción.

Para ello, se acude de nuevo a la página de estadísticas de la UdeG para consultar una carrera que usualmente es considerada de difícil acceso, solo para comprobar que nuestro modelo funciona correctamente. Se seleccionó un calendario que no hubiera sido

utilizado durante el proceso de entrenamiento y se seleccionaron los datos necesarios manualmente.

Para este ejercicio se eligió la carrera de Médico Cirujano del calendario B de 2021. Se copiaron manualmente los datos de Calendario (2 por ser calendario B), Puntaje mínimo requerido para esa carrera en ese calendario (180.41), Cupo (367 alumnos), así como número de admitidos (367) y número de aspirantes (4403). La respuesta que se espera recibir por parte del modelo de predicción es un cero (‘0’), el cual representa que la carrera consultada estaría en la categoría ‘0’ de nuestro árbol, lo que corresponde a carreras con probabilidad baja de ingreso.

Para poder pasar estos datos al modelo y solicitar una predicción se requiere primero utilizar el mismo formato de datos que se utilizó al entrenar el modelo; es decir, se tiene que crear un *data frame* con los datos que se consultaron manualmente y que se usarán para la predicción de prueba.

```
#crear un dataframe de prueba para una carrera altamente demandada
medico_cirujano_cucs_2021B = {'CALENDARIO':[2], 'PUNTAJE_MINIMO':[180.41], 'CUPO':[367], 'ADMITIDOS':[367], 'ASPIRANTES':[4403]}
test0_df = pd.DataFrame(medico_cirujano_cucs_2021B)
test0_df.head()
```

	CALENDARIO	PUNTAJE_MINIMO	CUPO	ADMITIDOS	ASPIRANTES
0	2	180.41	367	367	4403

Figura 27. *Creación de data frame con datos nuevos para probar el funcionamiento del modelo con una carrera que se espera sea de probabilidad baja de ingreso.*

El resultado es un *data frame* de una sola línea que se puede usar para obtener una predicción utilizando el modelo ya entrenado. Para obtener la predicción se usa la función `predict()` de la librería Scikit Learn utilizando el modelo ya entrenado y el *data frame* con los datos de la carrera a consultar.

```
#probar la predicción de la clase a que pertenece este dataframe
test0_pred = clf.predict(test0_df)
print(test0_pred)
```

```
[0]
```

Figura 28. *Predicción de prueba con datos nuevos para comprobar el funcionamiento del modelo.*

El resultado es un ‘0’, lo cual demuestra que el modelo está prediciendo correctamente que la carrera de Médico Cirujano en el CUCS en el calendario B de 2021

cae en la categoría o clasificación ‘0’ del árbol por lo tanto es una de las carreras consideradas con baja probabilidad de ingreso.

Para obtener una respuesta de ‘1’ o de carrera con probabilidades altas de ingreso, nuevamente se crea un *data frame* con datos escogidos manualmente, esta vez correspondientes a la carrera de Licenciatura en Informática del Calendario 2021 B de CUCEI:

```
lic_informatica_cucei_2021B = {'CALENDARIO':[2], 'PUNTAJE_MINIMO':[150.37], 'CUPO':[275], 'ADMITIDOS':[275], 'ASPIRANTES':[703]}
test1_df = pd.DataFrame(lic_informatica_cucei_2021B)
test1_df.head()
```

	CALENDARIO	PUNTAJE_MINIMO	CUPO	ADMITIDOS	ASPIRANTES
0	2	150.37	275	275	703

Figura 29. *Creación de data frame con datos nuevos para probar el funcionamiento del modelo con una carrera que se espera sea de probabilidad alta de ingreso.*

Se utilizó este *data frame* con la función de predicción para comprobar el funcionamiento del modelo para casos de carreras con probabilidad alta de ingreso, o categoría ‘1’ del árbol de decisión.

```
test1_pred = clf.predict(test1_df)
print(test1_pred)
```

```
[1]
```

Figura 30. *Predicción para la carrera de Licenciatura en Informática en CUCEI, calendario 2021 B.*

El resultado que arroja el modelo es un ‘1’, lo cual demuestra que el modelo es capaz de predecir qué carreras caen en la categoría ‘0’ o ‘1’, es decir, cuáles son consideradas de probabilidad baja o probabilidad alta de ingreso según sus características de calendario, puntaje mínimo, cupo, número de admitidos y número de aspirantes aún cuando estos datos no sean parte del modelo, es decir que es capaz de hacer predicciones de datos nuevos utilizando los datos con los que fue entrenado.

Con esto ya se tiene un modelo confiable y funcional que se puede utilizar para realizar predicciones de nuevas carreras de interés. Sin embargo, no es viable que un usuario externo pueda interactuar con el modelo todavía ya que hasta ahora solamente está disponible en el Jupyter Notebook utilizado para el entrenamiento del modelo y para

ejecutar las pruebas descritas anteriormente. Además, la interfaz de Jupyter Notebook no es una interfaz amigable para usuarios finales.

Para que el modelo pueda ser utilizado por usuarios externos y que puedan interactuar con el modelo para obtener predicciones sobre las carreras de su interés es necesario exportar el modelo entrenado a un archivo que pueda ser utilizado por ejemplo, por una aplicación web.

En las siguientes secciones se explica el proceso seguido para exportar el modelo y utilizarlo para interactuar con una interfaz gráfica.

4.6. Desarrollo de interfaz gráfica

Para que la aplicación pueda utilizar el modelo de aprendizaje automático entrenado anteriormente, se requiere guardar el modelo ya entrenado como un archivo binario que pueda utilizarse en el motor de la aplicación (*back-end*) para que la interfaz gráfica (*front-end*) pueda mostrar los resultados de las predicciones a los usuarios. Para almacenar el modelo entrenado se utiliza un módulo de Python llamado Pickle (https://docs.python.org/3/library/pickle.html).

```
#importamos la librería Pickle para poder almacenar el modelo entrenado en disco
import pickle
```

```
#usamos Pickle para guardar el modelo entrenado como un archivo que podemos utilizar más tarde
with open('/Users/mariomorales/Documents/ITESO/TOG/modelo_ingreso_udg/model/modelo_udg.pkl', 'wb') as file:
    pickle.dump(clf, file, protocol=pickle.HIGHEST_PROTOCOL)
```

Figura 31. *Importación del módulo Pickle y almacenamiento del modelo entrenado como un archivo binario.*

Antes de pasar al desarrollo de la interfaz gráfica, se realizaron pruebas con los mismos *data frames* que se utilizaron anteriormente para verificar que el modelo estuviera funcionando, pero esta vez se utilizó el archivo binario creado por Pickle para ejecutar las pruebas en lugar de llamar el modelo directamente desde Jupyter Notebook. Esto para comprobar que el modelo almacenado como archivo binario contenga la información necesaria para hacer las predicciones correctamente:

```
with open(f'/Users/mariomorales/Documents/ITESO/TOG/modelo_ingreso_udg/model/modelo_udg.pkl', 'rb') as f:
    model = pickle.load(f)
```

```
test0_pickle_pred = model.predict(test0_df)
print(test0_pred)
```

```
[0]
```

```
test1_pickle_pred = model.predict(test1_df)
print(test1_pred)
```

```
[1]
```

Figura 32. *Cargar el archivo creado por Pickle y ejecución de pruebas para comprobar que el modelo tiene el comportamiento esperado.*

Se comprueba que las pruebas arrojan los mismos resultados con el modelo creado por Pickle que con las pruebas realizadas directamente con el modelo desde Jupyter Notebook, así que se guardará este archivo como el modelo que servirá como *back-end* de la aplicación.

Para el desarrollo de la interfaz gráfica se utilizó un *web framework* llamado Flask (https://github.com/pallets/flask), el cual utiliza el estándar WSGI (*Web Server Gateway Interface*) del lenguaje Python para establecer la comunicación entre el servidor web y la aplicación Python que se desarrolló.

Las funcionalidades mínimas que se busca que la aplicación ofrezca al usuario son:

- Que el usuario pueda seleccionar la carrera de interés entre las distintas opciones que ofrece la UdeG.
- Que la aplicación procese la información según la selección del usuario utilizando el modelo de aprendizaje automático de árbol de decisión en el *back-end*.
- Que la aplicación muestre al usuario el resultado de la predicción: probabilidad alta o baja de ingreso según el resultado que arroje el modelo.

4.6.1. Estructura de la aplicación Flask

El siguiente gráfico muestra la estructura de la aplicación o el directorio del proyecto, incluyendo los archivos de la aplicación Python, los archivos estáticos (HTML y CSS), así como los archivos de configuración y el archivo con el modelo de aprendizaje automático:

```
webapp
├──app
│   ├──static/css
│   │   └──style.css
│   ├──templates
│   │   └──index.html
│   └──main.py
├──data
│   └──PuntajesMinimos_last.csv
├──venv
├──Procfile
├──modelo_udg.pkl
├──requirements.txt
├──runtime.txt
├──wsgi.py
└──README.md
```

El directorio `webapp` es el directorio principal donde se alojan todos los archivos que conforman la aplicación.

El directorio `app` es donde se almacenan los archivos estáticos de estilo (`style.css`) y HTML (`index.html`), así como la lógica principal de la aplicación Python (`main.py`).

El directorio `data` contiene un archivo CSV (`PuntajesMinimos.csv`) con información de puntajes mínimos, número de admitidos, cupo y número de aspirantes de los últimos dos calendarios de admisiones a la UdeG. Estos datos se usarán como referencia para efectuar las predicciones: el modelo tomará estos datos de referencia (que no fueron parte del entrenamiento del modelo) para contrastarlos con el árbol de decisión de nuestro modelo (que contiene información de los calendarios de un periodo de 10 años) y determinar a qué categoría pertenece la carrera seleccionada por el usuario (probabilidad alta o probabilidad baja).

El directorio `venv` contiene datos de configuración para el ambiente virtual de Python requerido para ejecutar la aplicación.

El archivo `Procfile` es un archivo necesario para especificar los comandos que se ejecutarán al iniciar la aplicación en el servidor web del servicio de alojamiento llamado Heroku, el cual utilizaremos para alojar la aplicación web.

El archivo llamado `modelo_udg.pkl` es el modelo de aprendizaje automático que se creó anteriormente usando Pickle y es el que utilizará la aplicación para efectuar las predicciones.

El archivo de texto `requirements.txt` especifica los requerimientos técnicos o dependencias de la aplicación Python. Este archivo se genera automáticamente con el comando `pip freeze > requirements.txt`.

El archivo `runtime.txt` especifica al servicio de Heroku el ambiente de Python necesario para ejecutar la aplicación.

El archivo `wsgi.py` es el archivo que se ejecutará inicialmente por el servidor web. Este importa la aplicación principal (`main.py`) y la ejecuta, funcionando como una interfaz entre el servidor web y la lógica principal de la aplicación.

4.6.2. *Back-end*

Como ya se mencionó antes, el *back-end* o motor de la aplicación es la parte de la aplicación que se encargará de la comunicación con el modelo de aprendizaje automático así como el resto de la lógica de la aplicación; es decir, manejar las solicitudes HTTP del cliente y enviarlas al servidor web según corresponda.

Todo el código necesario para esto se encuentra en el archivo `main.py`. La primera parte de este archivo contiene las librerías necesarias, una línea de código para inicializar la aplicación, y una línea de código que se encarga de cargar el modelo entrenado de árbol de decisión:

```
1  import numpy as np
2  from flask import Flask, request, jsonify, render_template, redirect, url_for
3  import pickle
4  import pandas as pd
5  import csv
6
7  #Initialize Flask app
8  app = Flask(__name__)
9
10 #Loads the trained model
11 model = pickle.load(open('modelo_udg.pkl', 'rb'))
12
```

Figura 33. *Importación de librerías, inicialización de la aplicación y carga del modelo.*

La siguiente sección del archivo contiene la lógica para la ruta principal de la aplicación (o *home "/"*); cuando la solicitud HTTP del cliente es una solicitud `GET`, la aplicación carga la plantilla HTML definida en el archivo `index.html`, el cual contiene la estructura de la página web que se mostrará al usuario. También se agregó un método `POST` para permitir a los usuarios limpiar la selección y volver a cargar la página principal para hacer una nueva selección sin tener que salir de la aplicación y entrar nuevamente.

```
13  @app.route("/", methods = ['GET', 'POST']) # Homepage and data read
14  def index():
15      '''
16      Reads data from csv file
17      '''
18      with open('data/PuntajesMinimos_last.csv') as csv_file:
19          data = csv.reader(csv_file, delimiter=',')
20          carreras = []
21          for row in data:
22              carreras.append({
23                  "CALENDARIO": row[0],
24                  "CENTRO": row[1],
25                  "CARRERA": row[2],
26                  "PUNTAJE_MINIMO": row[3],
27                  "CUPO": row[4],
28                  "ADMITIDOS": row[5],
29                  "ASPIRANTES": row[6]
30              })
31
32      return render_template("index.html", carreras=carreras)
33
```

Figura 34. *Definición de lógica para la ruta principal o* home: "/".

La siguiente sección define la lógica de la ruta `/select_value`, que es la que maneja la solicitud de predicción hecha por el usuario. Aquí se define la estructura del menú desplegable que mostrará al usuario las diferentes opciones de carreras a elegir según su interés para poder elegir una de ellas y solicitar la predicción. Una vez que se selecciona la carrera y se oprime el botón de **Obtener predicción**, se envía la solicitud `POST` con los datos de esa carrera, se procesa usando el modelo de aprendizaje automático y a continuación se muestra al usuario el resultado de la predicción en la misma página (`index.html`).

```python
@app.route("/select_value", methods=['GET','POST'])
def select_value():
    '''
    Reads the selection from the dropdown menu
    '''
    seleccion = request.form.get('seleccion_carrera')

    #Clean list to remove white spaces and new lines
    seleccion_nnl = seleccion.replace('\r\n','')
    selection_clean = seleccion_nnl.replace(' ','')
    seleccion_list = selection_clean.split(',')

    #Creates the data frame to be sent to the predicion model
    input_variables = pd.DataFrame(seleccion_list).T

    #Executes the prediction using the trained model
    # print(input_variables)
    prediction = model.predict(input_variables)[0]

    #if statement to determine if the probability is high or low according to the prediction result
    if prediction == 0:
        result = 'Probabilidad baja'
    elif prediction == 1:
        result = 'Probabilidad alta'

    # rendering the predicted result on the page
    return render_template('index.html', prediction_text=result)

```

Figura 35. *Definición de lógica para la ruta* "/select_value".

Finalmente, se definen el resto de rutas que conformarán la aplicación y se define la instrucción para indicar que se ejecute el código de la aplicación cuando se corra el programa.

```python

@app.route('/about/')
def about():
    return render_template('about.html')

@app.route('/contact/')
def contact():
    return render_template('contact.html')

if __name__ == "__main__":
    app.run(debug=True)
```

Figura 36. *Sección final donde se definen el resto de rutas y se define la instrucción para ejecutar el programa.*

4.6.3. *Front-end*

El *front-end* o la interfaz gráfica está definida en el archivo `index.html` que se encuentra en el directorio `templates`. En este archivo se define cómo se verá la página principal de la aplicación para los usuarios finales cuando ingresen a la aplicación desde un navegador web.

Los elementos principales de este archivo son:

- Un menú desplegable con la oferta de carreras disponibles para seleccionar
- Un botón para enviar la solicitud de predicción al servidor
- Un botón para iniciar una nueva selección
- Una página web que muestra el menú, una breve explicación de su funcionamiento y un menú principal de navegación

El código para mostrar el menú desplegable, el botón de envío de la solicitud de predicción, así como para mostrar el resultado o generar una nueva solicitud es el siguiente:

```
33
34      <h1>Predicción de Posibilidades de Ingreso a la UDG</h1>
35      <form class="form-inline" method="POST" action="{{ url_for('select_value') }}">
36          <div class="form-group">
37            <p>Selecciona la carrera de interés:</p>
38                  <p> CALENDARIO (A=1, B=2) - CENTRO - CARRERA</p>
39              <div class="input-group">
40                      <select name="seleccion_carrera" class="selectpicker form-control">
41                          {% for carrera in carreras %}
42                          <option value = "{{ carrera.CALENDARIO }},
43                                           {{ carrera.PUNTAJE_MINIMO }},
44                                           {{ carrera.CUPO }},
45                                           {{ carrera.ADMITIDOS }},
46                                           {{ carrera.ASPIRANTES }}"
47                                           SELECTED>
48                                           {{ carrera.CALENDARIO }} -
49                                           {{ carrera.CENTRO }} -
50                                           {{ carrera.CARRERA }}
51                          </option>
52                          {% endfor %}
53                      </select>
54              </div>
55              <br>
56              <button type="submit" class="btn btn-default">Obtener predicción</button>
57          </div>
58      </form>
59      <p>{{ prediction_text }}</p>
60      <form method="POST" action="/">
61          <button type="submit" class="btn btn-secondary">Obtener nueva predicción</button>
62      </form>
```

Figura 37. *Código HTML para mostrar el menú desplegable, enviar la solicitud de predicción y mostrar el resultado en pantalla o crear una nueva solicitud.*

Además del código HTML la interfaz gráfica utiliza algunas definiciones en CSS para determinar cómo se verá la página web en el navegador. Estas configuraciones se encuentran en el archivo `style.css`.

4.6.4. Desplegar la aplicación

Una vez que la aplicación está lista y con una interfaz gráfica para que los usuarios puedan interactuar con ella, el último paso es desplegar la aplicación y hacerla disponible a los usuarios finales a través de Internet. Para esto se utilizó un servicio de alojamiento de aplicaciones web llamado Heroku (https://www.heroku.com), que permite el alojamiento de aplicaciones web en diferentes lenguajes de programación, incluyendo Python.

A continuación se describen los pasos para desplegar la aplicación en Heroku:

NOTA: Es necesario tener una cuenta creada en Heroku y descargar el Heroku CLI.

1. Crear ambiente virtual de Python.

   En el directorio de la aplicación, correr:

   ```
   $ python3 -m venv env
   $ source env/bin/activate
   ```

2. Instalar Flask y Gunicorn en el ambiente virtual:

   ```
   $ pip install flask gunicorn
   ```

3. Crear el archivo con las dependencias de la aplicación:

   ```
   $ pip freeze > requirements.txt
   ```

4. Crear archivo llamado `Procfile`:

   ```
   $ vim Procfile
   ```

5. Colocar el siguiente texto en el archivo `Procfile` y guardar cambios:

   ```
   web: gunicorn wsgi:app
   ```

6. Crear archivo llamado `runtime.txt`:

   ```
   $ vim runtime.txt
   ```

7. Escribir en el archivo `runtime.txt` la versión de Python a utilizar:

   ```
   python-3.8.11
   ```

8. Crear un archivo llamado `/wsgi.py` con el siguiente texto:

   ```
   from app.main import app
   if __name__ == "__main__":
       app.run(debug=True)
   ```

9. Crear un repositorio de git.

   ```
   $ git init
   ```

10. Correr el siguiente comando para hacer login en Heroku:

```
$ heroku login
```

11. Crear la aplicación en Heroku:

```
$ heroku create predict-udg
```

12. Hacer un push para enviar los archivos a los servidores de Heroku:

```
$ git push heroku main
```

13. Esperar a que los cambios se terminen de subir a los servidores.

Una vez que Heroku termina de hacer las configuraciones con los archivos enviados al repositorio, se generará una URL en la cual estará disponible la aplicación que se ha desplegado. En este caso la URL es la siguiente: https://predict-udg.herokuapp.com.

La interfaz gráfica ya desplegada en Heroku se ve como se muestra en la siguiente figura:

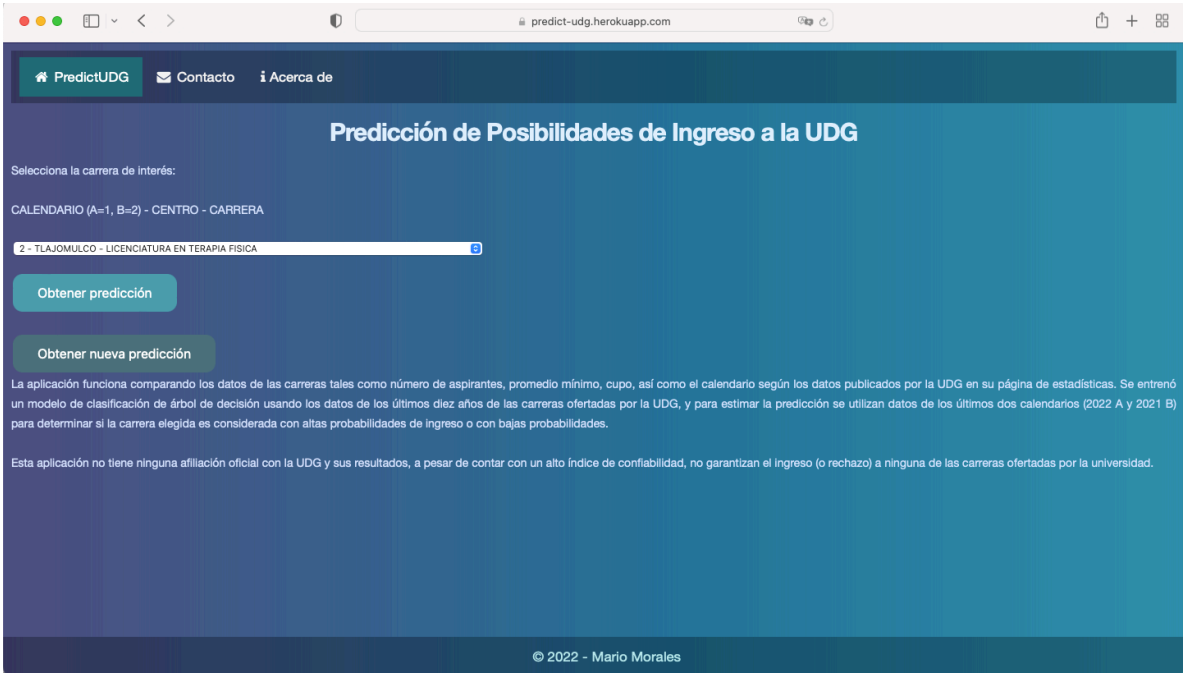

Figura 38. *Código HTML para mostrar el menú desplegable, enviar la solicitud de predicción y mostrar el resultado en pantalla o crear una nueva solicitud.*

Una vez que los usuarios seleccionan una carrera de interés y envían la solicitud de predicción, la página muestra los resultados en la parte baja, indicando si se trata de una carrera considerada de probabilidad alta o probabilidad baja de ingreso, con base en las categorías creadas por el modelo de árbol de decisión.

5. RESULTADOS Y DISCUSIÓN

Resumen: *En esta sección se discute el resultado del presente trabajo y se discute la relevancia o alcance de los mismos.*

5.1. Resultados

Los resultados obtenidos con el desarrollo del presente trabajo demuestran que es posible, mediante el análisis y procesamiento de grandes cantidades de datos, crear proyectos nuevos que ayuden a afrontar necesidades de usuarios o simplemente analizar problemas existentes desde nuevas perspectivas.

En este caso, se generó un sistema que analiza una solicitud de un usuario y procesa esa información usando el modelo de aprendizaje automático de árbol de decisión que se entrenó como parte de este proyecto, mismo que utilizó datos de estadísticas escolares disponibles públicamente correspondientes a un periodo de 10 años, para poder generar una predicción sobre si una carrera profesional puede considerarse con probabilidades altas o bajas de ingreso.

Para comprobar el funcionamiento de la aplicación, se generaron dos solicitudes distintas para ver los resultados que arroja el sistema. En el primer caso, se selecciona del Calendario "A", en el Centro Universitario de Ciencias de la Salud, la carrera de Licenciatura en Psicología, y se realiza la solicitud para obtener la predicción:

Figura 39. *Solicitud de predicción para la carrera de Licenciatura en Psicología en el Centro Universitario de Ciencias de la Salud para el Calendario "A".*

El resultado que arroja el sistema es “Probabilidad baja”, lo cual significa que esa selección representa una de las carreras que caen bajo la categoría del árbol de las que tienen bajas probabilidades de ingreso:

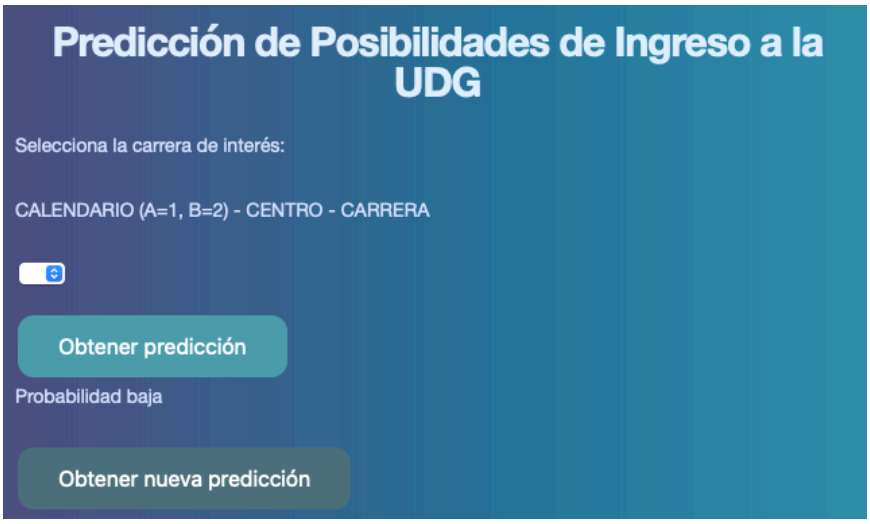

Figura 40. *Resultado de una solicitud que muestra como respuesta del sistema: “Probabilidad baja”.*

A continuación se realizó una solicitud para una carrera distinta; en este caso, se seleccionó una carrera para el Calendario “B”, en el Centro Universitario de Ciencias Económico Administrativas, para la carrera de Licenciatura en Contaduría Pública:

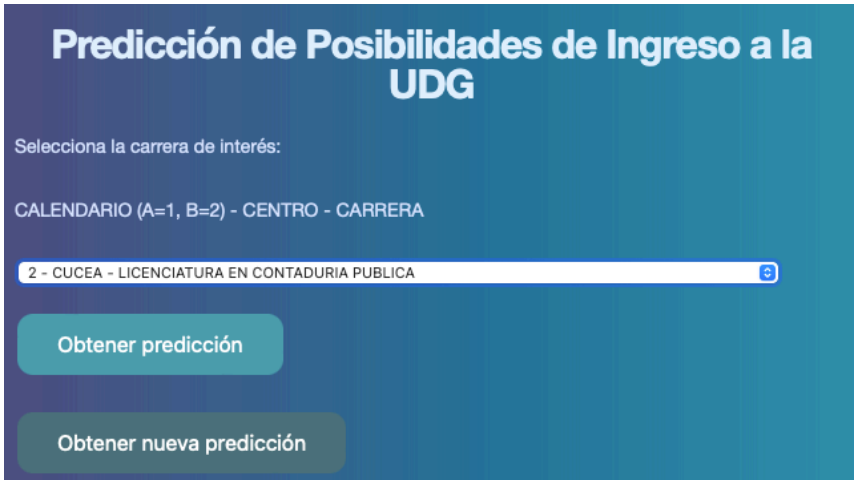

Figura 41. *Solicitud de predicción para la carrera de Licenciatura en Contaduría Pública en el Centro Universitario de Ciencias Económico Administrativas para el Calendario “B”.*

El resultado que arroja el sistema en este caso es de “Probabilidad Alta”, por tanto esta carrera se encuentra en la categoría de las carreras con alta probabilidad de ingreso según las categorías del modelo de árbol de decisión.

Figura 42. *Resultado de una solicitud que muestra como respuesta del sistema: "Probabilidad alta".*

5.2. Discusión

Al observar los resultados obtenidos se comprueba que el sistema es capaz de manejar distintas solicitudes y proporcionar predicciones según la solicitud. Se puede decir que es un sistema que cubre un caso de uso básico, ya que solo proporciona una predicción basada en datos correspondientes a las carreras de interés, pero no toma en cuenta otros datos de entrada que pudieran resultar de interés tales como promedio del alumno interesado, escuela de procedencia, datos demográficos como edad, entidad de residencia, sexo u otros.

Tomando en cuenta los objetivos iniciales de este proyecto se puede decir que se han cubierto los objetivos particulares, pues durante el desarrollo de este trabajo fue posible:

- Analizar y procesar los datos estadísticos disponibles públicamente en el sitio de estadísticas de la UdeG.
- Aplicar un modelo de aprendizaje automático de árbol de decisión para obtener predicciones de ingreso a distintas carreras de la oferta de la UdeG.
- Aplicar los resultados del análisis y procesamiento de los datos en una interfaz gráfica que permite enviar las solicitudes de predicciones así como ver los resultados de las mismas.

- Evaluar la eficacia del modelo. En este último punto, se considera que el modelo puede ser replicado para su uso en otras universidades u otras regiones, siempre y cuando se tenga acceso a los datos necesarios para hacer el entrenamiento del modelo.

Por su parte, el objetivo general del trabajo ("…encontrar las probabilidades que un determinado estudiante tiene de ingresar a alguna carrera universitaria"), aunque fue cubierto en parte, no ha sido cubierto en su totalidad, por lo que en la siguiente sección se detallan algunos puntos de mejoras que pueden ayudar a continuar el trabajo y ampliar el alcance del proyecto. En especial hace falta tomar en cuenta datos particulares del estudiante interesado, quien vaya a utilizar la aplicación, y mostrar resultados que se ajusten a su perfil particular y a sus intereses específicos, así como tomar en cuenta sus características actuales de promedio escolar, procedencia, etcétera.

6. CONCLUSIONES

Resumen: *En esta sección se presentan las conclusiones y se mencionan algunos aspectos que pudieran considerarse para desarrollarse como trabajo futuro siguiendo la línea de este proyecto.*

6.1. Conclusiones

Este trabajo surgió como un proyecto de interés personal pero que fue adquiriendo fuerza y forma a través de los meses y años como parte del curso de la Maestría en Informática Aplicada en ITESO.

De la inquietud inicial de comenzar a explorar los modelos de predicción o sistemas de recomendación que utilizaban modelos de aprendizaje automático a su ejecución y conclusión o materialización en lo que se presenta en este trabajo, hubo mucho trabajo que fue desechado, muchas horas de programación y análisis de las posibilidades o diferentes versiones que podía tomar el desarrollo de esta aplicación, y la gran mayoría de ellas no funcionaron o simplemente terminaban en una dirección distinta a la que se había concebido originalmente. Es verdad también que en el camino hubo que cambiar la dirección (inicialmente se comenzó a explorar los modelos de regresión lineal, hasta que se identificó que no iban a funcionar para este trabajo debido a la naturaleza de los datos con los que se contaban). Se consideró incluso desechar los datos que ya habían sido descargados y limpiados para recolectar y volver a organizar una nueva base de datos desde cero, haciendo encuestas a estudiantes de preparatoria. Sin embargo, al final se usaron los datos iniciales y se cambió el rumbo de la exploración de los modelos para crear el sistema de predicción.

Todo esto puede hacer parecer que hubo mucho trabajo que fue en vano, pero en realidad no fue del todo así. Una buena parte del trabajo del análisis de datos es llegar a conocer realmente los datos con los que se cuenta y analizar las posibilidades que ofrecen. Pero sí es verdad que se debe dedicar un tiempo considerable a este análisis inicial de los datos, así como al proceso de limpieza de los mismos.

Para poder tener un corpus de datos útil y que pueda asegurar buenos resultados en el trabajo posterior de un proyecto de ciencia de datos, es de vital importancia que todos los campos cuenten con datos limpios y bien estructurados. Buena parte del trabajo consiste en dar consistencia a la base de datos; asegurarse de que los datos provenientes de distintas fuentes cuenten al final con una misma estructura facilitará en gran medida su manipulación posterior.

Hay que mencionar que las librerías como Pandas, Matplotlib, SciPy, NumPy, SciKit-Learn, entre otras, facilitan mucho el trabajo ya que ofrecen comandos y herramientas que permiten agilizar la manipulación de los datos y operaciones para el análisis de los mismos, así como diferentes herramientas de visualización que permiten la toma de decisiones sobre siguientes pasos en los proyectos.

En este caso prácticamente todo el proceso de trabajo se hizo utilizando la plataforma de Jupyter Notebook, que permite crear espacios de trabajo donde es posible correr código en Python y utilizar las librerías mencionadas anteriormente con todas sus diferentes funcionalidades, desde gráficos y visualizaciones hasta el procesamiento de grandes cantidades de datos para generar *data frames* o analizar los mismos para brindar datos que ayuden a entender los datos contenidos en ellos.

Otra cosa que se puede mencionar es que gran parte de los recursos que se utilizaron durante el desarrollo de este proyecto eran desconocidos para mí, en parte o en su totalidad, y fueron las distintas necesidades que fueron apareciendo durante el desarrollo del proyecto las que me llevó a preguntar a profesores y compañeros, así como a investigar en recursos físicos y en línea para aprender cómo aplicar y llevar a cabo las distintas etapas del proyecto, desde el análisis inicial de los datos, hasta su consolidación como aplicación web. Sin embargo, todos estos recursos están focalizados en su tópico o área específica, y son muy pocos los recursos que contienen información que cubra casos como este de principio a fin.

Al final, se puede considerar este trabajo como un compendio o manual que explica cada parte del proceso de un trabajo de ciencia de datos, desde su etapa inicial hasta su codificación, incluyendo su publicación y despliegue como aplicación web, y puede ser utilizado ya sea para aprender acerca de este tipo de proyectos, llevar a cabo un proyecto similar aplicado a un caso de uso distinto, o incluso ampliar o mejorar las funcionalidades desarrolladas en este trabajo.

Si bien todo puede ser objeto de mejoras continuas, el trabajo que se presenta aquí funciona como un todo que cumple con el objetivo inicial de utilizar los datos estadísticos de la UdeG para generar un modelo de aprendizaje automático que sea capaz de arrojar

predicciones sobre probabilidades de ingreso a las distintas carreras que ofrece la universidad. Con todo, en la siguiente sección se detallan los puntos de mejora y las funcionalidades que se estiman pueden adicionarse como trabajo a futuro.

6.2. Trabajo a futuro

A continuación se listan algunas mejoras que pudieran añadir más o mejores funcionalidades para robustecer el sistema generado para este trabajo y mejorar la experiencia del usuario, así como ofrecer una respuesta mucho más útil o completa a los usuarios finales.

- Tomar en cuenta más datos que pueda ingresar el usuario tales como:
    - Promedio de preparatoria
    - Escuela de procedencia
    - Datos demográficos
- Proporcionar no solo la predicción de la carrera sino también un estimado de qué promedio necesita obtener el estudiante en el examen de ingreso para tener mejores posibilidades de ingreso
- Que la interfaz gráfica ofrezca una mejor opción de selección de la carrera de interés: en lugar de que sea un solo menú desplegable, separar los distintos campos para seleccionar cada uno por separado: Calendario, Centro y Carrera.
- Que al hacer clic en el botón para obtener la predicción la interfaz no borre la selección hecha sino que muestre la selección hecha y el resultado de la predicción.
- Que la predicción no solo muestre la leyenda “Probabilidad alta” o “Probabilidad baja” sino que ofrezca más datos contextuales para entender mejor el resultado.

Otra serie de funcionalidades que deberán ser consideradas para el mantenimiento o continuación del presente trabajo son:

- Actualización en futuras versiones de la aplicación que incluyan los datos más actuales de las carreras tanto para alimentar al modelo como para utilizar los datos más recientes de promedios y demanda para hacer las estimaciones.

- Mejorar la interfaz gráfica para hacerla más amigable con dispositivos móviles.
- Promover la aplicación entre los aspirantes a la universidad para dar a conocer el sistema y sus funcionalidades.
- Promover la aplicación en la universidad o entre instituciones que proporcionen cursos o clases de regularización para que sea utilizada como método informativo sobre las posibilidades de ingreso de los alumnos según sus condiciones actuales de promedio, etc.

Se considera que el estado actual de la aplicación es un buen punto de inicio para continuar creciendo sus funcionalidades y aplicaciones, así como un buen precedente para que más instituciones publiquen datos que puedan ser utilizados para su análisis ya que pueden ser útiles para más trabajos similares o llevar más a profundidad los proyectos ya existentes.

BIBLIOGRAFÍA

Referencias bibliográficas

Bonaccorso, G. (2018). *Machine Learning Algorithms - Second Edition.* Birmingham, UK: Packt Publishing Ltd.

Cady, F. (2017). *The Data Science Handbook.* Hoboken, NJ: John Wiley & Sons, Inc.

Cao, L. (2017). "Data Science: Challenges and Directions". *Communications of the ACM* (pp. 59-68).

Dean, J. (2014). *Data Mining, and Machine Learning: Value Creation for Business Leaders and Practitioners.* John Wiley & Sons, Incorporated.

Garreta, R., y Moncecchi, G. (2013). *Learning scikit-learn : Machine learning in python*. Packt Publishing, Limited.

Grus, J. (2019). *Data Science from Scratch.* Sebastopol, CA: O'Reilly Media, Inc.

Jamison, J. (2017). "Applying Machine Learning to Predict Davidson College's Admissions Yield". *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 765–766).

Jolly, K. (2018). *Machine Learning with Scikit-learn Quick Start Guide: Classification, Regression, and Clustering Techniques in Python*. Packt Publishing.

Lantz, B. (2013). *Machine Learning with R.* ProQuest Ebook Central.

Pedregoza *et al*. (2011) "Scikit-learn: Machine Learning in Python" en JMLR 12, pp. 2825-2830.

Van Geenen, D., y Wieringa, M. (2020). "Approaching data visualizations as interfaces: An empirical demonstration of how data are imag(in)ed". En M. Engebretsen, y H. Kennedy, *Data Visualization in Society* (pp. 141-156). Amsterdam: Amsterdam University Press.

Weber, W. (2020). "Exploring narrativity in data visualization in journalism". En M. Engebretsen, y H. Kennedy, *Data Visualization in Society* (pp. 295-312). Amsterdam: Amsterdam University Press.

Wu, J., Cai, W., Watkins, D., y Glanz, J. (22 de marzo de 2020). *How the Virus Got Out*. Recuperado el 03 de marzo 2021, de The New York Times: https://www.nytimes.com/interactive/2020/03/22/world/coronavirus-spread.html.

Xing, W., y Guan, M. (2017). "The Research Proposal of Racial Inequalities against Asian American in College Admission based on Regression Model". *Proceedings of the 8th International Conference on E-business, Management and Economics (ICEME 2017)* (pp. 62–65).

Referencias web

1.10. Decision Trees. (2007). Scikit-Learn. Recuperado: 10 de Marzo de 2022, desde: https://scikit-learn.org/stable/modules/tree.html

Pandas.DataFrame – pandas 1.4.2 documentation. (2008). Pandas.DataFrame API Reference Documentation. Recuperado: 28 de abril de 2022, desde: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html