Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado
Ingeniería Informática

Sistema Informático de apoyo a la orientación de estudiantes basado en test psicotécnicos e indicadores de educación

Computer-based Tool to support student orientation based on psycho-technical tests and education indicators

Pablo Bethencourt Díaz

La Laguna, 10 de septiembre de 2020

D. **Coromoto León Hernández**, con N.I.F. 78.605.216-W, profesora Catedrático de Universidad del área de Lenguajes y Sistemas Informáticos, adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora,

D. **Carmen Elvira Ramos Domínguez**, con N.I.F. 45.438.586-Q, profesora Titular de Universidad del área de Estadística e Investigación Operativa, adscrita al Departamento de de Matemática Fundamental, Estadística e Investigación Operativa de la Universidad de La Laguna, como tutora,

I N F O R M A (N)

Que la presente memoria titulada:

”*Sistema Informático de apoyo a la orientación de estudiantes basado en test psicotécnicos e indicadores de educación*”

ha sido realizada bajo su dirección por el estudiante D. **Pablo Bethencourt Díaz**, con N.I.F. 54.057.126-L.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 10 de septiembre de 2020

Agradecimientos

A mi tutora Coromoto León Hernández, por confiar en mi para este proyecto y orientarme durante todo el desarrollo.

A mi cotutora Carmen Elvira Ramos Domínguez, por su asesoramiento y por todos los conocimientos aportados.

A mi pareja, Elena, por todo el ánimo y cariño recibido durante esta experiencia.

A mis padres y hermana, por apoyarme siempre durante todos estos años.

Licencia

Resumen

El abandono universitario es un fenómeno negativo con gran impacto dentro de la sociedad española. Tratar de solventar esta problemática se ha convertido en una prioridad para el conjunto de las universidades del país.

En este trabajo se describe la implementación y el modo de uso de una herramienta software para mostrar información estadística de los datos de los alumnos de nuevo ingreso en la universidad. En la aplicación se emplean modelos predictivos que permiten determinar la relación existente entre que un alumno se encuentre en riesgo de abandonar la titulación y una serie de variables, como pueden ser la nota de acceso a la universidad o la nota media del primer curso.

En este documento se indican los procedimientos, herramientas y configuraciones empleados en la construcción de la aplicación haciendo uso de Django y Python, así como la forma en la que se ha validado y probado el código fuente desarrollado. También se muestra la forma de uso de la aplicación de manera que se permita a los usuarios cargar los datos y realizar tanto una regresión lineal como una regresión logística que permitan comprender qué factores repercuten en el abandono.

A modo de ejemplo de uso de la herramienta se muestran los resultados obtenidos para la cohorte del curso 2017/2018 del Grado en Ingeniería Informática de la Universidad de La Laguna. Para ellos se cuenta tanto con los datos de educación como con los del test psicotécnico de Resolución de Problemas (RP30).

Palabras clave: Abandono, Django, Python, Aplicación Web, PostgreSQL, Regresión Logística, Regresión Lineal, Numpy, Docker

Abstract

College dropout is a major issue for Spanish society. Trying to solve this problem has become a priority to universities all over the country.

This paper describes the implementation and how to use a software tool to show statistical information about incoming college students. In this application, predictive models are used to determine the relationship between a student at risk of dropping out, and several variables such as their admission grades or their average grades during the first year.

This paper shows the procedures, tools, and settings used on the building of the application using Django and Python, as well as the way in which the developed source code has been validated and tested.

It is also shown how to use the application so it allows the user to load data and make a linear regression as well as a logistic regression that allows understanding which factors are related to dropping out.

As an example of how to use the tool, the results obtained for the cohort of the academic year 2017/2018 of Grado en Ingeniería Informática de la Universidad de La Laguna are shown. Relying on education data as well as data from the aptitude test on problem solving: Resolución de Problemas (RP30).

Keywords: College dropout, Django, Python, Web Application, PostgreSQL, Logistic regression, Linear regression, Numpy, Docker

Índice general

1. **Introducción** **1**

1.1. El abandono en los estudios universitarios 1

1.1.1. El abandono en la ESIT 1

1.2. Estado del arte 2

1.2.1. Modelos existentes 2

1.2.2. ASIA 3

1.3. Objetivos 3

1.4. Dificultades encontradas 4

2. **Herramientas** **5**

2.1. Lenguajes de programación para análisis de datos 5

2.1.1. R 5

2.1.2. Python 6

2.1.3. Comparación 6

2.2. Plataformas de desarrollo de software 7

2.2.1. Django 7

2.3. Pipenv 8

2.4. Docker 8

2.5. Tecnologías Web 8

2.5.1. HTML5 8

2.5.2. CSS 9

2.5.3. Bootstrap 9

2.5.4. JavaScript 9

2.6. Base de datos 9

2.6.1. SQLite 10

2.6.2. PostgreSQL 10

2.7. Librerías utilizadas en el análisis de datos . . . 10
2.7.1. NumPy . . . 10
2.7.2. pandas . . . 10
2.7.3. matplotlib . . . 11
2.7.4. Seaborn . . . 11
2.7.5. SciPy . . . 11
2.7.6. Statsmodels . . . 11
2.7.7. Scikit-lear . . . 11
2.8. Despliegue . . . 12
2.8.1. Heroku . . . 12
3. Desarrollo 13
3.1. Requisitos y diseño . . . 13
3.1.1. Funcionalidades . . . 13
3.1.2. Técnicas Estadísticas empleadas y requisitos . . . 14
3.1.3. Diseño de la herramienta . . . 19
3.2. Preparación del proyecto . . . 20
3.2.1. Entorno virtual: Pipenv . . . 20
3.2.2. Configuración de Django . . . 20
3.2.3. Aplicaciones . . . 21
3.2.4. Docker . . . 22
3.3. Desarrollo de la base de datos . . . 22
3.3.1. Análisis de los datos . . . 22
3.3.2. Base de datos: PostgreSQL . . . 25
3.3.3. Modelos . . . 25
3.3.4. Preprocesado de los datos . . . 26
3.3.5. Visualización de los datos . . . 28
3.4. Aplicación: Análisis estadístico . . . 30
3.4.1. Diseño . . . 30
3.4.2. Configuración . . . 30
3.4.3. Funcionamiento . . . 31
3.5. Análisis estadístico: Unidimensional . . . 33
3.5.1. Tabla de frecuencias . . . 33
3.5.2. Gráficos . . . 35

3.5.3. Estadísticos descriptivos . . . 39
3.6. Análisis estadístico: Bidimensional . . . 40
3.6.1. Análisis de datos categóricos . . . 40
3.6.2. Gráficos . . . 41
3.6.3. Modelo: Regresión lineal simple . . . 44
3.6.4. Modelo: Regresión logística . . . 45
4. Verificación y pruebas 46
4.1. Tratamiento de errores . . . 46
4.2. Prueba: Bidimensional . . . 47
4.2.1. Prueba 1: Modelo regresión lineal . . . 47
4.2.2. Prueba 2: Modelo regresión logística . . . 48
5. Conclusiones y líneas futuras 52
5.1. Conclusiones . . . 52
5.2. Mejoras . . . 52
5.2.1. Definir usuarios y roles . . . 53
5.2.2. Aplicar mejoras en el Front-end . . . 53
5.2.3. Mejorar los modelos predictivos . . . 53
5.2.4. Base de datos: Aplicar filtros para mostrar los datos . . . 53
6. Summary and Conclusions 54
6.1. Conclusions . . . 54
6.2. Improvements . . . 54
6.2.1. Defining users and roles . . . 55
6.2.2. The application of improvements in the Front-end . . . 55
6.2.3. Improving the predictive models . . . 55
6.2.4. Database: Applying filters to show data. . . . 55
7. Presupuesto 56
A. Script para generar las variables estadísticas. 57

Índice de Figuras

2.1. Evolución en el uso de las etiquetas de Python y R en las consultas realizadas por los usuarios en *Stack Overflow* desde el año 2008 hasta el 2018 [1]. . . 6

3.1. Diseño del menú principal de la aplicación 20
3.2. Modelos cargados en la base de datos. 26
3.3. Formulario definido para cargar los datos. 27
3.4. Formulario y muestra de datos. 29
3.5. Diseño del panel definido para el análisis unidimensional. 30
3.6. Esquema del funcionamiento de la herramienta. Fuente: Elaboración Propia. 31
3.7. Esquema análisis unidimensional. 33
3.8. Tabla de frecuencia para la variable procedencia(cualitativa). 34
3.9. Tabla de frecuencia para la variable “créditos matriculados en el primer año”(discreta). 34
3.10. Tabla de frecuencia para la variable “Computabilidad y Algoritmia”(continua). 35
3.11. Gráfico de barras para la variable “Edad”. 36
3.12. Gráfico de sectores para la variable “Abandono”. 36
3.13. Polígono de frecuencias para la variable “RP30: Total”. 37
3.14. Histograma para la variable “Álgebra”. 37
3.15. Diagrama de caja y patilla para la variable “Computabilidad y Algoritmia”. . 38
3.16. Diagrama de tallo y hoja para la variable “RP30: Total”. 38
3.17. Tabla con los estadísticos para la variable “Nota media primer año”. 39
3.18. Esquema análisis bidimensional. 40
3.19. Tabla de contingencia para las variables “Sexo” y “Abandono”. 40
3.20. Gráfico de barras adosado para las variables “Nota media del primer año” y “Sexo”. 41
3.21. Gráfico de barras apilado para las variables “Nota media del primer año” y “Sexo”. 41

3.22 Gráfico de mosaicos para las variables “Abandono” y “Sexo”. 42
3.23 Gráfico de caja y patilla para las variables “Estadística” y “Sexo”. 42
3.24 Diagrama de dispersión con ajuste lineal para las variables “Cálculo” y “Nota media del primer año”. 43
3.25 Diagrama de dispersión con ajuste de función sigmoide para las variables “Álgebra” y “Abandono”. 43
3.26 Salida de del método *summary()* sobre el modelo lineal para las variables “Cáculo” y ”Nota media del primer año”. 44
3.27 Salida del método*anova_lm* sobre el modelo lineal para las variables “Cáculo” y ”Nota media del primer año. 45
3.28 Salida del método *summary()* sobre el modelo de regresión logístico para las variables “Nota media del primer año” y ”Abandono”. 45

4.1. Errores en una consulta 46
4.2. Prueba: Modelo de regresión lineal 47
4.3. Diagrama de caja y patilla: Modelo de regresión logística 1 49
4.4. Diagrama de caja y patilla: Modelo de regresión logística 2 49
4.5. Diagrama de dispersión: Modelo de regresión logística 1 50
4.6. Diagrama de dispersión: Modelo de regresión logística 2 50
4.7. Tabla con el análisis: Modelo de regresión logística 1 51
4.8. Tabla con el análisis: Modelo de regresión logística 2 51

Índice de Tablas

3.1. Variables estadísticas utilizadas en el proyecto 24

7.1. Presupuesto para el proyecto 56

Capítulo 1

Introducción

1.1. El abandono en los estudios universitarios

El fenómeno del abandono académico está presente en todas las universidades españolas, siendo una de las problemáticas más importantes a afrontar. Según los datos extraídos del estudio *U-Ranking 2019* [2], elaborado por la Fundación BBVA, uno de cada tres alumnos no finaliza sus estudios universitarios. La gravedad que muestran estos datos es preocupante para el conjunto de la sociedad española, y más concretamente para las instituciones universitarias, que en los últimos años, han comenzado a estudiar este fenómeno y trazar estrategias para combatirlo. Para comprender este problema es necesario primero entender el concepto utilizado para medirlo: 'La tasa de abandono'. Esta se define como el número de estudiantes de nuevo ingreso en una titulación que no están matriculados de la misma después de haber transcurrido dos cursos académicos a su primera matrícula. La información que brinda este indicador es fundamental si se quiere conocer qué tan atractiva está resultando para los alumnos una determinada titulación o cuán motivados están éstos para terminar sus estudios.

1.1.1. El abandono en la ESIT

Los estudios de Ingeniería que se imparten en la Escuela Superior de Ingeniería y Tecnología (ESIT): Electrónica, Informática Mecánica, y Química Industrial; presentan un anormal índice de abandono: entre el 37,4 % y 49,9 %, siendo las carreras con mayores tasas de abandono de la Universidad de La Laguna (ULL) [3]. Por si ésto no fuera lo suficientemente grave, hay que sumarle que, este tipo de carreras tecnológicas, definidas dentro de las carreras llamadas STEM (del inglés, Ciencias, Tecnología, Ingeniería y

Matemáticas), agrupan a menos de la cuarta parte de los alumnos dentro del conjunto de universidades en España [4], estando cinco puntos por debajo de la media europea, además, en los últimos años, según el informe realizado por el Ministerio de Educación y Formación Profesional, el número de matriculados en carreras técnicas ha caído un 28 %. Estos datos tan negativos, donde el fenómeno del abandono juega un papel protagonista, muestra un gran problema de fondo que ha de ser tratado con urgencia.

Otro aspecto a considerar, y que cobra especial relevancia en la titulación de Ingeniería Informática, es la poca presencia de mujeres en los estudios de la ESIT. Según los datos extraídos del informe "Alumnado de nuevo ingreso en estudios de grado y primer y segundo ciclo" de la ULL [3], realizado durante los cursos comprendidos entre 2014 y 2019, las mujeres representan tan solo un 15,6 % del total de alumnos matriculados en este periodo, siendo concretamente un 12,3 % las matriculadas en Informática. Esta dinámica se reproduce a nivel nacional; donde, según los datos publicados por el Ministerio de Educación y Formación Profesional sobre los indicadores universitarios en el año 2018, el porcentaje de mujeres matriculadas en carreras de Informática fue de un 12,1 % en todo el país. Con estos datos, resulta vital comprender cómo es posible que, pese a ser estas las carreras que mayor demanda tienen en el mercado laboral, presenten tan malas cifras en cuanto a alumnos matriculados y a tasas de abandono.

1.2. Estado del arte

1.2.1. Modelos existentes

En la actualidad existen diversos modelos [5] que abordan la problemática del abandono en las titulaciones universitarios y suelen estar basados en el estudio de un conjunto de variables localizadas en distintos episodios de la trayectoria de los alumnos en su etapa universitaria o preuniversitaria, van desde la integración académica a la experiencia formativa previa o la integración social dentro de la universidad. Uno de los modelos con mayor peso en el estudio del abandono, es el propuesto por Vincent Tinto. Este modelo propone una serie de factores que afectan a los alumnos en el desempeño de su actividad formativa. Estos son: las características familiares y personales, las habilidades y capacidades y las experiencias preuniversitarias. También apunta a que el grado de integración académica y social del estudiante repercute en el compromiso que pueda tener de terminar los estudios. Enfocándonos en los estudios de abandono realizados

específicamente sobre titulaciones de Ingeniería Informática, se ha detectado que el mayor porcentaje de abandono se produce en el primer año académico, que las tasas de abandono son superiores a las de otras titulaciones de Ingeniería y que el principal motivo suele estar relacionado con dificultades de adaptación a la universidad. Debido a ésto, la mayoría de modelos definidos para estudiar el abandono en Ingeniería Informática trazan un perfil del alumno que abandona sus estudios [6] muy similar, donde se destacan las siguientes características: No haber elegido la titulación como primera opción, no tener conocimientos previos sobre la titulación, tener una nota de acceso relativamente baja, y un grado de integración en la universidad bajo.

Por último cabe destacar que se ha empezado a extender el uso de técnicas de aprendizaje automático (*machine learning*) y minería de datos (*data mining*) dentro del estudio del abandono universitario, no tanto como herramientas para medir los factores y variables que afectan directamente al fenómeno, sino como medios preventivos para localizar a alumnos más propensos al abandono.

1.2.2. ASIA

En cuanto a las herramientas utilizadas para analizar este fenómeno, actualmente, y dentro de España, destaca ASIA [7], una aplicación en línea para el seguimiento institucional del abandono en un determinado año, que ofrece información numérica y gráfica sobre la permanencia y el abandono de estudiantes de nuevo ingreso en la Universidad Politécnica de Madrid. Esta aplicación se basa en el uso de las variables típicas para este tipo de estudios, donde se pretende conocer la tipología de los estudiantes que abandonan en función de su género, el tipo y la opción de ingreso, la nota de acceso, el rendimiento académico o la edad.

1.3. Objetivos

El objetivo principal de este trabajo fin de grado es el diseño y desarrollo del prototipo de una aplicación web para analizar datos de abandono universitario.

En concreto se persigue mostrar la información estadística que permita predecir qué alumnos se encuentran en mayor riesgo de abandonar sus estudios en función de determinadas variables como pueden ser, la nota de acceso a la universidad o la nota media del primer curso.

Además, para la validación de la aplicación se utilizaran datos del Grado en Ingeniería Informática y como novedad se contrastará si tiene alguna influencia en el abandono las puntuaciones de un test psicotécnico denominado RP30 realizado sobre los alumnos de nuevo ingreso en el grado.

1.4. Dificultades encontradas

Durante la realización del proyecto se han encontrado múltiples dificultades de distinta índole. Se definen los procesos que más problemas han entrañado.

- Obtener los datos con los que realizar los distintos análisis fue un proceso largo, donde hubo que contemplar primero las variables que se iban a utilizar en el proyecto, y después la forma de obtenerlas.
- El proceso consistente en trasladar los gráficos generados dinámicamente, en función de la petición del usuario, a la plantilla donde se muestra, también ha supuesto una dificultad importante, teniendo que guardar la imagen con los gráficos en un *buffer* con la codificación adecuada para su correcta visualización en la plantilla.
- Trazar e interpretar un modelo de regresión logística no ha sido sencillo; ha supuesto un proceso de documentación intenso donde se ha debido adquirir una importante cantidad de nuevos conceptos estadísticos en un corto periodo de tiempo.

Capítulo 2

Herramientas

En este capítulo se recogen y definen las herramientas empleadas en el proyecto, se comentan sus características, se explican las funciones que desempeñan en la aplicación y los motivos que justifican su elección.

2.1. Lenguajes de programación para análisis de datos

La elección de un lenguaje de programación que permita gestionar datos y producir información estadística fue el primer paso a tomar en el desarrollo del proyecto. Para que la herramienta cumpliese con su cometido, era necesario que el lenguaje tuviese una base estadística extensa; debía tener librerías para la generación de estadísticos, la representación gráfica de los datos y la elaboración de modelos predictivos. Existen múltiples lenguajes de programación que cumplan con estás características: Scala, R, MATLAB, Python, Julia... No obstante se ha seleccionado R [8] y Python como únicos candidatos debido a que son, dentro de los lenguajes utilizados en la ciencia de datos, los más populares. También cabe destacar que existe cierta familiaridad en su uso puesto que se ha trabajado con ellos con anterioridad a diferencia que con el resto.

2.1.1. R

R puede ser entendido no sólo como un lenguaje de programación enfocado al análisis estadístico, sino también como un entorno de software libre que integra herramientas para el manejo de datos, la realización de cálculos estadísticos y de gráficos. Es un lenguaje desarrollado por estadísticos que destaca por su extenso catálogo de paquetes y librerías de código abierto y por permitir visualizar los datos con gran calidad. R forma

parte del sistema GNU y su distribución se realiza mediante licencia GNU GLP.

2.1.2. Python

Python es un lenguaje de programación de propósito general, multiparadigma, multiplataforma e interpretado, que cuenta, al igual que R, con un extenso abanico de librerías dedicadas a la ciencia de datos. Se caracteriza por su facilidad de uso y por su sintaxis sencilla y fácil de entender.

2.1.3. Comparación

Con el auge de disciplinas tales como la Inteligencia Artificial, el Aprendizaje Automático (*Machine Learning*) o el Análisis de Datos masivo (*Big Data*), la popularidad de ambos lenguajes se ha incrementado en los últimos años:

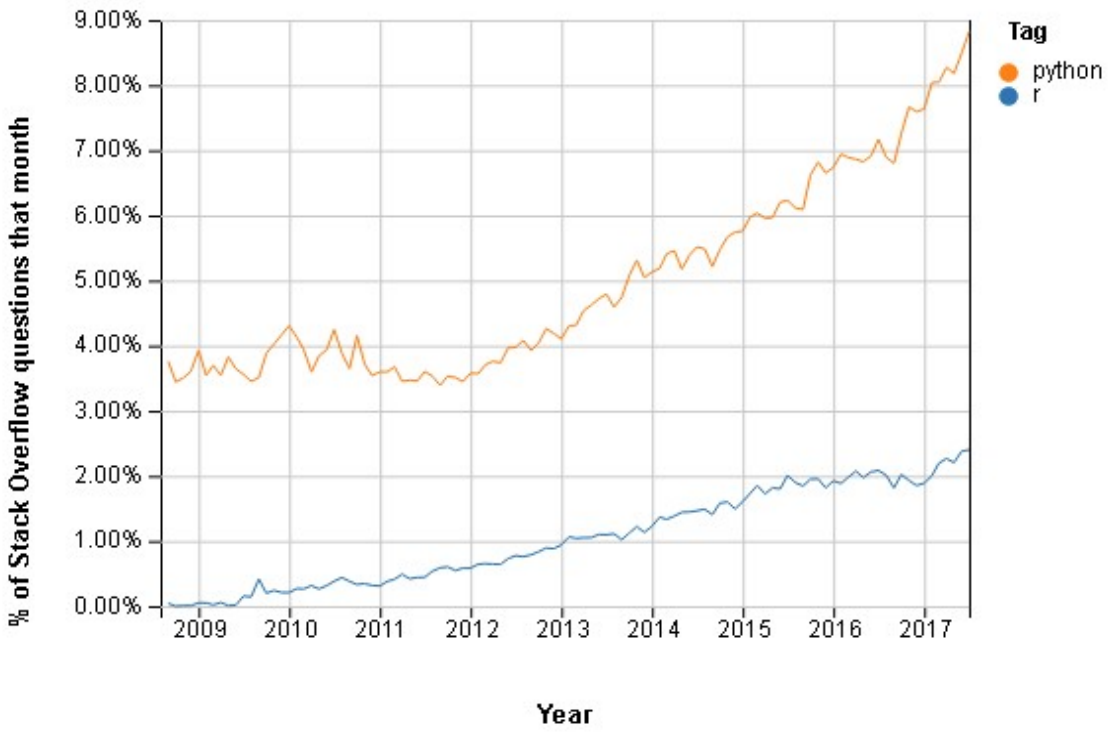


Figura 2.1: Evolución en el uso de las etiquetas de Python y R en las consultas realizadas por los usuarios en *Stack Overflow* desde el año 2008 hasta el 2018 [1].

Para la elección del lenguaje se han tenido en cuenta una serie de consideraciones:

- R está orientado al análisis estadístico, por lo que cuenta con librerías más completas para el tratamiento de datos que Python. Su desarrollo más específico hace que sea superior en este aspecto. Sin embargo, para el trabajo estadístico que abarca

este proyecto, Python cuenta con recursos de sobra, por lo que este criterio no es esencial a la hora de tomar la decisión.

- La herramienta es una aplicación web, y si bien existe una librería en R llamada *"Shiny"* para crear este tipo de aplicaciones, el objetivo que persigue el proyecto es definir una aplicación robusta, que cuente con su propia base de datos, que sea fuertemente configurable y adaptable, y que no se limite únicamente a mostrar la información estadística. Por lo que Python, al ser un lenguaje utilizado en la web que cuenta con sus propios *frameworks*, se ve claro vencedor en este apartado.

- Por último, mencionar que Python es un lenguaje con el que se ha trabajado muy frecuentemente, y en comparación con R, en mucha mayor medida. Por lo que la soltura adquirida programando en este lenguaje es fundamental a la hora de tomar la decisión.

Vistos estos aspectos, Python ha sido el lenguaje que más beneficios aporta y por tanto el elegido para desarrollar la aplicación.

2.2. Plataformas de desarrollo de software

Una vez definido el lenguaje de programación con el que se iba a trabajar, fue momento de seleccionar la plataforma de desarrollo (*framework*) con la que construir la aplicación web. Como el lenguaje seleccionado fue Python, trabajar con un *framework* escrito en el mismo lenguaje supone un ahorro muy significativo en tiempo y complejidad.

Actualmente, el *framework* más popular escrito en Python es Django [9].

2.2.1. Django

Django es un *framework* de desarrollo web de alto nivel y de código abierto, escrito en Python. Django hereda de Python la filosofía de *"pilas incluidas"*(*"batteries-included"*), que consiste en proveer a los usuarios los servicios más típicos en el desarrollo web: autentificación de usuarios, plantillas, vistas, interfaz de administración, soporte para multiples *backends* de base de datos, etc.. evitando que éstos tengan que ser configurados de forma separada.

El principal motivo que justifica la elección de este *framework* es la escalabilidad que ofrece. Django permite pasar de un prototipo de proyecto web sencillo con pocas

funcionalidades a una aplicación web completa, mediante el uso de *aplicaciones* o módulos, independientes entre ellos, que van añadiendo nuevas características mientras se mantiene la estabilidad y seguridad del proyecto.

2.3. Pipenv

Un entorno virtual en Python es una herramienta que permite crear un entorno aislado para un proyecto. Posibilita que cada proyecto pueda tener sus propias dependencias y que éstas sean independientes de las de otros. En este proyecto se utiliza Pipenv; se encarga de crear un fichero *Pipfile* que contiene las dependencias de software de la aplicación y un fichero *Pipfile.lock* encargado de producir siempre la misma configuración del proyecto en cualquier entorno.

2.4. Docker

Docker es una herramienta de código abierto que se encarga de automatizar la implementación de aplicaciones mediante el uso de contenedores portátiles y autosuficientes que incluyen todo lo necesario para que el software se ejecute, como bibliotecas, código o herramientas de sistema. Una de las principales diferencia entre Docker y un entorno virtual reside en que este último solo es capaz de aislar paquetes de Python, no puede aislar otro software empleado en el proyecto como la base de datos. También se ha de tener en cuenta que el entorno virtual apunta a una versión de Python ya existente; no contiene Python por sí mismo. De esta forma, se instaló Python dentro de Docker junto con la base de datos que empleaba el proyecto web y las dependencias de Python mediante los ficheros *Pipfile* y *Pipfile.lock* generados por Pipenv.

2.5. Tecnologías Web

2.5.1. HTML5

HTML5 es la quinta revisión del lenguaje básico de la World Wide Web, HTML, utilizado para definir la estructura y el contenido de una página Web. En la aplicación se combina este estándar web con las variables y etiquetas que proporciona Django para dar formato al contenido de las plantillas.

2.5.2. CSS

CSS es el lenguaje de diseño gráfico empleado para definir y crear la presentación y el estilo de un documento HTML. En el proyecto, CSS se utiliza sobretodo para dar formato a los menús desplegables y a algunos otros elementos como botones, tipografías y etiquetas. Sin embargo la mayor parte del diseño recae sobre Bootstrap [10].

2.5.3. Bootstrap

Bootstrap es un *framework* de CSS enfocado en el desarrollo front-end. Contiene un conjunto de elementos tales como: plantillas de diseño con tipografía, menús de navegación, formularios, botones y otros componentes de diseño que están basados en HTML y CSS, así como algunas extensiones de JavaScript. La elección de un *framework* de estas características simplifica y agiliza el desarrollo de la parte *front-end* de la herramienta, permitiendo que la aplicación final tenga una apariencia cuidada y atractiva, con un diseño dinámico, sin tener que invertir excesivo tiempo en un aspecto que, a priori, no se contempla como prioritario en el proyecto.

2.5.4. JavaScript

Es un lenguaje de programación interpretado, orientado a objetos, basado en prototipos, imperativo, dinámico y con tipado débil. Es el lenguaje de programación definido para la web, principalmente del lado del cliente, aunque también para el lado del servidor. Permite diseñar webs dinámicas. El uso de JavaScript en la aplicación web se centra en mostrar y ocultar las opciones de los distintos submenús cuando el usuario marque la casilla correspondiente en los formularios.

2.6. Base de datos

Durante el desarrollo de la aplicación se ha trabajado con dos bases de datos; en una fase temprana del proyecto se utilizó SQLite [11], pero a medida que la aplicación iba creciendo en complejidad se hizo necesario emplear PostgreSQL [12] como base de datos más robusta.

2.6.1. SQLite

SQLite es una biblioteca en lenguaje C que implementa un motor de base de datos relacional pequeño, rápido y basado en ficheros. Destaca por su fiabilidad y su facilidad de uso. Viene integrado por defecto en Django. Para un proyecto local o que no maneje una gran cantidad de datos, puede resultar ideal, sin embargo, para una aplicación como la que se plantea, donde se pretende almacenar y manejar datos del alumnado de nuevo ingreso de distintos cursos y titulaciones, se ha considerado conveniente utilizar una base de datos más potente: PostgreSQL.

2.6.2. PostgreSQL

PostgreSQL es un sistema de base de datos relacional de objetos de código abierto, multiplataforma, escalable y que cuenta con soporte para gran cantidad de lenguajes de programación, entre ellos Python. Es la opción más popular para los desarrolladores de Django, por lo que permite encontrar abundante información sobre su implementación y su uso.

2.7. Librerías utilizadas en el análisis de datos

2.7.1. NumPy

NumPy (*Numerical Python*) [13] es la extensión por excelencia dentro del campo de la computación númerica en Python. Proporcionó toda las estructuras de datos, algoritmos y funciones para todo el tratamiento de los datos dentro de la herramienta.

2.7.2. pandas

Pandas [14] es una biblioteca de software basada en NumPy para la manipulación y análisis de datos. Proporciona estructuras de datos de alto nivel y funciones diseñadas para el trabajo con datos estructurados. La estructura de datos más utilizada en la ciencia de datos está incluida dentro de esta librería y se denomina *DataFrame*.

El *DataFrame* es la estructura de datos utilizada en el tratamiento de datos de la parte de análisis bidimensional de la aplicación. Es un objeto en forma de tabla, similar a una matriz, con columnas con los datos (que pueden ser de distintos tipos) y que contienen etiquetas por filas y columnas.

2.7.3. matplotlib

matplotlib [15] es la biblioteca más popular en Python para la generación de gráficos y otras visualizaciones de datos. Es el módulo principal utilizado en la aplicación para la elaboración de los gráficos, tanto en la parte unidimensional como en la bidimensional.

2.7.4. Seaborn

Seaborn [16] es una librería de Python para producir gráficos. Está basada en matplotlib. En el proyecto se utiliza conjuntamente con matplotlib para la parte de representación gráfica, proporcionando los métodos para generar aquellos gráficos de los que esta última no dispone.

2.7.5. SciPy

SciPy [17] es una biblioteca compuesta por herramientas, funciones y algoritmos matemáticos. Está basada en NumPy y proporciona métodos para generar diversos procedimientos estadísticos en la aplicación.

2.7.6. Statsmodels

Statsmodels [18] Es un paquete de análisis estadísticos que implementan modelos de análisis de regresión. Es la biblioteca empleada para elaborar los modelos predictivos de regresión lineal y logística.

2.7.7. Scikit-lear

Scikit-learn [19] es una biblioteca para aprendizaje automático de software libre que se ha convertido en la herramienta más importante de *machine learning* en Python. Pese a ello, el uso que se hace de ella en la aplicación está únicamente centrado en generar algún ajuste y en la comparación de los resultados con los obtenidos mediante los modelos de statsmodels.

2.8. Despliegue

2.8.1. Heroku

Para el despliegue de la aplicación web en la nube se ha seleccionado Heroku [20]. Heroku es una herramienta *PaaS* (*Platform-as-a-Service*) que tiene soporte para Django y PostgreSQL y cuenta con un plan de servicios gratuito.

Capítulo 3

Desarrollo

En este capítulo se explicarán las distintas fases que se han ido recorriendo en la construcción de la aplicación web. Abarca desde el planteamiento inicial, donde se precisan las funciones que debe tener, hasta el desarrollo de cada uno de los apartados que conforman la herramienta.

La aplicación web puede ser accedida desde el siguiente enlace:

`https://warm-stream-04750.herokuapp.com/`

3.1. Requisitos y diseño

La aplicación web se plantea como una herramienta estadística, donde, a partir de los datos de los alumnos de nuevo ingreso en una titulación, se trate de obtener información que permita comprender cuáles son las variables que más influyen en el abandono de la titulación. A partir de esta premisa se pueden definir los requisitos mínimos que ha de tener una aplicación web de estas características: almacenamiento, tratamiento y gestión de datos, y utilización de modelos y técnicas estadísticas para generar la información.

En cuanto al diseño, se ha pretendido que la aplicación web sea lo más intuitiva posible, aspirando a que su uso y funcionamiento quede claro desde el primer momento. Se trata de mostrar las opciones principales directamente y evitar implementar funcionalidades que puedan desvirtuar el propósito que persigue el proyecto.

3.1.1. Funcionalidades

La aplicación web presenta tres funcionalidades que garantizan el correcto funcionamiento de la misma:

1. Carga de datos: La carga de datos en la web se hará a través de un formulario, donde el usuario solo tiene que cargar los ficheros correspondientes. La propia aplicación web se encarga de procesar la información y generar las variables con las que se trabajará en el apartado de *Herramienta*. De esta forma se permite añadir grandes volúmenes de datos en muy poco tiempo, evitando que se tenga que manipular manualmente la base de datos y permitiendo que se puedan generar nuevas variables a partir de la información que ofrecen otras, sin necesidad de crearlas previamente.

2. Visualización de datos: Es la parte encargada de mostrar al usuario los datos almacenados en la base de datos. El objetivo es mostrar la información de los alumnos, de una determinada titulación y curso, de forma similar a como se vería en una hoja de cálculo, donde cada columna se corresponde con una variable y cada fila con un estudiante de la muestra, salvo en la primera fila que recoge los nombres de las variables.

3. Análisis estadístico de los datos: como su propio nombre indica, está funcionalidad permite tomar los datos almacenados en la aplicación web y generar información estadística con ellos. Es la funcionalidad que justifica el interés del proyecto y la que nos va a permitir, por ejemplo, determinar si una variable influye en el abandono del alumnado en la carrera de Ingeniería Informática.

3.1.2. Técnicas Estadísticas empleadas y requisitos

Las técnicas empleadas en el proyecto son, por un lado, el análisis exploratorio de los datos, utilizada para el análisis estadístico unidimensional y por otro, el análisis de datos categóricos, gráficos y los modelos predictivos de regresión lineal y logística, para la parte de análisis bidimensional.

Análisis Unidimensional: Análisis exploratorio

El análisis exploratorio de datos trata de organizar y analizar la información que brindan los datos. Consiste en generar tablas de frecuencias, gráficos y medidas descriptivas que permiten explorar los datos identificando características tales como: valores atípicos, concentraciones de valores o la forma de la distribución. Se suelen emplear las siguientes técnicas [21]:

- *Tablas de frecuencias*: Muestran cómo se distribuyen las frecuencias en función de los valores que tome la variable. La tabla a mostrar ha de ser específica para cada tipo de variable.
- *Gráficos*: Se visualiza gráficamente la distribución de frecuencias de las variables. Al igual que en el punto anterior, se ha de considerar la variable empleada para mostrar unos gráficos u otros.
- *Medidas descriptivas*: Calcula una serie de medidas que aportan información global del conjunto de datos, y que se pueden clasificar en medidas de centralización o posición, de dispersión o variabilidad, y de forma.

Análisis Bidimensional

- *Análisis de datos categóricos*:
  - Tablas de frecuencias y/o contingencia: Tablas de doble entrada que muestran la distribución de frecuencias para los diferentes pares de categorías o clases de dos variables.
  - Prueba de independencia: Test de la Chi-cuadrado para comprobar si existe relación o son independientes dos variables cualitativas nominales.
  - Medidas de asociación: Para expresar el grado de relación existente entre dos variables cualitativas, eliminando el efecto del tamaño muestral.
- *Gráficos*: Representación de algunos gráficos que permiten comprobar la distribución de frecuencias de una variable entre varios grupos limitados por las categorías de otra, como los diagramas de barras adosados y apilados, o los diagramas de caja y patilla, y otros que permiten determinar la relación existente entre las variables analizadas como el gráfico de regresión, superponiendo la línea de regresión lineal o la curva de regresión logística.
- *Regresión lineal simple*: La regresión lineal es un modelo matemático que trata de determinar la relación lineal existente entre una variable dependiente o regresora, X y una variable continua Y, llamada dependiente o respuesta. El objetivo del modelo es predecir el valor de la variable respuesta en función del valor que toma la variable independiente.

  Las hipótesis que se deben cumplir para poder aplicar la regresión son las siguientes:

1. Linealidad entre la variable respuesta Y y la variable regresora X.
2. Normalidad de las observaciones Y, dado cada valor de la variable regresora X.
3. Igualdad de varianzas de variable dependiente Y para cada valor de la variable X.
4. Independencia de las observaciones de Y.

El modelo usado en la regresión lineal es el siguiente:

$$Y = \beta_1 X + \beta_0 + \epsilon$$

Esta función nos dice que, dado un valor de X, el valor observado de Y viene determinado por el valor predicho por el modelo más un cierto error, donde:

- El parámetro β_0 indica la ordenada en el origen.
- El parámetro β_1 determina la pendiente estimada del modelo lineal. Si $\beta_1 = 0$, no existe relacional lineal entre las variables. Si por el contrario $\beta_1 \neq 0$ se indica que si existe relación lineal.
- Por último el parámetro ϵ representa el error o residuo.

Para estimar los parámetros β_0 y β_1 se emplea el método de los mínimos cuadrados, consistente en minimizar la suma de todos los errores o residuos al cuadrado. De esta forma se obtiene:

$$\beta_1 = \frac{\sigma_{xy}}{{\sigma_x}^2} \quad y \quad \beta_0 = \overline{y} - \beta_1 \overline{x}$$

Para medir la bondad del ajuste del modelo lineal a la nube de puntos, en la aplicación se han implementado las siguientes técnicas:

- *Medidas de asociación*:
  - Coeficiente de correlación: Mide la asosiación lineal entre las dos variables, toma valores entre -1 y 1, donde una dependencia fuerte se traduce en valores cercanos a 1 y -1 y la ausencia de dependencia con valores próximos a 0.

- Coeficiente de determinación: Se obtiene elevando al cuadrado el coeficiente de correlación, es más robusto que éste y puede tomar valores comprendidos entre 0 y 1. Se puede entender como la proporción de variabilidad de Y que es explicada por X, donde un valor cercano a 1 implica una mayor proporción de variabilidad de la variable respuesta Y que es explicada por el modelo lineal de regresión.

- *Análisis de la varianza (ANOVA)*: Se contrasta la hipótesis nula de que no existe relación lineal entre las dos variables frente a que si la hay. Dicho contraste se resuelve basándose en la descomposición de la varianza de la variable respuesta Y (suma de cuadrados totales) en la varianza residual (suma de cuadrados de los residuos) y la varianza de la regresión (suma de cuadrados de la regresión). Dichas variables son Chi-cuadrados con $n-1$, $n-2$ y 1 grados de libertad, respectivamente. El cociente de cada suma de cuadrados, (de la regresión y de los residuos) partidos por sus grados de libertad, son estimadores de la varianza de Y, (llamados errores cuadráticos medios de la regresión y del residual), que se comparan, mediante el cociente de ambos que es el estadístico, la F de *Fisher-Snedecor*. En función del p-valor que tiene asociado se comprueba si se rechaza o no. Si el p-valor es bajo ($< 0{,}05$) se rechaza la hipótesis nula demostrando que existe una relación lineal entre X e Y, si por el contrario el p-valor es superior, la hipótesis se acepta.

- *Contrastes sobre β_1 y β_0*: Contrasta la hipótesis para $H_0 : \beta_1 = 0,\ \ H_1 : \beta_1 \neq 0$ y $H_0 : \beta_1 = 0,\ \ H_1 : \beta_0 \neq 0$. Los estadísticos que resuelven ambos contrastes se distribuyen como *t-student*, y se definen como la estimación de cada parámetro dividido por su correspondiente error estándar. Si se acepta la hipótesis nula de que $\beta_1 = 0$, se puede concluir que el modelo no es adecuado para representar la nube de puntos.

Para completar el modelo, se ha definido un gráfico de dispersión que permite, mediante un ajuste lineal, determinar la relación existente entre la variable Y y la variable X.

Por último, para la predicción de un determinado valor basta con sustituir en el modelo el valor concreto de X que se le pase y se obtendrá el valor predicho para Y.

- *Regresión logística simple* [22]: Es un tipo de análisis de regresión utilizado para

determinar la relación existente entre una variable cualitativa y dicotómica Y (variable dependiente) con una variable cuantitativa independiente X. El objetivo de este modelo es predecir la probabilidad de que ocurra Y en función de los valores de X. La variable dependiente dicotómica ha de tomar los valores 0 y 1, por lo que previamente a la creación del modelo se debe definir una variable ficticia, conocida como *variable dummy*, que traduzca los valores originales. La probabilidad de que ocurra el suceso considerado como éxito se define como $p = P(Y = 1)$, siendo $1 - p = P(Y = 0)$ la probabilidad del suceso contrario. Entonces se define $odds$ como sigue:

$$odds = \frac{p}{1-p}$$

El modelo de regresión logística expresado en función de los $odds$ sería:

$$\frac{p}{1-p} = e^{\beta_1 X + \beta_0}$$

Y expresado en términos de probabilidad:

$$p = \frac{1}{1 + e^{-(\beta_1 X + \beta_0)}} = \frac{e^{\beta_1 X + \beta_0}}{1 + e^{\beta_1 X + \beta_0}} \tag{3.1}$$

La nube de puntos no se podría aproximar al modelo lineal $Y = \beta_1 X + \beta_0$, ya que al tomar la X valores extremos la Y tomaría valores inferiores que 0 y superiores a 1. Para evitar esto, la regresión logística transforma el valor devuelto por la regresión lineal empleando la función logística o sigmoide: $\frac{1}{1+e^{-x}}$. De esta forma, para valores grandes de X, e^{-x} vale 0 y la función sigmoide 1, y para valores pequeños de X, e^{-x} tiende a infinito y la función sigmoide vale 0.

Para estimar los parámetros β_1 y β_0 en regresión logística se utiliza el método de máxima verosimilitud, que busca los coeficientes que maximizan la probabilidad de los datos observados.

Para medir la bondad del ajuste del modelo de regresión logística, se hace uso de las siguientes técnicas:

- Contrastes sobre β_1 y β_0: Para comprobar si el modelo es significativo, esto es, para contrastar si los parámetros β_1 y β_0 son nulos o no, se usa como estadístico el cociente de las estimaciones de los parámetros y sus errores estándar que son variables Normales estándar. Sus cuadrados son los denominados estadísticos de Wald (chi-cuadrados). Si se concluye que β_1 es distinto de 0, se puede asumir

que el modelo es significativo y por tanto, la probabilidad de que la variable respuesta Y tome el valor 1, dependerá de los valores que tome la variable independiente X.

- Test de Verosimilitud: Se utiliza la medida *Likelihood ratio* (razón de verosimilitud) basada en -2 veces el logaritmo de la razón o cociente entre la verosimilitud de los datos observados con el modelo completo (con variable predictora) y la verosimilitud del modelo nulo (sin variables predictoras) [23]. Un valor elevado de este estadístico, o de forma equivalente, un p-valor próximo a 0 indican un buen ajuste del modelo.

- Coeficiente pseudo R^2 de McFadden: Este coeficiente intenta ser un equivalente al de determinación, R^2, del modelo de regresión lineal, puesto que éste no existe dentro de los modelos de regresión logística. Los valores que toma oscilan entre 0 y 1 donde, cuando más próximo a 1 sea, mejor será el ajuste. Su fórmula viene dada por:

$$R^2_{McF} = 1 - \frac{\ln \hat{L}(modelo)}{\ln \hat{L}(modelo\ \ nulo)}$$

Una vez se han estimado los coeficientes del modelo, es posible hallar la probabilidad de que la variable dependiente Y pertenezca a una de las categorías en función de un valor dado de la variable X, véase la ecuación 3.1.

3.1.3. Diseño de la herramienta

Para el diseño se ha decidido utilizar una interfaz simple, donde todas las funcionalidades aparecen en el menú superior de la web. La primera opción del menú es el "Inicio", que actúa como *home*, contiene el nombre del proyecto y el objetivo que persigue el mismo. La segunda opción es denominada "Subir Datos", y, tal como su nombre indica, es desde donde se cargan los datos a la base de datos. Luego está la opción "Ver Datos", opción definida para recuperar la información de la base de datos y ser mostrada a los usuarios. Por último, se encuentra la opción "Herramienta", esta despliega un pequeño menú cuando es pulsada con los análisis estadísticos implementados (unidimensional y bidimensional).

El diseño de cada una de las plantillas que conforman la herramienta web será mostrado y detallado en las secciones correspondientes que tratan su desarrollo.

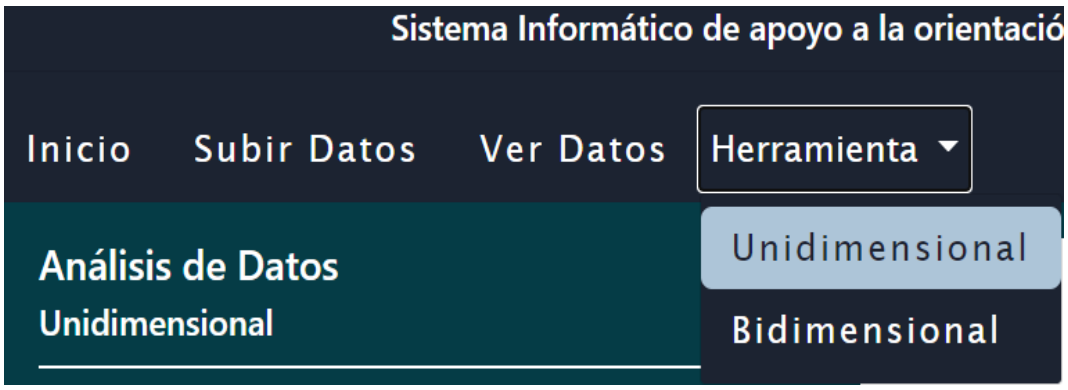


Figura 3.1: Diseño del menú principal de la aplicación

3.2. Preparación del proyecto

3.2.1. Entorno virtual: Pipenv

El primer paso, previo a la construcción de la aplicación web, fue establecer el entorno virtual en el que se iba a desarrollar el proyecto. Con Pipenv aislamos las dependencias que se han ido instalando a lo largo del desarrollo de la herramienta y generamos los ficheros *Pipfile* y *Pipfile.lock* para declarar todas las requisitos software de Python utilizados y poder crear *builds* de la aplicación web deterministas.

3.2.2. Configuración de Django

Una vez que se configuró el entorno virtual, se procedió a instalar y configurar Django. Este proceso se realizó siguiendo una serie de pasos:

1. Se instaló la versión de Django mediante Pipenv. La versión utilizada es la 3.0.8.
2. Se creó el *superusuario* que permite acceder a la interfaz de administración. Desde la interfaz de administración se podrá interactuar con los datos almacenados de forma visual, controlando que se estén almacenando adecuadamente.
3. Se definió e instalaron las aplicaciones que añadirán las funcionalidades necesarias para la herramienta.
4. Se crearon las plantillas (ficheros html) donde se van a mostrar los contenidos de la aplicación web.
5. Se crearon los ficheros *Dockerfile* y *docker-compose.yml*.
6. Se instaló la base de datos PostgreSQL.

7. Se definieron los modelos para almacenar los datos.

8. Finalmente y tras crear los modelos, se realizó la migración de los mismos para que se añadiesen a la base de datos.

Cuando ya realizaron estás configuraciones, la base del proyecto quedó edificada, y se pudo comenzar a trabajar en cada una de las aplicaciones.

3.2.3. Aplicaciones

Una aplicación[1] o app en Django es un paquete que representa una parte del proyecto que tiene asignada una función determinada dentro del mismo. Normalmente, un proyecto Django está constituido por múltiples aplicaciones (para la gestión de usuarios, sistema de comentarios, etc.) que se construyen individualmente pero se relacionan entre sí.

En este proyecto web se definen principalmente dos aplicaciones:

- Alumnos. Se encarga de toda la parte de gestión de datos. contiene los modelos para la base de datos, el formulario para subir los datos a la aplicación web, el formulario para mostrar los datos y la vista que maneja toda la lógica de esta aplicación.

- Herramienta. Es la aplicación definida para obtener los procedimientos estadísticos: Contiene los formularios mediante los cuales los usuarios seleccionan la información estadística que desean generar y las vistas que satisfacen estas peticiones.

[1] En este proyecto cada vez que se utilice el término *aplicación* estará referido a la definición que tiene dentro de Django. Para referenciar el conjunto del proyecto, se utilizan los términos *aplicación web*, *proyecto* o *proyecto web* y *herramienta*.

3.2.4. Docker

Para crear el entorno de desarrollo del proyecto, se tuvo que crear y configurar los ficheros *Dockerfile* y *docker.compose.yml*. El fichero *Dockerfile* contiene una lista con las instrucciones que permiten crear la imagen. Dentro de este fichero se define la imagen base (Python 3.8) que Docker utiliza para construir la imagen del proyecto. Se define también el directorio de trabajo, donde se guarda el código de la aplicación web, se precisan las variables de entorno y se instalan las dependencias con Pipenv.

Una vez creada la imagen, utilizamos *docker-compose.yml* para controlar cómo ejecutar los contenedores que se construirán en función de la imagen de Docker previamente creada. Los servicios (o contenedores) que vamos a tener corriendo son la base de datos y el servicio web.

3.3. Desarrollo de la base de datos

3.3.1. Análisis de los datos

Como ya se ha mencionado repetidamente en apartados anteriores, en el proyecto se trabaja con información de alumnos de nuevo ingreso. En concreto, el análisis y las pruebas se realizan sobre los datos de los alumnos matriculados en el grado de Ingeniería Informática de la Universidad de La Laguna en el curso 2017/2018, proporcionados por el Gabinete de Planificación de la ULL.

Archivos con los datos

Se trabaja con tres ficheros de datos distintos:

1. Fichero con información de los alumnos de nuevo ingreso en la titulación: Contiene información genérica, de procedencia, de acceso y académica de los dos primeros cursos, de los alumnos de nuevo ingreso en la titulación. Este fichero contiene varias filas con información para cada alumno. Los datos que recoge cada fila son: el curso, el plan, el DNI, el sexo, la edad al ingresar a la titulación, la isla de procedencia, la ocupación, si es becario, la convocatoria de prescripción, la preferencia, el cupo, la modalidad, la especialidad, la nota de bachiller/ciclo, la nota de la prueba de acceso, la nota de admisión, el curso del acta, la asignatura, el grupo de acta, la convocatoria del acta, la calificación y la calificación numérica.

Es el fichero que más datos recoge y del que se extrae la mayoría de variables que emplea la aplicación web desarrollada.

2. Fichero con los resultados del RP30: El RP30 es un test psicotécnico que mide un conjunto de aptitudes, como el razonamiento, la memoria de trabajo o la actitud espacial. Está compuesto por 30 problemas con diferentes grados de dificultad.

   Una de las principales motivaciones del proyecto fue determinar qué grado de relación existía entre los resultados obtenidos en esta prueba con que los estudiantes abandonen o no la titulación.

   Este fichero se obtiene toda la información alusiva a los resultados del test. Los campos son: DNI, nombre, apellidos, aciertos, fallos, total, edad, nota de bachiller y nota de la Ebau.

3. Fichero con las asignaturas matriculadas. Los campos que recoge son: Curso, plan, DNI, si el alumno está matriculado del curso siguiente (2018-19), si el alumno está matriculado dos cursos después (2019-20), las asignaturas de las que está matriculado y los grupos y cursos de cada asignatura.

   La mayoría de campos son redundantes a los del primer fichero, no obstante, los campos que indican si el alumno está matriculado de los dos cursos posteriores nos permiten conocer si el alumno abandona o continua con sus estudios.

Definición de las variables

Estas son las variables que se utilizan en la aplicación web, clasificadas por su tipo:

Variables			
Cualitativas		Discretas	Continuas
Nominales	Dicotómicas		
Procedencia	Sexo	Edad	Nota de admisión
Modalidad	Trabajo	Preferencia	Media 1º año
Especialidad	Beca	C. matriculados	Media 2º año
Convocatoria	Abandono	C. presentados	Nota Asignaturas
		C. aprobados	
		RP30: Aciertos	
		RP30: Fallos	
		RP30: Total	

Tabla 3.1: Variables estadísticas utilizadas en el proyecto

Es importe realizar una serie de consideraciones respecto al contenido de la tabla:

- Las variables definidas en la tabla que comienzan con “C." hacen referencia a los créditos del alumno, y están recogidas para los dos años, es decir, existe una variable “créditos matriculados" para el primer curso y otra para el segundo curso.

- Dentro del apartado de las variables continuas, “Nota Asignaturas" engloba todas las calificaciones obtenidas por los alumnos para cada asignatura, tomando siempre la última calificación obtenida. Aunque se muestre como una única variable en la tabla, esto es solo para facilitar la visualización, en la base de datos se define un campo para cada asignatura.

  Las asignaturas que se toman son las de los dos primeros años de la titulación, puesto que, aunque puedan aparecer alumnos matriculados en asignaturas de cursos posteriores, son casos muy excepcionales, dando como resultado variables que contienen un número de observaciones demasiado pequeño como para ser una variable significativa para el análisis.

- Las variables obtenidas del fichero del RP30 (aciertos, fallos y el total) y la variable *Edad* presentan un campo de variabilidad muy amplio, por lo que se les da el

tratamiento de una variable continua.

3.3.2. Base de datos: PostgreSQL

Para configurar PostgreSQL dentro del proyecto Django se requirieron de cuatro pasos fundamentales:

1. Para la instalación se puede optar por dos aproximaciones; la primera consiste en instalar PostgreSQL localmente, y la otra, que es la que se ha implementado, es obtener una imagen Docker de PostgreSQL desde el fichero *docker-compose.yml* definiendo un nuevo servicio para crear el contenedor con la configuración de la base de datos.

2. Luego se instaló un adaptador para base de datos que permite a Python comunicarse con PostgreSQL: *psycog2*.

3. En los ajustes de Django, se cambió el motor de base de datos, que por defecto viene configurado con *sqlite3*. En el fichero de ajustes del proyecto se sustituyó la configuración previa y se añadieron los siguientes campos; ENGINE, NAME, USER, PASSWORD, HOST y PORT con la nueva configuración para postgreSQL. En el primer campo se incorporó el nuevo motor de base de datos para Django *'django.db.backends.postgresql'*, los campos siguientes (el nombre, el usuario y la contraseña) por conveniencia se definieron como 'postgres'. El campo HOST se definió como 'db', que es el nombre del servicio configurado en el primer paso en el fichero *docker-compose.yml*, y por último el puerto a 5432, siendo este el puerto por defecto para PostgreSQL.

4. Por último se realizó la migración y se creó un nuevo *superusuario* para acceder a la interfaz de administración.

3.3.3. Modelos

Un modelo en Django es un tipo de objeto que se guarda en la base de datos. Se podría entender como una tabla que contiene los campos o atributos (*django fields*) que definen los comportamientos de los datos que se van a almacenar.

Los modelos que utilizamos en el proyecto se definen dentro del fichero *models.py* de

la aplicación “alumnos". Estos se implementan como subclases del módulo *models* que comunica a Django que el objeto creado es una tabla de la base de datos.

En la herramienta, se crea un modelo base *Alumnos* que contiene los campos que son genéricos en todos los grados (todos menos las asignaturas). Este primer modelo se define como abstracto, indicando que no será guardado en la base de datos.

El resto de modelos heredarán del modelo *Alumnos* los datos comunes a todos los grados y definirán los suyos propios. Estos modelos reciben el nombre del grado del que se quiere guardar la información. Se debe definir uno nuevo para cada titulación. En este proyecto se ha definido un modelo para el grado de Informática y otro para Mecánica.

Vemos los dos modelos alojados en la base de datos:

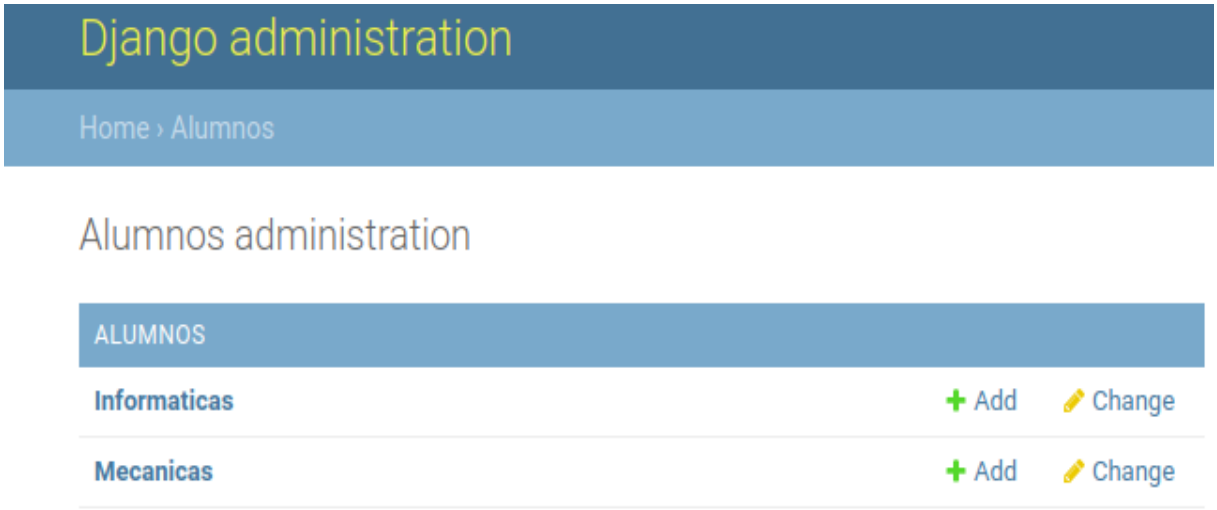


Figura 3.2: Modelos cargados en la base de datos.

3.3.4. Preprocesado de los datos

En este apartado se explica todo el proceso previo que experimentan los datos hasta ser cargados en la base de datos.

Formulario

Como se muestra en la Figura 3.3, el formulario contiene un total de cinco campos, tres de ellos son para subir los ficheros y dos campos para seleccionar, primero el grado, que define el modelo de la base de datos donde se almacenarán los datos, y luego el curso, que se guarda para cada usuario como un nuevo campo en la tabla junto con el resto de variables.

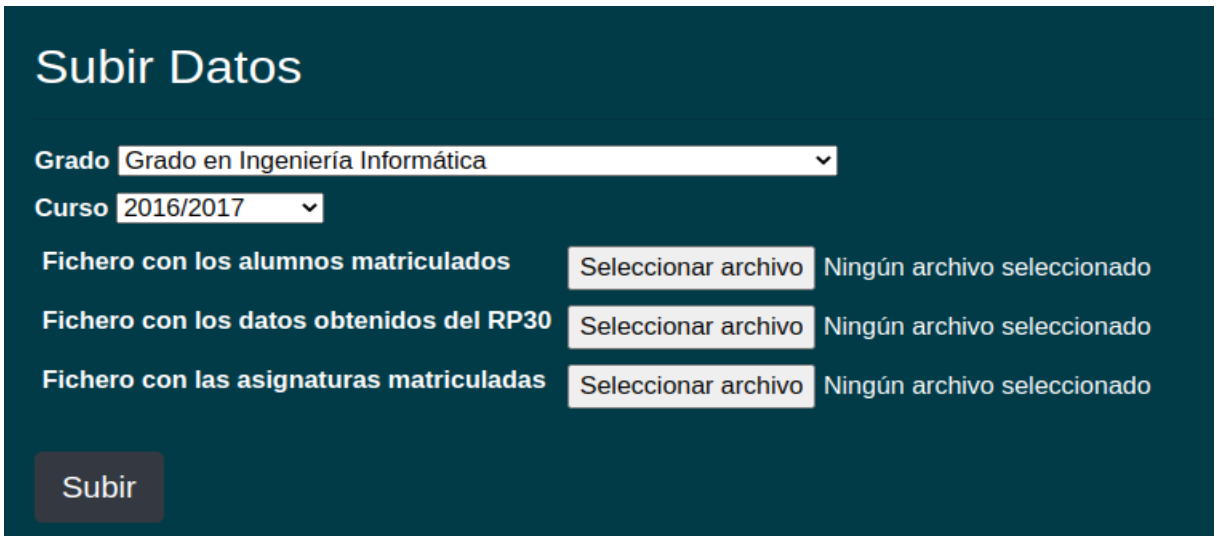


Figura 3.3: Formulario definido para cargar los datos.

Cuando el usuario envía los datos, la petición es capturada por la vista de la aplicación, que se encarga de gestionarla.

Vista

Una vista es una función en Python enlazada a una url que contiene toda la lógica que se encarga de gestionar las peticiones web de los usuarios y devolver una respuesta.

La vista obtiene los resultados del formulario, crea variables para almacenar cada uno de los campos, y pasa esta información a la función que contiene el *script* encargado de extraer la información de los ficheros y de crear las variables finales a almacenar en la base de datos.

Script

Al trabajar con tres ficheros de datos distintos, se desarrolló un *script* que fuese capaz de ir leyendo la información de cada uno de ellos, creando las variables requeridas para el proyecto.

El proceso definido en el *script* es el siguiente: Se guarda el contenido de esos ficheros en tres listas, donde cada fila en los ficheros de datos representa una sublista dentro de éstas. De la primera lista, obtenida del fichero con los alumnos matriculados, se obtienen dos nuevas listas que contienen los DNIs y los cursos a los que pertenecen los datos.

Como el DNI es un campo presente en todos los ficheros, todas las variables se pueden obtener recorriendo la lista con los DNIs. De esta forma, se definió un bucle que va

buscando el DNI correspondiente dentro de cada una de las sublistas, obteniendo las distintas variables.

Algunas variables, como el número de créditos (matriculados, presentados y aprobados) y las nota media, se obtienen a partir de otras variables, además de ser recogidas para cada curso académico. Para conseguir estos datos, es necesario recorrer la lista con los cursos dentro del bucle, guardando el número de asignaturas y las notas para luego contabilizar los valores y hallar los promedios.

Esta dinámica se repite para cada usuario; se crea una lista, se constituyen las variables y se añaden a dicha lista, cuando se cambia de DNI, la lista ya completa se almacena dentro de una lista final que, cuando se termine de recorrer el bucle contendrá las cabeceras y la información de cada uno de los alumnos.

Por último, la lista final se pasa a una nueva función junto con el valor de la variable grado. Esta nueva función comprueba la titulación a la que pertenecen los datos y crea un objeto con cada uno de los elementos de la lista para guardarlos en el modelo correspondiente.

3.3.5. Visualización de los datos

Formulario

En la parte encargada del análisis estadístico en la aplicación web, solo se trabaja con los alumnos de un curso y una titulación en concreto a la vez, debido a que se ha considerado que la forma correcta de trabajar con los datos es tratando de que todos obedezcan al mismo contexto, evitando posibles inferencias externas que puedan afectar al análisis.

Esa dinámica se mantiene a la hora de visualizar los datos, por lo que el formulario va a pedir al usuario el grado y el curso para mostrar esa información en concreto.

La tabla con las variables se cargará justo debajo, en caso de no existir datos almacenados para esa consulta, se mostrará un mensaje al usuario indicándolo. La tabla no recoge todas las variables, puesto que son muchas y dificultaría la lectura cargar una tabla con demasiada información, por ello se cargan las variables que se han considerado más relevantes: DNI, sexo, edad, procedencia, si trabaja, si recibe beca, convocatoria de prescripción, preferencia, modalidad, nota de admisión, los aciertos, fallos y total del RP30, abandono, créditos matriculados primer y segundo año y la nota media del primer y segundo año.

Figura 3.4: Formulario y muestra de datos.

Vista

La vista comprueba el valor del grado para realizar la consulta sql a la tabla correspondiente en la base de datos, a dicha consulta se le pasa como condición el curso. El resultado que devuelve la base de datos se guarda en una variable que devuelve la vista.

La variable devuelta contiene un array con todos los datos de cada alumno, esa variable está definida en el *template*, donde, mediante un bucle, se recorre el array y cada elemento se va añadiendo a un objeto html tabla que se muestra cuando la petición se resuelve.

3.4. Aplicación: Análisis estadístico

3.4.1. Diseño

Esta aplicación cuenta con dos *templates*, uno para el análisis estadístico unidimensional y otro para el análisis estadístico bidimensional. Estos *templates* o plantillas están divididos en dos partes, como se puede apreciar en la Figura 3.5, en la parte izquierda se muestra un panel, que contienen un formulario para que el usuario interactúe con las opciones destinadas a seleccionar los estadísticos que se pretenden generar. La otra parte, muestra un marco vacío donde se cargan los resultados estadísticos generados por la herramienta correspondientes a la petición del usuario.

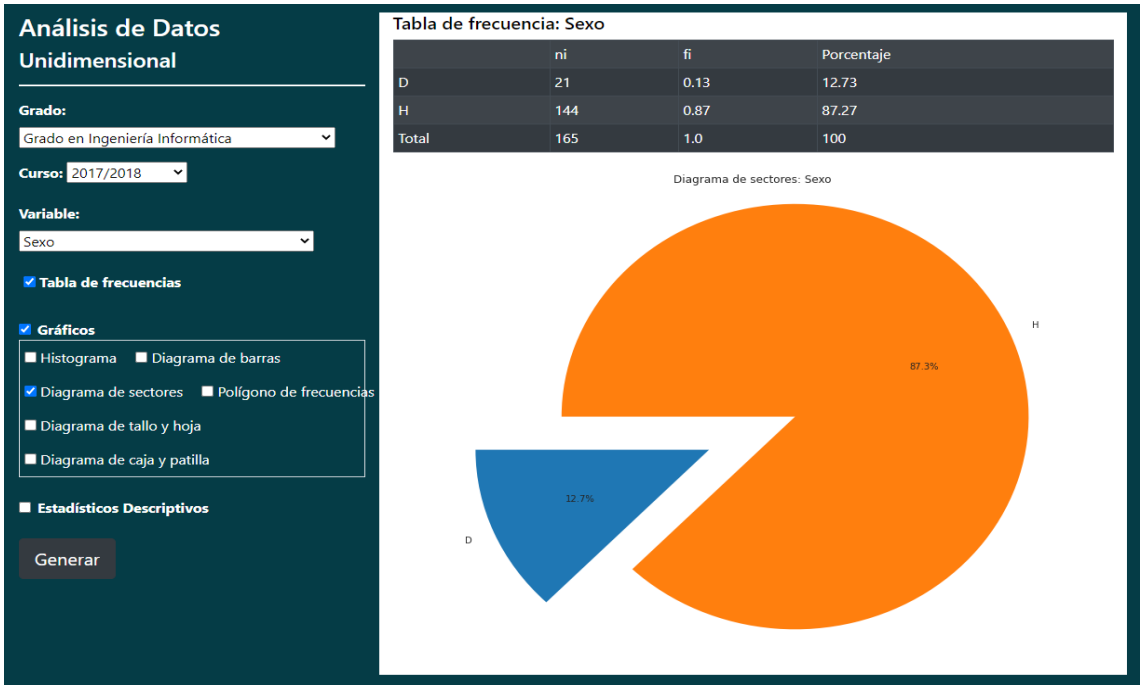


Figura 3.5: Diseño del panel definido para el análisis unidimensional.

3.4.2. Configuración

El flujo de trabajo que sigue la aplicación se conoce como patrón MVT[24] (Modelo - Vista - Template) que permite definir claramente cada una de las funciones de la que consta cada proceso y diferenciarlos del resto.

A lo largo del desarrollo del proyecto se ha podido comprender cuáles son esas funciones que cumplen los distintos procesos en Django, a modo de resumen se describen nuevamente:

- M - Modelo. Referente a la capa de datos (base de datos) dentro de la aplicación web. Sus funciones principales son: validación de datos, acceso a estos, sus comportamientos, la forma en la que son almacenados, etc.
- V - Vista. Contiene todo el código que gestiona la lógica de la aplicación. Comunica el modelo con el *template*.
- T - Template. Es la parte de presentación del proyecto, las que se muestra al usuario en la página web.

No obstante, para llegar a comprender el funcionamiento de esta aplicación en concreto, es necesario abordar detalladamente cada una de las fases en las que intervienen estos elementos y que acciones realizan.

3.4.3. Funcionamiento

El esquema que recoge la Figura 3.6 muestra el funcionamiento general que sigue la aplicación para generar la información estadística:

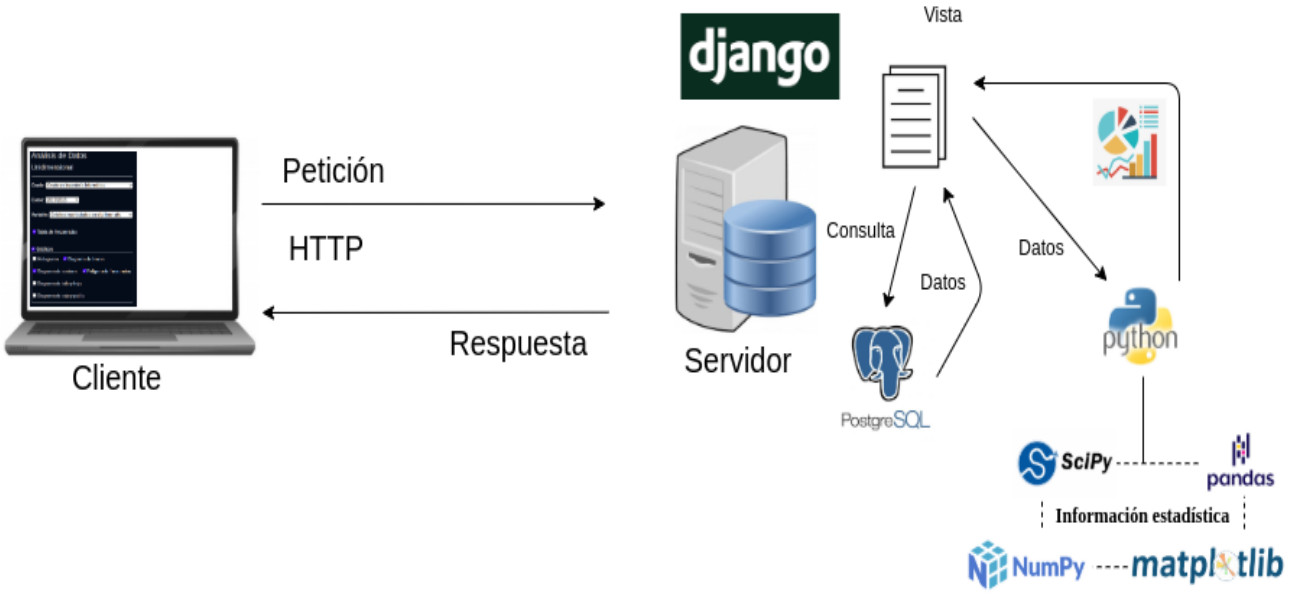


Figura 3.6: Esquema del funcionamiento de la herramienta. Fuente: Elaboración Propia.

En primer lugar el usuario rellena el formulario, señalando los procedimientos estadísticos que desea generar, y lo envía al servidor realizando una petición. Cabe destacar que, como se van a traer datos de la base de datos del servidor, la petición HTTP se realiza mediante el método GET.

La petición es capturada en Django por la vista, que tiene enlazada la dirección url desde la que se realizó y se encarga de gestionar toda la lógica de la misma; llamando a los métodos correspondientes para hacer la consulta a la base de datos y obtener los resultados estadísticos. La vista tiene creada una instancia del formulario y un diccionario "*context*"que guardará las variables con los resultados de la petición del usuario y que será enviado en la respuesta del servidor.

Una vez que el cliente envía el formulario, la vista comprueba su validez y el método empleado. Luego se encarga de obtener la información solicitada siguiendo la siguiente lógica:

1. Extrae las variables *grado*, *curso* y *variable* del formulario para realizar la consulta a la base de datos y conseguir la lista (o listas) con los datos sobre los que se va a realizar el análisis y llamar a un método que define el tipo al que pertenece la variable elegida, es decir si se trata de una variable cualitativa, discreta o continua.

2. Se crea una instancia de la clase correspondiente al análisis, donde se le pasa como parámetros los datos obtenidos en el paso anterior. En el caso del análisis unidimensional, a la clase correspondiente se le pasa el nombre de la variable, el tipo y la lista con los datos, y para el caso bidimensional se pasan los mismos elementos pero para cada una de las dos variable.

3. Luego la vista va comprobando aquellos campos que han sido seleccionados en el formulario para llamar a las funciones encargadas de esos procedimientos estadísticos. Por ejemplo, si para el análisis unidimensional el usuario ha pedido generar una tabla de frecuencias, la vista se encarga de llamar a la función definida con dicho fin en la clase, donde mediante las librerías estadísticas indicadas para esa tarea, se genera el resultado, que es enviada nuevamente a la vista, donde se almacena en el diccionario.

4. Cuando la vista ha revisado cada una de las opciones del formulario, envía la respuesta.

La plantilla, en la sección correspondiente, extrae la información del diccionario, construyendo y rellenando con dicha información los objetos html que se van a mostrar al usuario.

3.5. Análisis estadístico: Unidimensional

El siguiente esquema, recogido en la Figura 3.7, muestra la configuración implementada para el tratamiento de las variables en el análisis unidimensional.

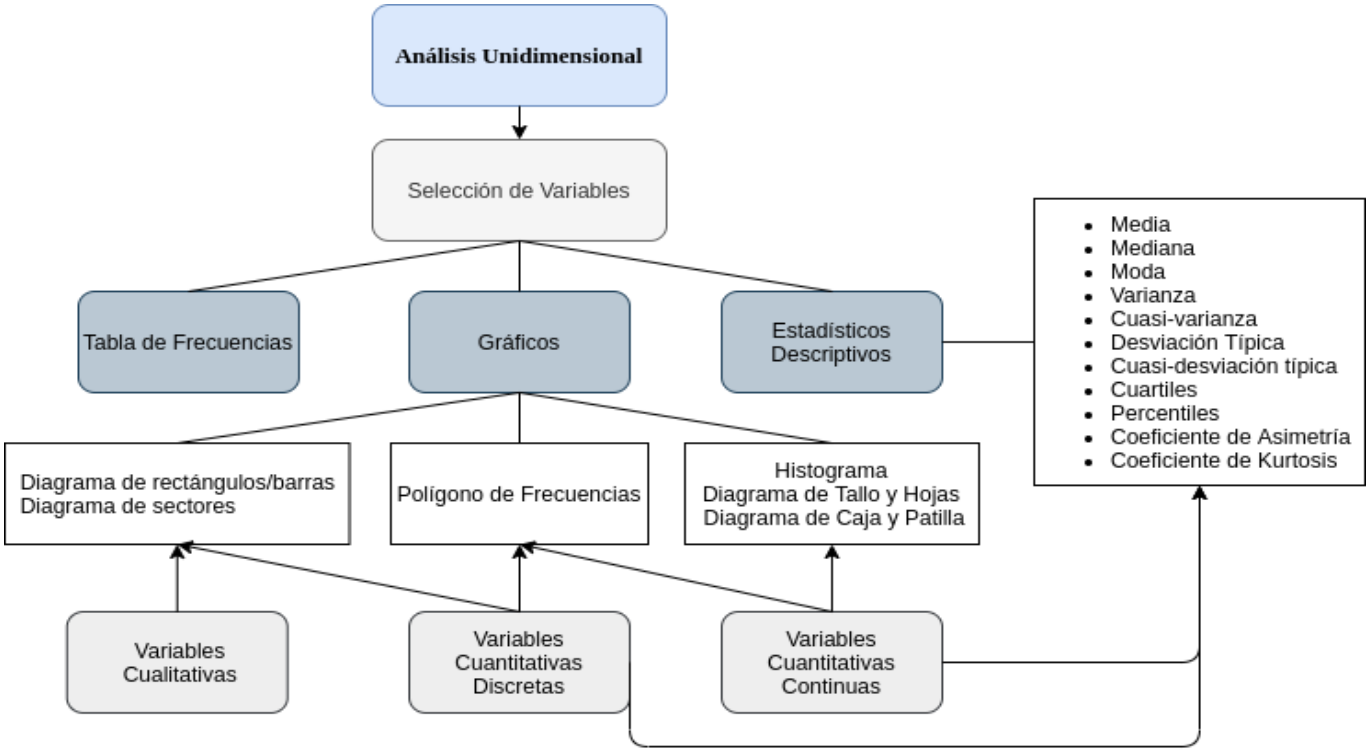


Figura 3.7: Esquema análisis unidimensional.

Se puede observar el flujo que se define en el formulario, donde primero se selecciona la variable a estudiar y luego cada una de las categorías disponibles para el estudio de los datos. En la figura también se observan los tipos de variables y elementos existentes, dentro esas categorías, que pueden generar.

3.5.1. Tabla de frecuencias

La tabla de frecuencias está disponible para cualquier variable, no obstante, los campos de la tabla varían en función la variable analizada.

Si se trabaja con una variable cualitativa, los campos que mostrará la tabla serán: las modalidades que toma la variable, las frecuencias absolutas, las frecuencias relativas y los porcentajes. Véase Figura 3.8.

Tanto para las variables cualitativas como para las variables cuantitativas discretas, para obtener las frecuencias absolutas se utiliza la función de pandas *value_counts*, que devuelve un objeto *Series* con el total para cada valor único de la variable. Para el resto

de campos utilizamos el mismo objeto con las modificaciones pertinentes, por ejemplo, para las frecuencias relativas se divide el objeto devuelto por el total de elementos de la muestra.

Tabla de frecuencia: Procedencia

	ni	fi	Porcentaje
Fuerteventura	1	0.01	0.61
La Palma	4	0.02	2.42
Lanzarote	1	0.01	0.61
Tenerife	159	0.96	96.36
Total	165	1.0	100

Figura 3.8: Tabla de frecuencia para la variable procedencia(cualitativa).

Si por el contrario trabajamos con variables discretas, a la tabla se le añaden, además de los campos anteriores, las frecuencias absolutas y relativas acumuladas.

Es importante hacer un inciso sobre el tratamiento que han tenido algunas variables discretas a la hora de ser mostradas tanto en la tabla de frecuencias como en los gráficos.

La variable 'Edad' y las variables destinadas a contar el número de créditos en cualquiera de sus formas (matriculados, presentados y aprobados), presentan muchos valores con una o muy pocas apariciones dentro del conjunto, alejados frecuentemente de la mediana, siendo su representación, tanto en la tabla de frecuencias como en los gráficos, ineficiente, por lo que muchos de estos valores se han agrupado bajo una misma denominación.

En la Figura 3.9 se observa un ejemplo donde se ha aplicado dicha modificación.

Tabla de frecuencia: creditos_matriculados_primer_año

	ni	Ni	fi	Fi	Porcentaje
Menos de 30	3	3	0.02	0.02	1.82
30	2	5	0.01	0.03	1.21
36	1	6	0.01	0.04	0.61
42	3	9	0.02	0.05	1.82
48	3	12	0.02	0.07	1.82
54	2	14	0.01	0.08	1.21
60	149	163	0.9	0.99	90.3
Más de 60	2	165	0.01	1.0	1.21
Total	165	165	1.0	1.0	100

Figura 3.9: Tabla de frecuencia para la variable "créditos matriculados en el primer año"(discreta).

Para las variables continuas se ha dado un procedimiento especial puesto que se han de agrupar mediante intervalos o clases: Primero se define el número de intervalos, luego el tamaño y por último los intervalos propiamente dichos. Para generar las frecuencias absolutas esta vez se emplea la función *cut* de pandas, encargada de agrupar los valores en los intervalos creados, para el resto de campos, el procedimiento es igual que en los casos anteriores. Además de los campos presentes para el resto de variables, a la tabla se añade la marca de clase como muestra la Figura 3.10.

Tabla de frecuencia: CYA

	Xi	ni	Ni	fi	Fi	Porcentaje
(2.0, 4.0]	3.0	13	13	0.26	0.26	26.0
(4.0, 6.0]	5.0	17	30	0.34	0.6	34.0
(6.0, 8.0]	7.0	15	45	0.3	0.9	30.0
(8.0, 10.0]	9.0	5	50	0.1	1.0	10.0
Total		50	50	1.0	1.0	100

Figura 3.10: Tabla de frecuencia para la variable “Computabilidad y Algoritmia”(continua).

3.5.2. Gráficos

Para generar los gráficos, el usuario despliega el menú del formulario “Gráficos” y selecciona aquellos que quiera mostrar. Cuando realiza la petición, la vista comprueba qué gráficos han sido marcados e introduce en una lista el nombre de cada uno de ellos para pasarla al método correspondiente dentro de la clase. Lo primero que hace la función es comprobar la viabilidad de crear los distintos gráficos en función del tipo de la variable empleada, luego crea un objeto figura al que se van añadiendo los distintos gráficos (*Axes*) presentes en la lista.

Para enviar la imagen a la vista, se ha de pasar en bytes, por lo que se guarda en un *buffer* y se codifica para luego ser añadida al *template*.

Los distintos gráficos ofrecidos por la herramienta para el análisis unidimensional son los siguientes:

- Gráfico de barras. Generado mediante el método *bar* de la biblioteca *matplotlib.pyplot*.

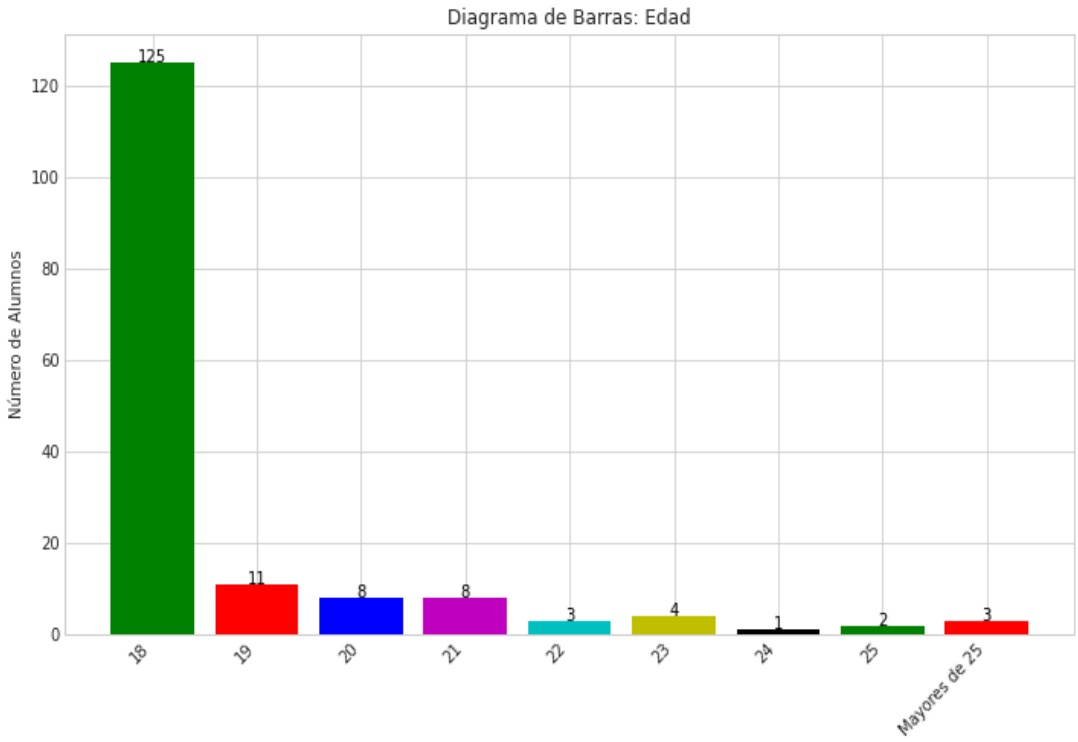


Figura 3.11: Gráfico de barras para la variable “Edad”.

- Diagrama de sectores. Generado mediante el método *pie* de la biblioteca *matplotlib.pyplot*.

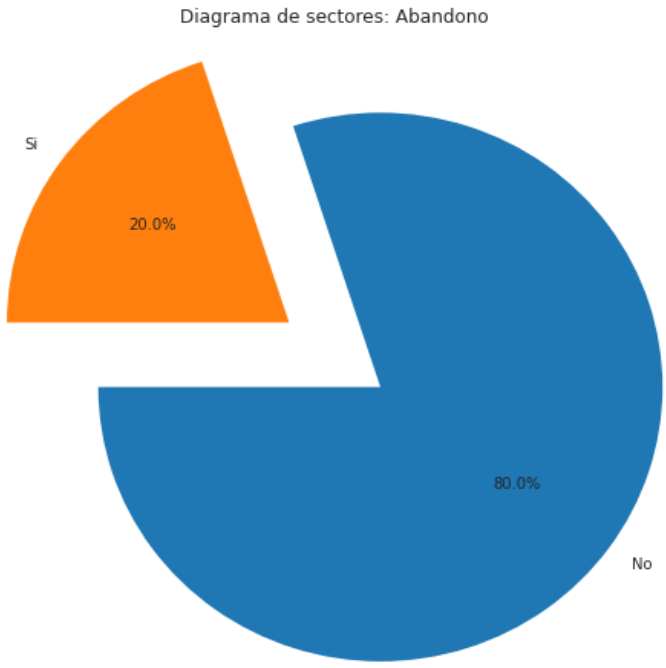


Figura 3.12: Gráfico de sectores para la variable “Abandono”.

- Polígono de frecuencias. Generado mediante el método *plot* de la biblioteca *matplotlib.pyplot*.

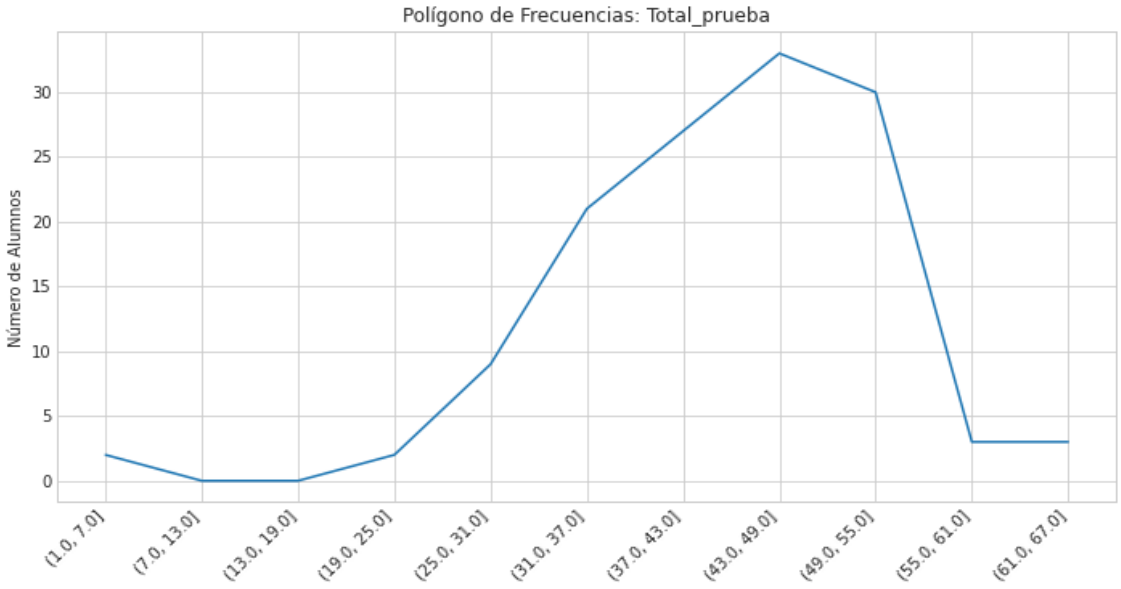


Figura 3.13: Polígono de frecuencias para la variable “RP30: Total”.

- Histograma. Generado mediante el método *hist* de la biblioteca *matplotlib.pyplot*.

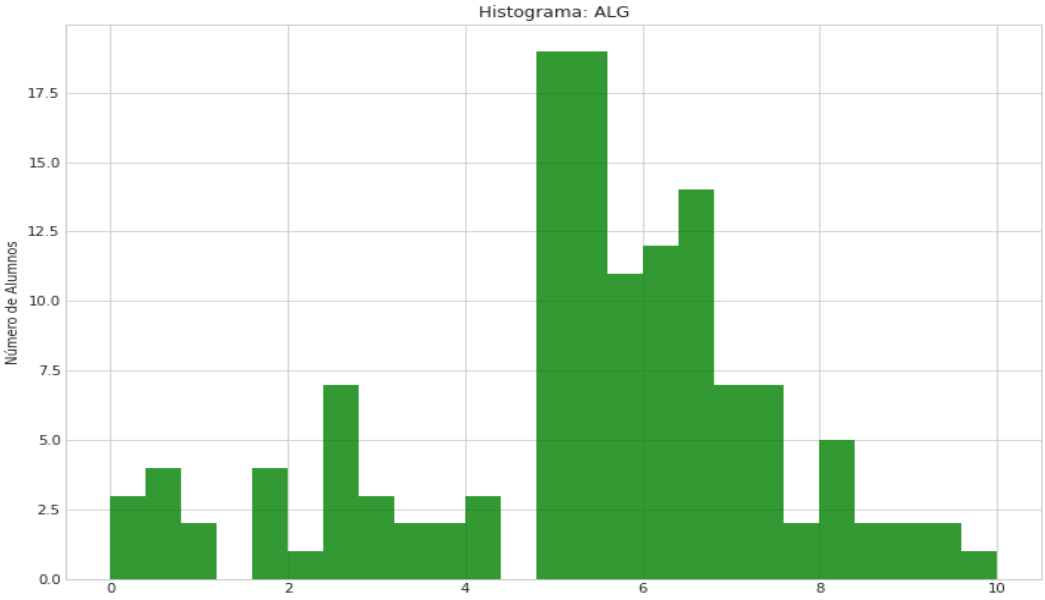


Figura 3.14: Histograma para la variable “Álgebra”.

- Diagrama de caja y patilla. Generado mediante el método *boxplot* de la biblioteca *matplotlib.pyplot*.

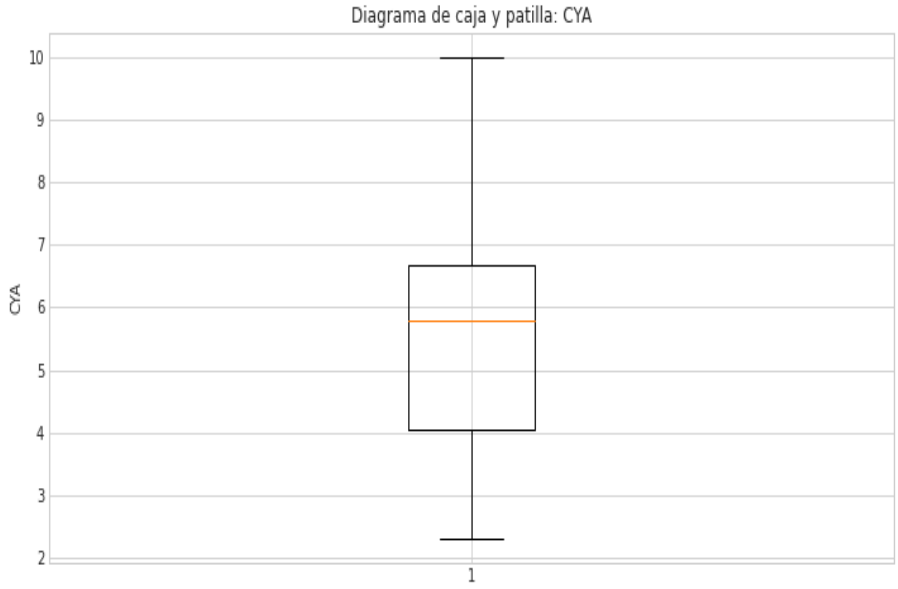


Figura 3.15: Diagrama de caja y patilla para la variable “Computabilidad y Algoritmia”.

- Diagrama de tallo y hoja. Generado mediante el método *stem_graphic* de la biblioteca *stemgraphic*.

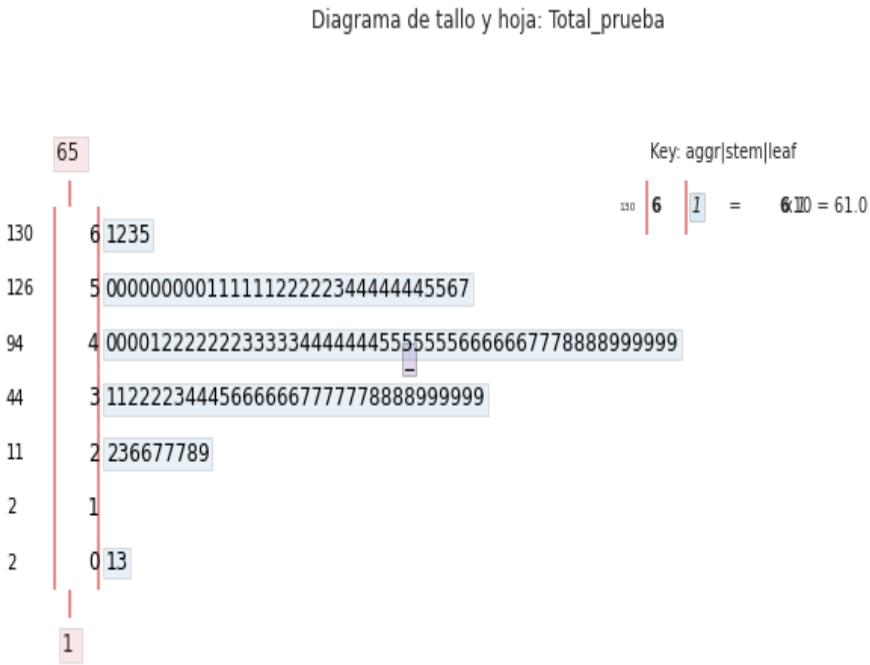


Figura 3.16: Diagrama de tallo y hoja para la variable “RP30: Total”.

3.5.3. Estadísticos descriptivos

El procedimiento para generar los estadísticos descriptivos sigue el mismo proceso que para el caso de los gráficos: El usuario selecciona del menú los estadísticos a generar, la vista los captura e incluye en una lista, la lista se pasa a la función que los produce y envía para ser mostrados en el *template*.

En la Figura 3.17 se observa la salida para una consulta donde se han seleccionado todos los estadísticos descriptivos disponibles de la variable *Nota media primer año*.

Estadísticos Descriptivos

Media ($\bar{x}$)	5.53
Mediana (M)	5.615
Moda (Mo)	4.38
Varianza (σ^2)	2.67
Cuasi-varianza (s^2)	2.68
Desviación Típica (σ)	1.63
Cuasi-desviación Típica (s)	1.64
Cuartiles	Q1: 4.34, Q2: 5.62, Q3: 6.6
Percentiles	P10: 3.53, P50: 5.62, P90: 7.62
Coeficiente de Asimetría	-0.05
Coeficiente de Kurtosis	-0.19

Figura 3.17: Tabla con los estadísticos para la variable “Nota media primer año”.

3.6. Análisis estadístico: Bidimensional

El diseño para trabajar en el análisis bidimensional se recoge en la Figura 3.18.

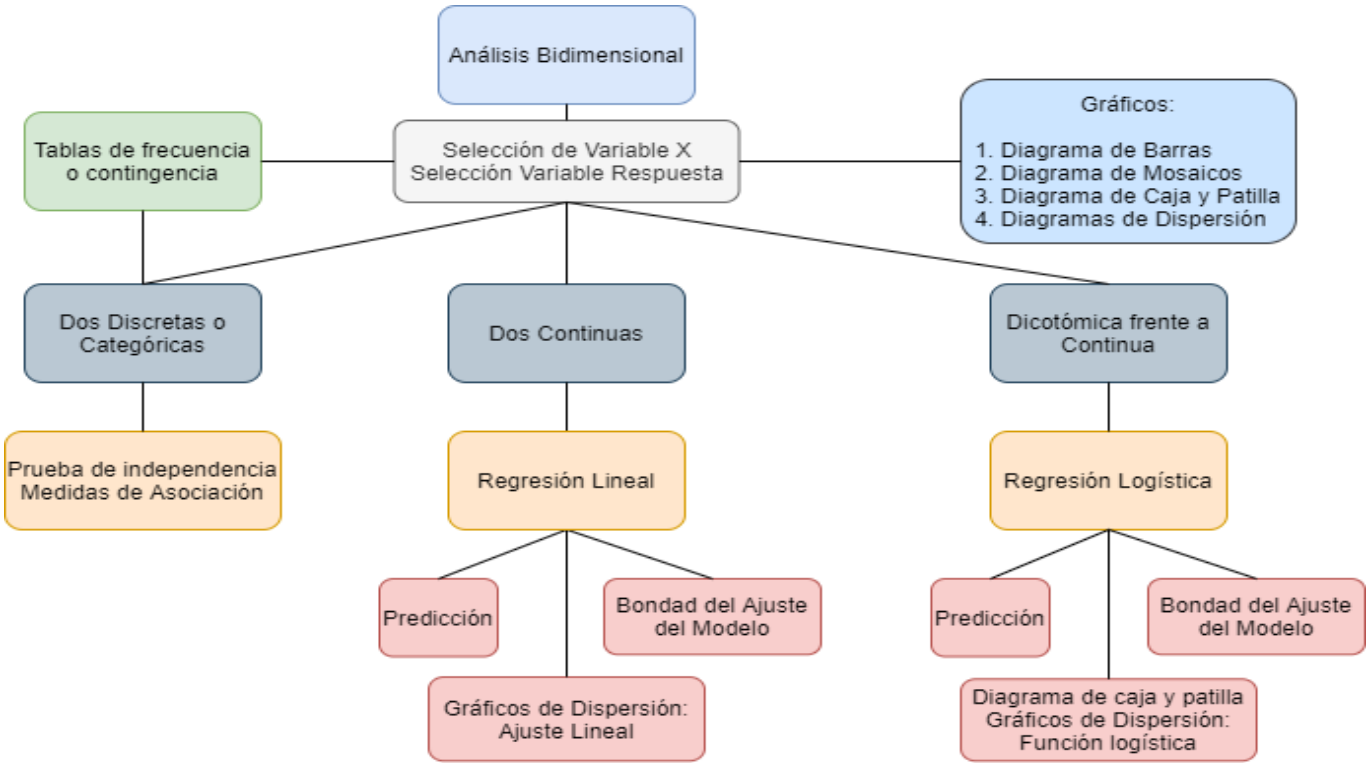


Figura 3.18: Esquema análisis bidimensional.

Para el tratamiento de datos dentro de la clase definida para el análisis bidimensional se hace uso de la estructura de datos *dataframe* ofrecida por *pandas*.

3.6.1. Análisis de datos categóricos

Tabla de contingencia

Para generar las tablas de frecuencias o contingencias se hace uso de un *dataframe* con ambas listas, donde el contenido de cada columna representa una columna en la tabla que se devuelve.

Tablas de Frecuencias Bidimensionales: Sexo y Abandono

Abandono	No	Si	Total
Sexo			
D	20	1	21
H	112	32	144
Total	132	33	165

Figura 3.19: Tabla de contingencia para las variables “Sexo” y “Abandono”.

3.6.2. Gráficos

Para generar los gráficos se sigue el mismo procedimiento que en la parte unidimensional. Los gráficos que soporta la aplicación son los siguientes:

- Diagrama de barras: tanto el diagrama de barras apilado como el adosado. Se generan mediante los métodos *plot.bar* y *plot.barh(stacked=True)*, respectivamente, sobre el *dataframe*.

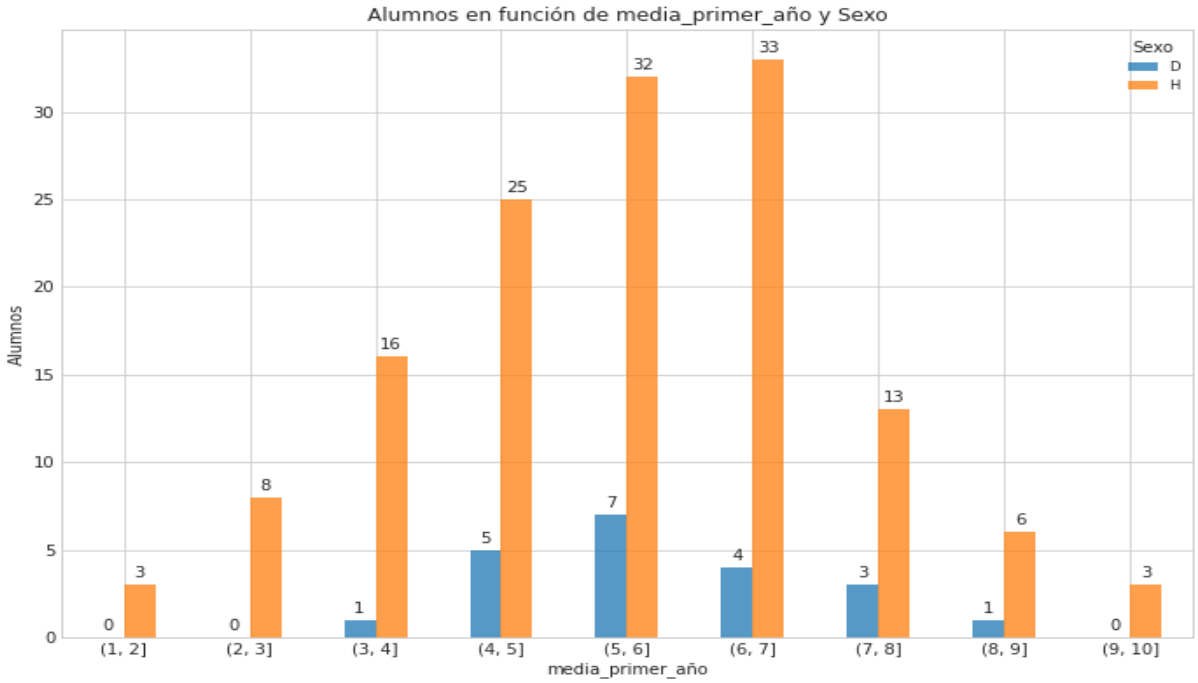


Figura 3.20: Gráfico de barras adosado para las variables “Nota media del primer año” y “Sexo”.

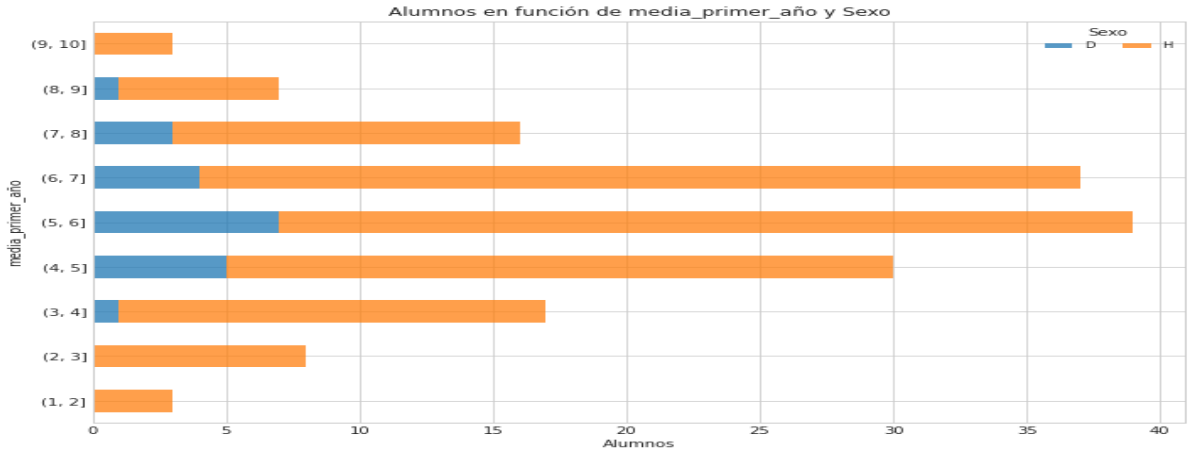


Figura 3.21: Gráfico de barras apilado para las variables “Nota media del primer año” y “Sexo”.

- Diagrama de mosaicos. Generado mediante *graphics.mocaicplot* del paquete *statsmodels*.

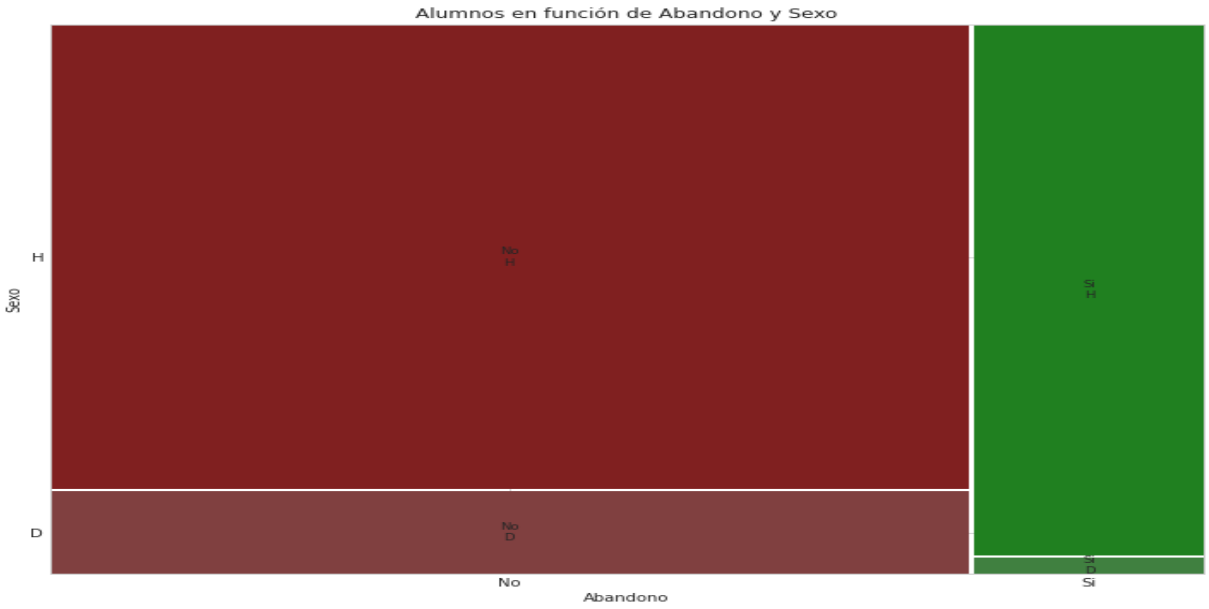


Figura 3.22: Gráfico de mosaicos para las variables “Abandono” y “Sexo”.

- Diagrama de caja y patilla. Generado mediante el método *bloxplot* de la librería *seaborn*.

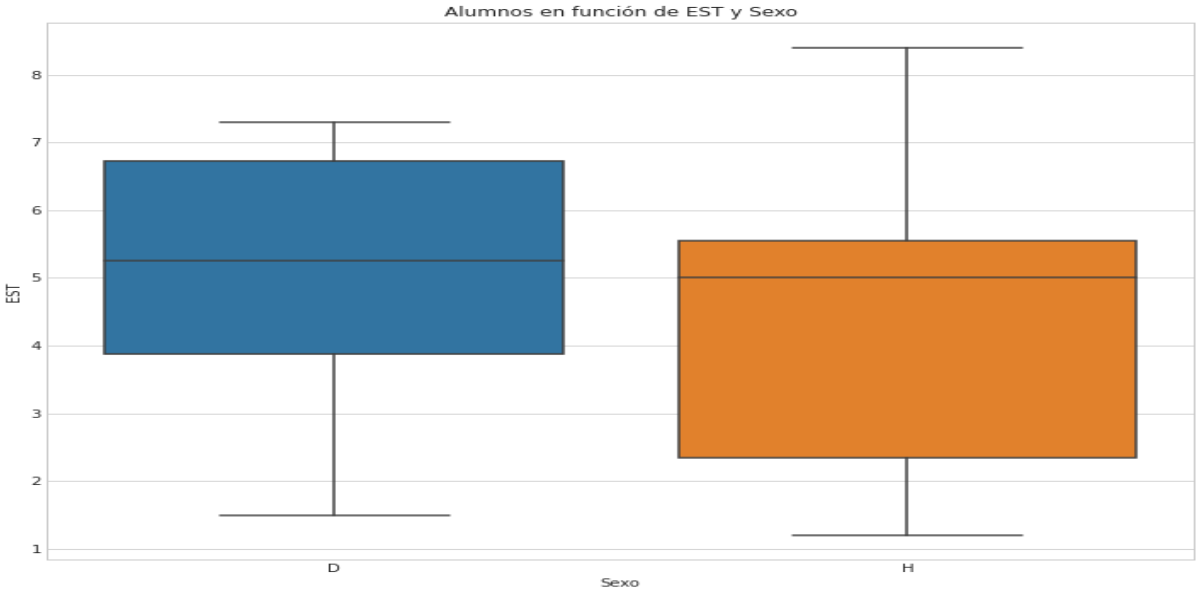


Figura 3.23: Gráfico de caja y patilla para las variables “Estadística” y “Sexo”.

- Diagrama de dispersión. Para el ajuste lineal se utiliza el método *regplot* de la librería *seaborn*.

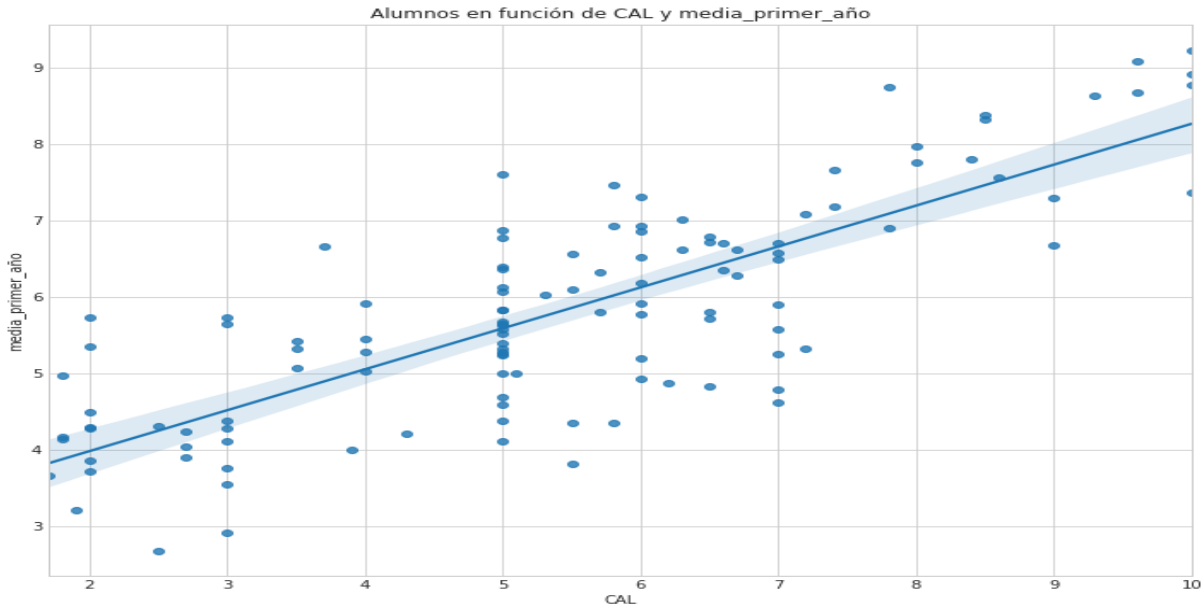


Figura 3.24: Diagrama de dispersión con ajuste lineal para las variables “Cálculo” y “Nota media del primer año”.

Para el diagrama de dispersión con la función sigmoide superpuesta, se emplea el gráfico básico (*plot*) de la librería *matplotlib* al que se le pasa como parámetro la función *predict_proba*, del modelo de regresión logística perteneciente a *sklearn*, que se encarga de dibujar la curva.

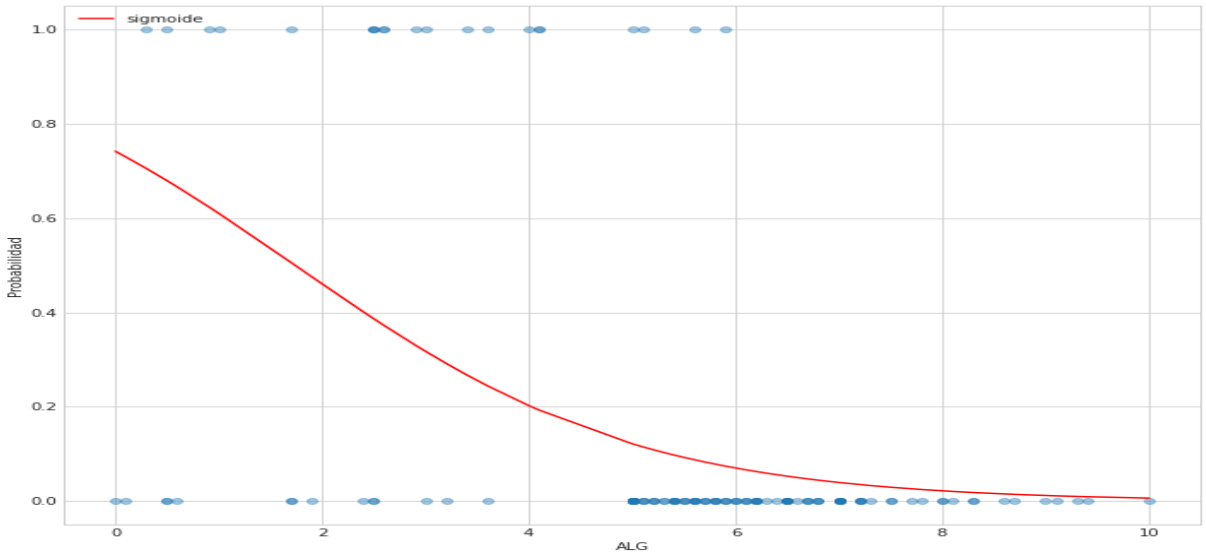


Figura 3.25: Diagrama de dispersión con ajuste de función sigmoide para las variables “Álgebra” y “Abandono”.

3.6.3. Modelo: Regresión lineal simple

Para construir los modelos de regresión lineal se utilizó la estimación de mínimos cuadrados ordinarios, mediante el módulo OLS (*ordinary least squares*) del paquete *statsmodels*. El primer argumento pasado a este método es la variable dependiente Y y el segundo la variable independiente X. Se utiliza el método *fit()* para entrenar el modelo:

```
modelo = sm.OLS(Y, X).fit()
```

Dentro de la función encargada de esta parte, se definieron los elementos que permitieron extraer la información estadística requerida para este análisis, de forma que puedan ser mostrados en una tabla al usuario.

Para obtener la estimación de los parámetros $\beta 0$ y $\beta 1$ del modelo, se emplea el atributo *params*, que devuelve una lista con ambos valores.

El método *summary()* devuelve los resultados estadísticos correspondientes al modelo, desde donde se extraen los contrastes sobre $\beta 0$ y $\beta 1$, y los coeficientes de correlación y determinación.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       media_primer_año   R-squared:                       0.632
Model:                            OLS   Adj. R-squared:                  0.629
Method:                 Least Squares   F-statistic:                     202.9
Date:                Wed, 09 Sep 2020   Prob (F-statistic):           2.11e-27
Time:                        10:31:01   Log-Likelihood:                -152.65
No. Observations:                 120   AIC:                             309.3
Df Residuals:                     118   BIC:                             314.9
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.9177      0.218     13.378      0.000       2.486       3.350
CAL            0.5350      0.038     14.245      0.000       0.461       0.609
==============================================================================
Omnibus:                        1.612   Durbin-Watson:                   2.151
Prob(Omnibus):                  0.447   Jarque-Bera (JB):                1.681
Skew:                          -0.252   Prob(JB):                        0.432
Kurtosis:                       2.714   Cond. No.                         16.3
==============================================================================
```

Figura 3.26: Salida de del método *summary()* sobre el modelo lineal para las variables “Cáculo” y ”Nota media del primer año”.

Para los resultados del ANOVA, se emplea la función *anova_lm* sobre el modelo, los resultados que se obtienen son:

	df	sum_sq	mean_sq	F	PR(>F)
media_primer_año	1.0	82.095534	82.095534	27.613297	0.000003
Residual	51.0	151.625221	2.973044	NaN	NaN

Figura 3.27: Salida del método*anova_lm* sobre el modelo lineal para las variables “Cáculo” y ”Nota media del primer año.

Para la predicción se toma el valor de la variable “x” del formulario y mediante la función *predict* del módulo se obtiene el valor “y”:

```
y_pred = modelo.predict(x)
```

3.6.4. Modelo: Regresión logística

Por último, el modelo de regresión logística se elaboró mediante el método *logit* de *statsmodels*. Antes de aplicar el modelo se debe sustituir en el *dataframe* los valores de la variable dicotómica dependiente Y por su codificación en 0s y 1s.

El proceso es similar que para el modelo lineal: se crea el modelo y se entrena, los estadísticos se obtienen nuevamente con el parámetro *summary()*:

Logit Regression Results

Dep. Variable:	Abandono	No. Observations:	160
Model:	Logit	Df Residuals:	158
Method:	MLE	Df Model:	1
Date:	Wed, 09 Sep 2020	Pseudo R-squ.:	0.1956
Time:	15:39:09	Log-Likelihood:	-60.910
converged:	True	LL-Null:	-75.725
Covariance Type:	nonrobust	LLR p-value:	5.229e-08

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	2.4620	0.819	3.005	0.003	0.856	4.068
media_primer_año	-0.7996	0.172	-4.636	0.000	-1.138	-0.462

Figura 3.28: Salida del método *summary()* sobre el modelo de regresión logístico para las variables “Nota media del primer año” y ”Abandono”.

De los resultados, se toman los estadísticos requeridos para el análisis para ser mostrados al usuario.

Capítulo 4

Verificación y pruebas

En este capítulo se describe el tratamiento que se da a los errores en la herramienta y se comprueba la funcionalidad de ambos modelos predictivos analizando los resultados obtenidos.

4.1. Tratamiento de errores

Los errores que se pueden producir durante el análisis estadístico obedecen siempre a una misma causa: el usuario solicita un procedimiento estadístico que no es compatible con la variable (o variables) que ha seleccionado.

Para localizarlos se comprueba siempre el tipo de las variable, y si se detecta una incompatibilidad, se comunica al usuario mediante un mensaje de error que se carga justo al principio de los resultados de la petición. En la Figura 4.1 se aprecia un ejemplo de como se muestran estos errores.

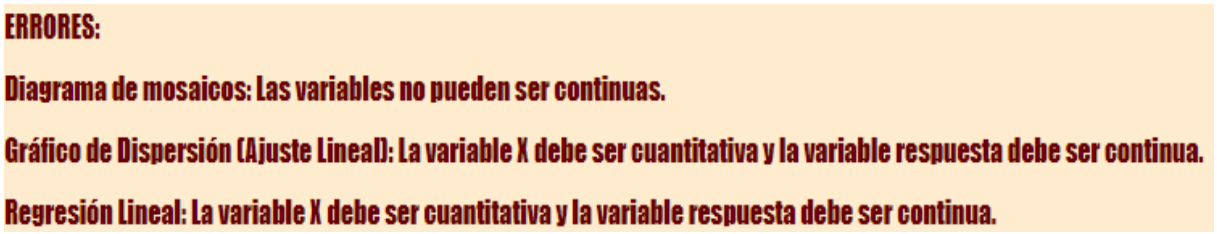
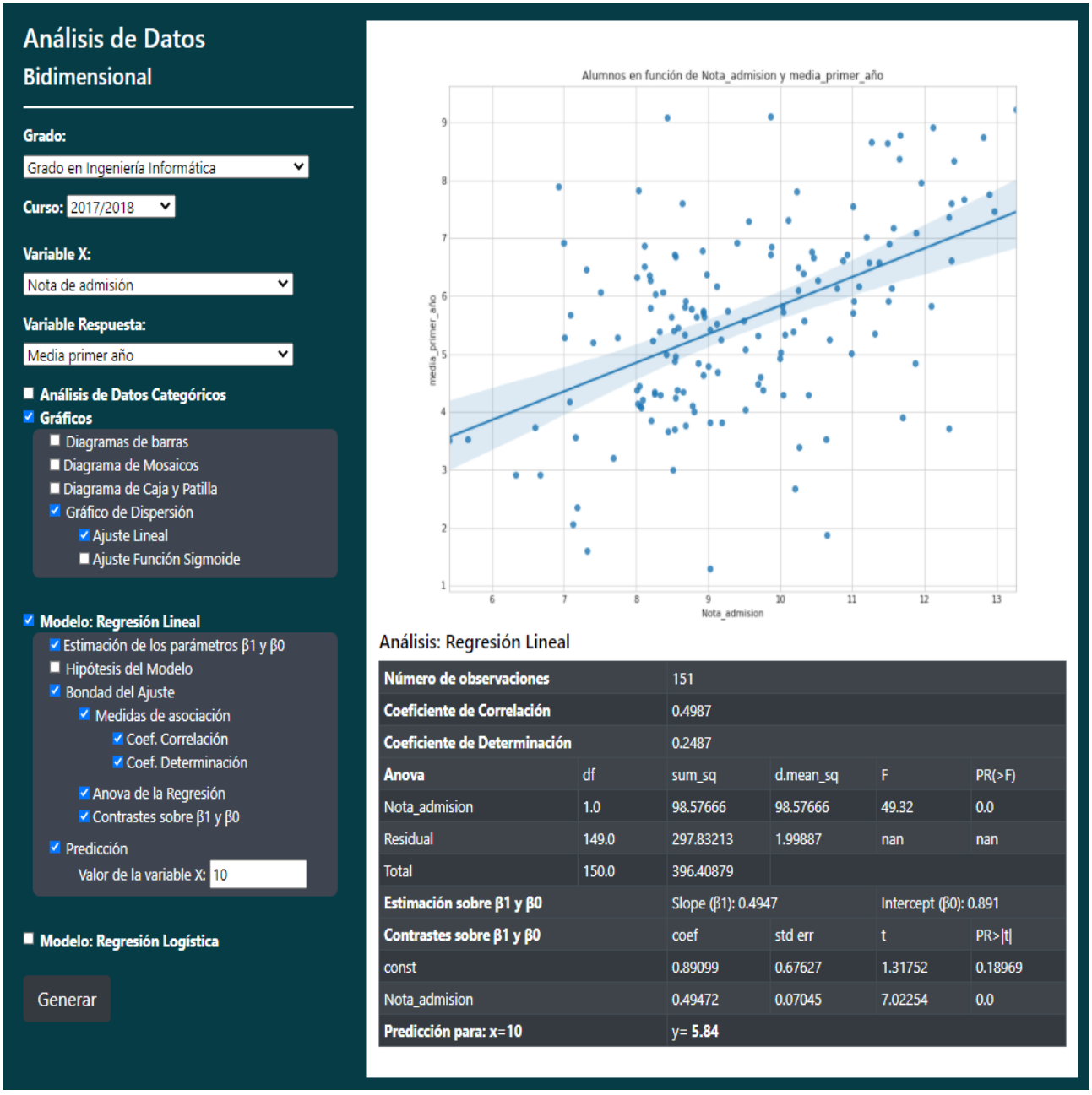


Figura 4.1: Errores en una consulta

4.2. Prueba: Bidimensional

4.2.1. Prueba 1: Modelo regresión lineal

Para esta prueba se buscó determinar la relación lineal existente entre la variable *Nota de admisión* para el grado y la variable *Nota media del primer año*. En la Figura 4.2 se observan los campos seleccionados para el análisis y la respuesta que se obtuvo.



Número de observaciones		151			
Coeficiente de Correlación		0.4987			
Coeficiente de Determinación		0.2487			
Anova	df	sum_sq	d.mean_sq	F	PR(>F)
Nota_admision	1.0	98.57666	98.57666	49.32	0.0
Residual	149.0	297.83213	1.99887	nan	nan
Total	150.0	396.40879			
Estimación sobre β1 y β0		Slope (β1): 0.4947		Intercept (β0): 0.891	
Contrastes sobre β1 y β0		coef	std err	t	PR>|t|
const		0.89099	0.67627	1.31752	0.18969
Nota_admision		0.49472	0.07045	7.02254	0.0
Predicción para: x=10		y= **5.84**			

Figura 4.2: Prueba: Modelo de regresión lineal

Se comentan paso por paso los resultados obtenidos del análisis:

En primer lugar, se observa que los puntos en el gráfico están muy dispersos; existe una tendencia positiva pero con una relación muy baja. Pese a ello, se puede apreciar

levemente que a medida que aumentan el valor de la variable nota de admisión también aumenta el valor de la nota media del primer año.

Analizando los resultados expuestos en la tabla, se tiene que el coeficiente de determinación es de 0.2487, un valor muy próximo a 0, que indica un ajuste pobre, denotando de entrada un modelo poco fiable.

En cuanto a los contrastes sobre $\beta 1$ y $\beta 0$, el valor del estadístico t es muy pequeño, siendo el p-valor de 0,18969, muy por encima del valor del nivel de significación (0,05), lo que indica que el modelo no es adecuado para ajustar a la nube de puntos.

Se concluye por tanto, que la variable *Nota de admisión* no es un buen indicador para predecir la nota media de los alumnos de primer curso, por lo que el valor predicho por el modelo no es significativo.

4.2.2. Prueba 2: Modelo regresión logística

Para la segunda prueba sobre el proyecto, se han comparado los resultados obtenidos en dos modelos de regresión logística. El primer modelo trata de ver si existe relación entre la variable nota media del primer curso y el abandono, para en caso afirmativo predecir la probabilidad de abandono en función de los valores de la nota media. El segundo modelo es similar, pero esta vez se analiza la relación entre la variable RP30:Total y el abandono.

Los gráficos utilizados en esta prueba han sido: diagrama de caja y patilla de las variables nota media del primer año y resultados del RP30, para cada categoría de la variable abandono y el diagrama de dispersión con ajuste a la función sigmoide.

Comenzamos comparando los diagramas de caja y patilla: Ya en la Figura 4.3 se puede apreciar que la variable (*Nota media del primer año*) se centra en distintos valores para los alumnos que han continuado en el grado que para los alumnos que han abandonado la titulación, ya que la mediana, línea central de la caja, para los primeros se sitúa en un valor próximo a 6 mientras que para los segundos es apenas superior a 4. Aunque ambas cajas tienen una altura parecida, las patillas de los que no abandonan son mayores, por tanto, hay más dispersión. Por su parte, el diagrama de caja y patillas para el modelo basado en la variable *RP30: Total* (Figura 4.4) muestra que los datos se centran en el mismo valor, si nos fijamos en ambas medianas. En el caso de los alumnos que continúan el 50 % de los datos centrales están más cerca de la mediana, sin embargo, ocurre como en el gráfico anterior, tienen más dispersión los que no abandonan por tener mayores

patillas. También la distribución de los que continúan es más simétrica, la otra parece que presenta una asimetría a la izquierda, porque la mediana no está en el centro de la caja sino ligeramente superior.

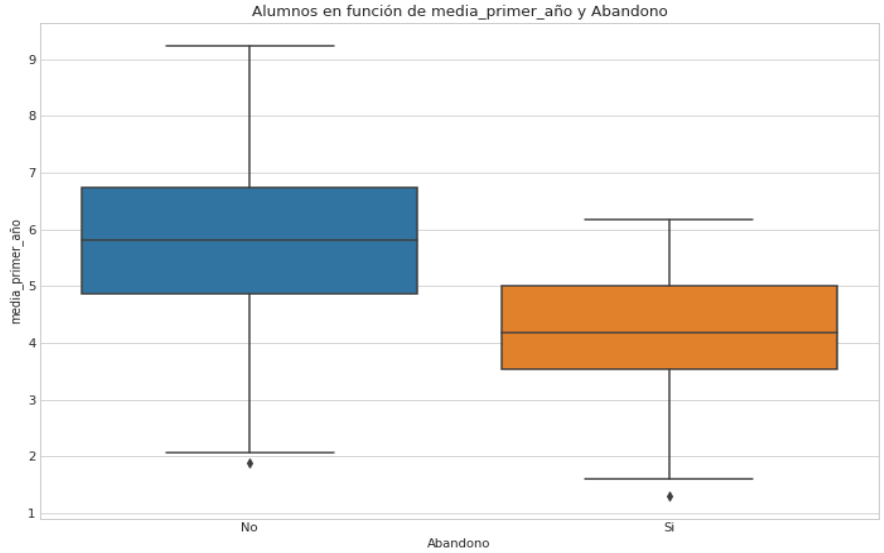


Figura 4.3: Diagrama de caja y patilla: Modelo de regresión logística 1

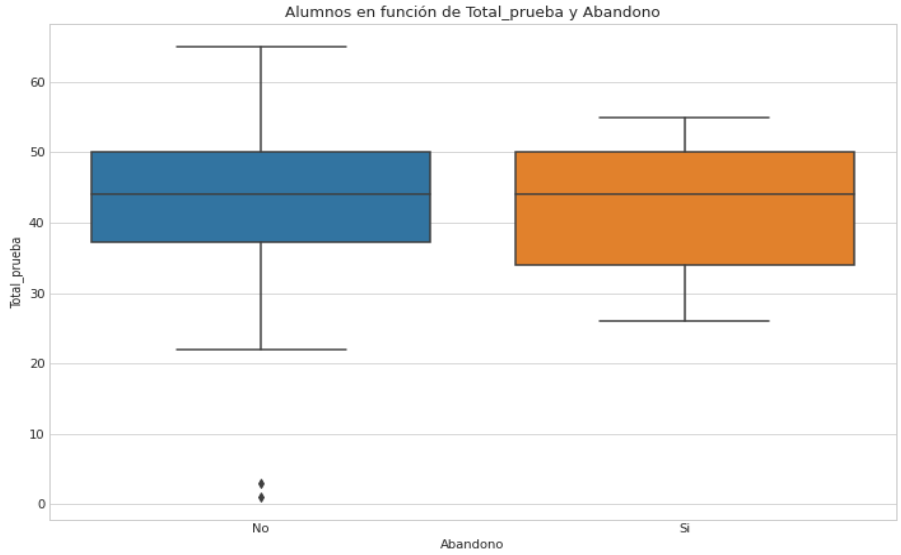


Figura 4.4: Diagrama de caja y patilla: Modelo de regresión logística 2

Siguiendo con el análisis de los gráficos, para el diagrama de dispersión con ajuste función sigmoide también se aprecian diferencias importantes. En el primer caso (Figura 4.5), se puede apreciar una que, para valores bajos de la nota media, la curva se intenta aproximar a 1 de la variable abandono, mientras que para valores grandes la curva se aproxima a 0, valor de no abandona. En el segundo modelo (Figura 4.6), la curva es prácticamente inexistente, dando a entender que no son variables óptimas para un

modelo de estas características.

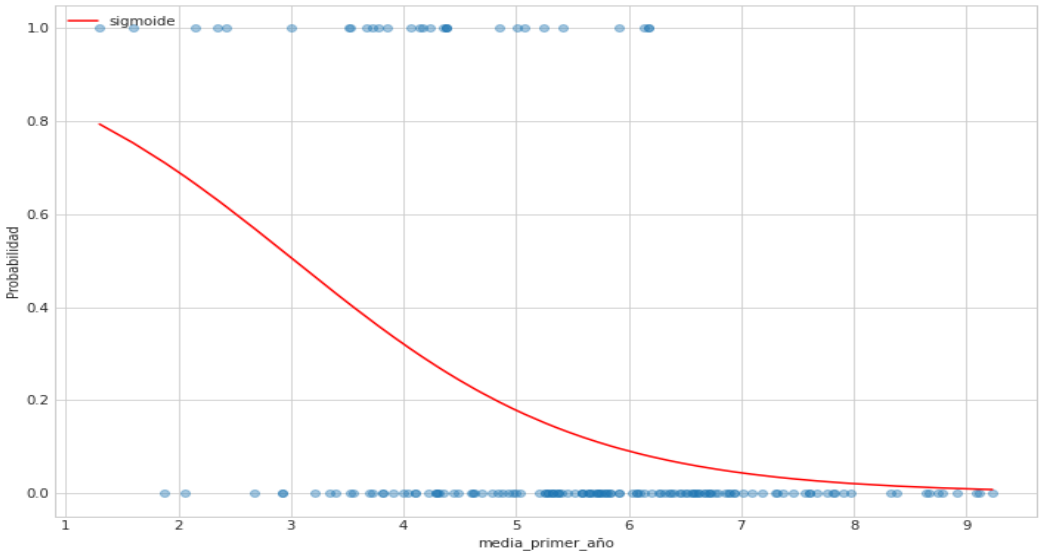


Figura 4.5: Diagrama de dispersión: Modelo de regresión logística 1

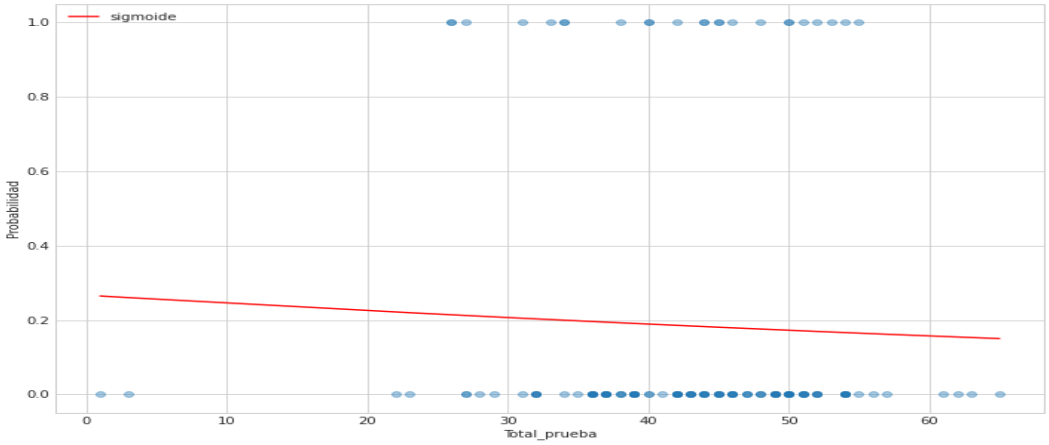


Figura 4.6: Diagrama de dispersión: Modelo de regresión logística 2

Las técnicas estadísticas por su parte han sido: la estimación de los parámetros $\beta 0$ y $\beta 1$, la bondad de ajuste del modelo mediante los contrastes sobre $\beta 0$ y $\beta 1$, el test de verosimilitud, el coeficiente pseudo R^2 de McFadden y la predicción. Los resultados obtenidos son los siguientes:

Empezando por el primer modelo (Figura 4.7), los resultados del contraste sobre los parámetros devuelven un p-valor de 0, por lo tanto la variable nota media influye en la decisión de abandonar o no. El valor del estadístico R^2 de McFadden es 0.80436, lo que implica que el modelo de regresión logística con la variable nota media es un 80,436 más probable (o mejor) que sin ella. Y como el pseudo R^2 de McFadden vale 0.19564 la variable nota media tiene una capacidad de explicar el abandono del 19.564 %. Por

último, el test de verosimilitud muestra un p-valor de 0, un ratio elevado y si comparamos el *log-likelihood*, que es el logaritmo de la verosimilitud de los datos observados, con el modelo con variable predictora, es mayor -60,9104 que el mismo en el modelo nulo o sin variable predictora, que vale -75,7254, determinando nuevamente un buen ajuste del modelo.

Análisis: Regresión Logística

Número de observaciones	160			
Estimación de β1 y β0	Slope (β1): -0.7996		Intercept (β0): 2.462	
Contrastes sobre β1 y β0	coef	std err	z	P>|z|
const	2.46195	0.8194	3.00459	0.00266
media_primer_año	-0.79958	0.17246	-4.6364	0.0
Test de Verosimilitud	Log-likelihood	LLnull	Likelihood ratio	LLR p_value
	-60.9104	-75.7254	29.63	0.0
Coeficiente R2 de McFadden	0.80436	**Pseudo R2 de McFadden**		0.19564
Predicción para: x=3.5	Probabilidad: **0.41664**			

Figura 4.7: Tabla con el análisis: Modelo de regresión logística 1

En el segundo modelo(Figura 4.8), para los contrastes y el test de verosimilitud, se obtienen p-valores muy superiores a 0,05 y los valores del estadístico z y del *likelihood ratio* son muy pequeños, enfatizando la falta de relación entre las variables y la poca robustez del modelo. El pseudo R^2 de McFadden devuelve un valor demasiado pequeño como para que el modelo pueda ser considerado.

Análisis: Regresión Logística

Número de observaciones	130			
Estimación de β1 y β0	Slope (β1): -0.0111		Intercept (β0): -1.0109	
Contrastes sobre β1 y β0	coef	std err	z	P>|z|
const	-1.01095	0.95951	-1.05361	0.29206
Total_prueba	-0.01114	0.02208	-0.5047	0.61377
Test de Verosimilitud	Log-likelihood	LLnull	Likelihood ratio	LLR p_value
	-62.0568	-62.1816	0.2496	0.6174
Coeficiente R2 de McFadden	0.99799	**Pseudo R2 de McFadden**		0.00201
Predicción para: x=60	Probabilidad: **0.15716**			

Figura 4.8: Tabla con el análisis: Modelo de regresión logística 2

Se puede concluir por lo tanto que la nota media de los alumnos de primer año si es un buen indicador para determinar la probabilidad de que un alumno se encuentra de abandonar el grado. Por el contrario, los resultados del test psicotécnico RP30 no permiten predecir la probabilidad respecto al abandono.

Capítulo 5

Conclusiones y líneas futuras

5.1. Conclusiones

Un proyecto de estas características requirió de un planteamiento extenso. Supuso trabajar con un elevado número de herramientas y bibliotecas, además de procedimientos estadísticos, donde el proceso de aprendizaje tuvo que ser constante.

Los objetivos principales propuestos se han cumplido. La aplicación web es funcional, se puede obtener información estadística sobre los datos de los alumnos de nuevo ingreso en informática y ayuda a analizar qué variables están más relacionas entre ellas o influyen en el abandono. La herramienta ha permitido conocer que los resultados del RP30 no guardan relación con el abandono y no es un indicador adecuado para trabajar esta problemática.

El desarrollo de este proyecto ha dejado muchas enseñanzas: se conocieron nuevos modelos y técnicas estadísticas, se ha interiorizado el funcionamiento del patrón MVT comprendiendo cada uno de sus elementos, se han aprendido a utilizar nuevas librerías y herramientas, etc. Además de que se ha experimentado todo el proceso que supone definir un proyecto web desde su planteamiento inicial hasta su despliegue en la nube.

5.2. Mejoras

Se han considerado una serie de propuestas para mejorar el funcionamiento de la herramienta, que se recogen en los siguientes subapartados.

5.2.1. Definir usuarios y roles

Una de las principales funcionalidades que puede requerir esta herramienta es un sistema de autenticación de usuarios que permita ocultar ciertas aplicaciones del proyecto a los usuarios no registrados.

En concreto, la aplicación del proyecto definida para subir los datos no debería estar disponible para cualquier visitante en la web, tampoco es pertinente que cualquier usuario pueda visualizar los datos almacenados, se está tratando con información académica de alumnos reales y no es conveniente que esté disponible a cualquier persona.

5.2.2. Aplicar mejoras en el Front-end

El aspecto visual de la aplicación web no ha sido una prioridad en el desarrollo de la misma, la mayoría de elementos utilizados en el *template* se han extraído de *Booststrap* por el ahorro significativo de tiempo que supone. La constante en el proyecto ha sido anteponer siempre la correcta operatividad de cada una de las funcionalidades.

Se podría dar un lavado de cara a la aplicación para que tenga un aspecto más cuidado y atractivo.

5.2.3. Mejorar los modelos predictivos

Se deja abierta la línea de investigación de un modelo de regresión logística con variables categóricas, e incluso un modelo de regresión logística múltiple con variables de ambos tipos y las interacciones entre las mismas.

5.2.4. Base de datos: Aplicar filtros para mostrar los datos

Si el usuario quiere visualizar los datos almacenados siempre se le van a mostrar las mismas columnas. Una mejora importante sería aplicar un filtrado previo para que las consultas sean personalizables.

Capítulo 6

Summary and Conclusions

6.1. Conclusions

A project like this required a broad approach. I had to work with several tools and libraries along with statistic procedures, where the process of learning was constant.

The main goals proposed have been achieved. The web application is functional, one can obtain statistical information about the incoming students' data in computing science, and it helps to analyse the variables are more related among them or if they influence in dropping out. The tool has allowed knowing that the results of RP30 hold no relationship whatsoever with dropping out, and it is not an appropriate indicator to work with this issue.

The developing of this project has brought into light several lessons, such as the discovery of new statistical models and techniques; the assimilation of the functioning of the MVT pattern, understanding each one of its elements; along with knowledge on how to work with new libraries and tools, among others. Furthermore, I have experimented with the whole process of defining a web project since its initial approach until its deployment on the cloud.

6.2. Improvements

I have considered a series of proposals for the improvement of the operation of the tool.

6.2.1. Defining users and roles

One of the main features that this tool might require is a users' authentication system that allows hiding some uses of the project to the non-registered users.

Particularly, the application of the project is defined to upload data that should not be available to any visitor, neither it is appropriate that any user can see the stored data; we are dealing with academic information from real students and it is not desirable for it to be available to anyone who wants to see it.

6.2.2. The application of improvements in the Front-end

The visual aspect of the web application has not been a priority in the developing of the project; most of the elements that have been used in the template have, eventually, been removed from Bootstrap in order to save a significant amount of time; the correct operability of each of the features have been always the most important goal.

However, I could work more on the application for it to have a more attractive look.

6.2.3. Improving the predictive models

A line of research remains open: the research of a logistic regression model with categorical variables and even a multiple logistic regression model.

6.2.4. Database: Applying filters to show data.

If the user wants to see the stored data, they are going to be shown the same columns. An important improvement would be the application of previous filters for customizable inquiries.

Capítulo 7

Presupuesto

En este capítulo se muestra un presupuesto aproximado del proyecto que se ha desarrollado. El periodo considerado para el servicio de alojamiento y para el mantenimiento es de 4 meses.

Tarea	Descripción	Horas	Coste
Diseño de la aplicación web	Planificar la herramienta: documentación, definir los requisitos y realizar el prototipado	20h	12€/h
Desarrollo de la aplicación web	Trabajar en el desarrollo de la herramienta web.	150h	15€/h
Alojamiento	Contratar el servicio de alojamiento de Heroku mediante el plan *Production*		100€
Despliegue	Despliegue de la herramienta en Heroku	5h	12€/h
Mantenimiento	Controlar y corregir posibles fallas en la herramienta	80h	12€/h
Total		223h	3.610€

Tabla 7.1: Presupuesto para el proyecto

Apéndice A

Script para generar las variables estadísticas.

```
/*****************************************************************************
*
* Fichero .h
*
*****************************************************************************
*
* AUTORES
*   Pablo Bethencourt Díaz
*
* FECHA
*   20 de Junio de 2020
*
* DESCRIPCION
*   Función con el script que lee de los ficheros con los datos y genera las
*   variables que se almacenan en la base de datos de la aplicación web.
*****************************************************************************/
def read_csv(f1, f2, f3):
    cabecera = ['Dni', 'Sexo', 'Edad', 'Isla', 'Trabajo', 'Beca', 'Conv.Preinscrip', 'Preferencia',
            'Nota admision', 'Aciertos', 'Errores', 'Total Prueba', 'Abandono']
    Alumnos = []

    data_matriculados = f1.read().decode('UTF-8')
    data_rp30 = f2.read().decode('UTF-8')
    data_abandono = f3.read().decode('UTF-8')
    io_matriculados = io.StringIO(data_matriculados)
    io_rp30 = io.StringIO(data_rp30)
    io_abandono = io.StringIO(data_abandono)
    next(io_matriculados)
```

```
next(io_rp30)
next(io_abandono)

lista_datos = [row for row in csv.reader(io_matriculados)]
lista_rp30 = [row for row in csv.reader(io_rp30)]
lista_abandono = [row for row in csv.reader(io_abandono)]

lista_dni = list(dict.fromkeys([str(lista[2]) for lista in lista_datos]))
lista_cursos = list(dict.fromkeys([lista[18] for lista in lista_datos]))
lista_asignaturas = list(dict.fromkeys([lista[20] for lista in lista_datos]))

# Añadimos nuevos campos a la cabecera:

# Creditos
lista_cursos.sort()
for curso in lista_cursos:
    cabecera.append(f"Créditos matriculados {curso}")
    cabecera.append(f"Créditos presentados {curso}")
    cabecera.append(f"Créditos aprobados {curso}")

# Asignaturas
lista_asignaturas.sort()
lista_asignaturas = [ asignatura.title() for asignatura in lista_asignaturas]
for asignatura in lista_asignaturas:
    cabecera.append(asignatura)

# Media
for curso in lista_cursos:
    cabecera.append(f"Media {curso}")

tam = len(lista_datos)
i = 0
for dni in lista_dni:
    aciertos = None
    fallos = None
    total = None
    for sublist in lista_rp30:
        if sublist[0] == dni:
            if sublist[4] != "":
                aciertos = int(sublist[4])
```

```
        else:
            aciertos = None
        if sublist[5] != "":
            fallos = int(sublist[5])
        else:
            fallos = None
        if aciertos == None and fallos == None:
            total = None
        elif sublist[6] == "":
            total = None
        else:
            total = int(sublist[6])
for sublist in lista_abandono:
    if sublist[2] == dni:
        if sublist[3] == 'N' and sublist[4] == 'N':
            abandono = 'Si'
        else:
            abandono = 'No'
if lista_datos[i][17] != '':
    nota_adm = float(lista_datos[i][17].replace(',','.'))
else:
    nota_adm = None

trabajo = 'No' if lista_datos[i][6] == 'No aplica' or lista_datos[i][6] == 'No consta' else
prefe = int(lista_datos[i][9]) if lista_datos[i][9] != '' else None
alumno = [lista_datos[i][2], lista_datos[i][3], lista_datos[i][4], lista_datos[i][5], trabaj
         prefe, lista_datos[i][11], lista_datos[i][13], nota_adm, aciertos, fallos, total, a

lista_curso1 = []
lista_curso2 = []
contador = 0
while (dni == lista_datos[i][2]):
    curso = lista_datos[i][18]
    creditos_matriculados = 0
    cretidos_presentados = 0
    creditos_aprobados = 0

    while (curso == lista_datos[i][18] and dni == lista_datos[i][2]):
        creditos_matriculados += 6
        asignatura = lista_datos[i][20]
```

```
            notas = []
            while (lista_datos[i][20] == asignatura):
                if lista_datos[i][23] != 'NP':
                    notas.append(float(lista_datos[i][24].replace(',','.')))
                i += 1
                if i == tam: break

            if notas:
                cretidos_presentados += 6
                if max(notas) >= 5.0:
                    creditos_aprobados += 6
                    if curso == lista_cursos[0]:
                        lista_curso1.append([asignatura, max(notas)])
                    else:
                        lista_curso2.append([asignatura, max(notas)])
                else:
                    if curso == lista_cursos[0]:
                        lista_curso1.append([asignatura, notas[0]])
                    else:
                        lista_curso2.append([asignatura, notas[0]])
            else:
                if curso == lista_cursos[0]:
                    lista_curso2.append([asignatura, ' '])
                else:
                    lista_curso2.append([asignatura, ' '])
            if i == tam: break

        contador += 1
        alumno.extend([creditos_matriculados, cretidos_presentados, creditos_aprobados])
        if i == tam: break

    if contador == 1:
        alumno.extend([None, None, None])

    for asignatura in lista_asignaturas:
        nota_f = []
        for asig_c1 in lista_curso1:
            if asig_c1[0].title() == asignatura:
                if asig_c1[1] != ' ':
                    nota_f.append(asig_c1[1])
```

```
        for asig_c2 in lista_curso2:
            if asig_c2[0].title() == asignatura:
                if asig_c2[1] != ' ':
                    nota_f.append(asig_c2[1])
        if nota_f:
            if len(nota_f) == 2:
                alumno.append(nota_f[-1])
            else:
                alumno.append(nota_f[0])
        else:
            alumno.append(None)

    media_notas = []
    for nota in lista_curso1:
        if nota[1] != ' ':
            media_notas.append(nota[1])
    if len(media_notas) != 0:
        media = sum(map(float,media_notas))/float(len(media_notas))
        alumno.append(round(media, 2))
    else:
        alumno.append(None)

    media_notas.clear()
    for nota in lista_curso2:
        if nota[1] != ' ':
            media_notas.append(nota[1])
    if len(media_notas) != 0:
        media = sum(map(float,media_notas))/float(len(media_notas))
        alumno.append(round(media, 2))
    else:
        alumno.append(None)

    Alumnos.append(alumno)

Alumnos.insert(0, cabecera)

return Alumnos
```

Bibliografía

[1] Sunil Kappal. R vs. python. *DZone*, 2018. https://dzone.com/articles/r-or-python-data-scientists-delight.

[2] J. Pérez, F. y Aldás. Indicadores sintéticos de las universidades españolas. *Fundación BBVA*, 2019. https://www.fbbva.es/wp-content/uploads/2019/04/Informe-U-Ranking-FBBVA-Ivie-2019.pdf.

[3] Gabinete de Análisis y Planificación. Alumnado de nuevo ingreso en estudios de grado y primer y segundo ciclo. *Universidad de La Laguna*, Consultado en septiembre 2020. https://riull.ull.es/xmlui/handle/915/3971.

[4] I. Rubio. Las matriculaciones en carreras técnicas bajan pese a la demanda laboral. *El País*, (24 de septiembre de 2019). https://elpais.com/tecnologia/2019/09/24/actualidad/1569332904_298329.html.

[5] F. Fonseca, G. y García. Permanencia y abandono de estudios en estudiantes universitarios: un análisis desde la teoría organizacional. *resu.anuies.mx Revista de la Educación Superior 45(179) (2016) 25–39*, 2016.

[6] Alfonsa García et al. Abandono de primer año en la ingeniería informática. *Actas de las XX JENUI. Oviedo, 9-11 de julio 2014, 151-158*, 2014.

[7] asia. aplicación para el seguimiento institucional del abandono., Consultado en septiembre 2020. http://permanencia.etsisi.upm.es/.

[8] R-project. «r-project.org» [Internet]: https://www.r-project.org/.

[9] Django. «djangoproject.com» [Internet]: https://www.djangoproject.com/.

[10] Bootstrap. «getbootstrap.com» [Internet]: https://getbootstrap.com/.

[11] Sqlite. «sqlite.org» [Internet]: https://www.sqlite.org/index.html.

[12] Postgresql. «postgresql.org» [Internet]: https://www.postgresql.org/.

[13] Numpy. «NumPy.org» [Internet]: https://numpy.org/.

[14] pandas. «pandas.pydata.org» [Internet]: https://pandas.pydata.org/.

[15] matplotlib. «matplotlib.org» [Internet]: https://matplotlib.org/.

[16] seaborn. «seaborn.pydata.org» [Internet]: https://seaborn.pydata.org/.

[17] Scipy. «scipy.org» [Internet]: https://www.scipy.org/.

[18] statsmodels. «statsmodels.org» [Internet]: https://www.statsmodels.org/stable/index.html.

[19] Scikit-learn. «scikit-learn.org» [Internet]: https://scikit-learn.org/stable/.

[20] Heroku. «heroku.com/.» [Internet]: https://www.heroku.com/.

[21] Análisis exploratorio de dato.(s.f.) (En Wikipedia), Consultado en septiembre 2020. Recuperado de https://es.wikipedia.org/wiki/An%C3%A1lisis_exploratorio_de_datos.

[22] María Elvira Ferre Jaén. Feir 45: Regresión logística. *Apuntes del curso FEIR3*, 2014/2015. http://gauss.inf.um.es/feir/45/#226_bondad_de_ajuste_en_r.

[23] Joaquín Amat Rodrigo. Regresión logística simple y múltiple. *RPubs*, 2016. https://rpubs.com/Joaquin_AR/229736.

[24] J. Holovaty, A. y Kaplan-Moss. El libro de django 1.0, Consultado en septiembre 2020. El patrón de diseño MTV. Recuperado de https://uniwebsidad.com/libros/django-1-0/capitulo-5/el-patron-de-diseno-mtv.