Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Ingeniería de Software

Desarrollo de una aplicación para el Análisis de Componentes Principales

Alumno:
Álvaro García Torres

Tutor:
Diego García Álvarez

...

Agradecimientos

Quiero agradecer a mi familia, especialmente a mis padres, y amigos por su cariño y apoyo inquebrantable a lo largo de mis estudios y de toda mi vida.

Gracias también a mi tutor, Diego García Álvarez, por toda su valiosa ayuda y consejos en el desarrollo del trabajo y por su continua disponibilidad y todo el tiempo que me ha dedicado.

Finalmente, quiero agradecer a todos mis profesores del Grado que me han transmitido las bases, los conocimientos y el interés en la Informática.

Resumen

Este trabajo presenta el desarrollo una aplicación de escritorio para trabajar de manera gráfica el Análisis de Componentes Principales (PCA) de los datos almacenados en un fichero. El PCA es una poderosa técnica de reducción de la dimensionalidad y una de las técnicas más conocidas y usadas en Aprendizaje Automático (no supervisado).

La aplicación está pensada para usuarios que saben interpretar este tipo de análisis, pero carecen de conocimientos de programación para generar el modelo PCA y los gráficos asociados. La aplicación permite representar las gráficas más usuales del PCA, facilitando al usuario la visualización de los resultados obtenidos, y proporciona ayuda en la determinación del número de componentes a extraer.

La aplicación ha sido desarrollada en Python y con la librería de interfaces gráficas Qt para facilitar el diseño de la interfaz de usuario.

Abstract

This work involves the development of a desktop application to easily implement Principal Component Analysis (PCA) on data from a file. PCA is a powerful dimensionality reduction technique and one of the most well-known and widely used techniques in (unsupervised) Machine Learning.

The application is designed for users who understand how to interpret this type of analysis but lack programming knowledge to generate the PCA and the associated graphs. The application automatically generates the most common PCA plots, making it easier for users to visualize the results obtained, and provides assistance in determining the number of components to extract.

The application has been developed in Python using the Qt graphical interface library to facilitate the design of the user interface.

Índice general

Glosario de términos

K-fold Procedimiento de validación cruzada basado en particionar los datos disponibles en K hojas o *folds* y usar las hojas alternando el papel de conjunto de entrenamiento y de validación.

Loadings Pesos de la combinación lineal de las variables originales para obtener las componentes principales.

Scores Valores o coordenadas que toman los individuos en cada componente principal.

Scree plot Gráfico con las variabilidades explicadas por cada componente principal.

Siglas

MVC Modelo Vista Controlador.

PCA Principal Components Analysis o Análisis de Componentes Principales.

PLS Partial Least Squares.

Capítulo 1

Introducción

Este documento corresponde a la memoria del Trabajo Fin de Grado del Grado en Ingeniería Informática Mención en Ingeniería de Software de la Escuela de Ingeniería Informática de la Universidad de Valladolid. A lo largo de los capítulos de esta memoria se explicarán las fases por medio de las cuales se ha desarrollado la aplicación implementada en este Trabajo Fin de Grado y se detallará la estructura y las decisiones tomadas para su desarrollo.

1.1. Contexto y motivación

En la actualidad, existen una gran cantidad de diferentes problemas y necesidades que requieren una solución informática. Entre estos diferentes tipos de problemas se encuentran los estadístico/matemáticos, los cuales han evolucionado conjuntamente a la evolución de las tecnologías informáticas a lo largo de los últimos años. En esta evolución, han surgido distintas herramientas que permiten abordarlos y que facilitan el trabajo de las personas que no disponen de los suficientes conocimientos de programación necesarios para elaborar sus propias soluciones.

Por su parte, en el campo de la Informática existen diversas maneras de dar forma a estas soluciones. Se pueden realizar para dispositivos móviles o fijos, pueden disponer de interfaces que se adapten a las distintas necesidades de los usuarios que las emplean, e incluso se puede acceder a ellas desde páginas web.

La elección del medio en el que se despliegan las aplicaciones depende principalmente del problema y de las preferencias de quien diseña la aplicación. Las aplicaciones web por una parte permiten ser utilizadas desde un navegador y la mayor

parte de su funcionalidad se realiza desde el servidor, lo cual resulta conveniente para dispositivos que dispongan de recursos limitados. Sin embargo, las aplicaciones web, a diferencia de las de escritorio, requieren disponer de acceso a red y al servidor que suministra la aplicación, lo cual puede provocar problemas en caso de cortes de red o caídas del servidor.

Debido a esto, las aplicaciones de escritorio resultan una alternativa en aquellos casos en los que el acceso a internet no está asegurado o se requiera cierta seguridad frente a caídas del servicio durante el uso de la aplicación.

Es en este contexto que surge la herramienta creada para este proyecto, una aplicación de escritorio, la cual permitirá a través de gráficas visualizar el resultado del Análisis de Componentes Principales (*Principal Components Analysis*) o PCA para un conjunto de datos disponible y facilitar su correcta interpretación y aplicabilidad.

La reducción de la dimensionalidad es el proceso de transformar datos en dimensiones altas (o moderadamente altas) a una dimensión menor donde su tratamiento sea más simple. Existen numerosas razones para reducir la dimensionalidad de un conjunto de datos. Primeramente, los conjuntos de alta dimensionalidad imponen retos computacionales notables ya que, incluso, su almacenamiento y transmisión puede ser extremadamente costoso o, en ocasiones, imposible. Adicionalmente, el uso en técnicas predictivas suele ser problemático, ya que la presencia de variables redundantes y muy correladas entre sí es una bien conocida dificultad para muchas técnicas predictivas típicamente aplicadas [James et al., 2023, Peña, 2013].

Es importante notar que la reducción de la dimensionalidad suele ser un paso previo en la búsqueda de estructuras significativas en los datos o simplemente para obtener una simple visualización inicial de los mismos. La reducción de la dimensionalidad es también una técnica muy utilizada para monitorizar procesos en Control de Calidad, cuando las variables monitorizadas sean muchas y no sea razonable su tratamiento individualizado [García Álvarez, 2013].

Entre las técnicas para la reducción de la dimensionalidad más habituales e interpretables está, sin duda, la consideración de combinaciones lineales de las variables del conjunto de datos. Este será el enfoque que sigue el PCA. En concreto, el PCA trata de buscar, de entre todas las combinaciones lineales posibles, aquellas que permitan una reducción de la dimensionalidad óptima, en el sentido de alcanzar la mínima perdida de información posible. Esto se consigue transformando las variables originales, inicialmente correladas, en combinaciones lineales incorreladas y ordenadas de mayor a menor grado de información. Esta técnica es debida a Hotelling [1933] aunque sus orígenes aparecen en la regresión mínimo-cuadrática con residuales ortogonales introducida en Pearson [1901].

EL PCA es una de las técnicas básica del “aprendizaje no-supervisado”. El objeti-

vo de los métodos no-supervisados es buscar patrones de interés o variables "latentes" subyacentes en los datos, sin poder contar con información externa que nos pueda guiar en este objetivo. En contraposición, en los métodos supervisados se suele contar con una variable respuesta a predecir y el objetivo es aprovechar la información en un "conjunto de aprendizaje", para el que sí se conoce el valor de esa variable respuesta, para predecir la respuesta de nuevos individuos en los que dicha respuesta sea desconocida. Este enfoque supervisado es el que se sigue, por ejemplo, en las técnicas de regresión y de clasificación. Sin embargo, en los métodos de aprendizaje no-supervisado no existen valores de la variable respuesta disponibles en un conjunto de entrenamiento. Simplemente se cuenta con la información proporcionada por variables medidas en los individuos y se desea extraer de los datos todo el "conocimiento" útil que sea posible. De hecho, los métodos no supervisados son en muchos casos un paso previo a la utilización de técnicas supervisadas.

El uso de la aplicación desarrollada en este Trabajo Fin de Grado facilitará la visualización del PCA y aspectos como la adecuada selección del número de componentes principales a retener para un conjunto de datos concreto.

1.2. Objetivos

1.2.1. Objetivos de desarrollo

El principal objetivo de este proyecto es el de desarrollar una aplicación de escritorio para aplicar e interpretar el PCA que sea fácil de usar y que disponga de una interfaz de usuario sencilla. Con este fin, la aplicación ha sido desarrollada con los siguientes sub-objetivos de funcionamiento:

- Permitir a los usuarios visualizar y modificar la tabla los datos del fichero sobre el que se quiera realizar el análisis.
- Ofrecer diversas opciones de personalización del PCA tales como el número de componentes principales o la selección de una columna para etiquetar los datos.
- Mostrar diferentes gráficas relacionadas con el PCA, comparando en aquellas que lo requieran las componentes principales que seleccione el usuario.
- Documentar el proceso de desarrollo de software de la aplicación.

1.2.2. Objetivos académicos

Debido a que esta aplicación se trata de un proyecto realizado como Trabajo de Fin de Grado, existen otra serie de objetivos académicos que han sido propuestos:

- Aprender a utilizar la librería de desarrollo de interfaces gráficas Qt.
- Aplicar los conocimientos aprendidos en las distintas asignaturas del Grado en Ingeniería Informática a un proyecto de desarrollo de software.
- Afrontar un problema que requiere formarse en un campo diferente al de la Informática como es el PCA con bases estadístico-matemáticas no vistas en la titulación cursada.
- Mejorar en el desarrollo de aplicaciones con el lenguaje de programación Python.
- Aplicar los diferentes *frameworks* típicos de la Ingeniería del Software como el "Análisis" o el "Diseño".

1.3. Estructura de la memoria

Este documento se estructura de la siguiente forma:

Capítulo 2 - Fundamentos del PCA: Se comentarán en este capítulo los fundamentos matemáticos del PCA y los conceptos básicos de esta metodología.

Capítulo 3 - Planificación: Se detalla en este capítulo la planificación que se ha seguido para el desarrollo de la aplicación.

Capítulo 4 - Requisitos: En este capítulo se detallan los requisitos de la aplicación.

Capítulo 5 - Análisis: En este capítulo se expone el modelo de dominio y de análisis, así como la realización de los casos de uso del sistema.

Capítulo 6 - Diseño: En este capítulo se muestra el diseño que se ha adoptado en la aplicación y bocetos realizados de la interfaz de usuario.

Capítulo 7 - Implementación: Se comentarán las tecnologías utilizadas, la organización del código junto a un resumen de la implementación final y algunos ejemplos de uso de la misma

Capítulo 8 - Pruebas: Se proporcionan en este capítulo las pruebas por las que ha pasado la aplicación.

Capítulo 9 - Conclusiones y líneas de trabajo futuras: En este capítulo se comentan las principales conclusiones a las que se han llegado tras la realización de este proyecto y algunas posibles líneas futuras de trabajo.

Capítulo 2

Fundamentos del PCA

Dado que este Trabajo Fin de Grado se basa en el desarrollo de una aplicación para el uso del Análisis de Componentes Principales o (en inglés) *Principal Components Analysis* (PCA), se describirá brevemente las bases matemáticas que subyacen en esta metodología PCA en este capítulo. Así, se justificará el interés del PCA y se introducen algunos conceptos básicos que aparecerán de forma recurrente en esta memoria. Más detalles sobre como interpretar algunas salidas asociadas al PCA serán vistas en el Capítulo 7, mediante una ilustración de su uso al ser aplicado el PCA en conjuntos de datos reales.

Para llevar a cabo el PCA se parte de una matriz de datos $\mathbf{X}$, donde cada fila de esta matriz corresponderá a un individuo o caso (que indexamos en $i = 1, ..., I$) y cada columna será una variable que es medida en cada uno de los individuos (que indexamos en $j = 1, ..., J$). Así, se parte de I individuos a los que se miden J variables y se denota por x_{ij} al elemento en la posición (i, j) de dicha matriz, de tal forma que x_{ij} sea el valor que toma el individuo i en la variable j, dando lugar a una matriz $I \times J$.

Al usar PCA es común trabajar con variables centradas y estandarizadas. Esto quiere decir que las medias de las columnas de la matriz $\mathbf{X}$ son todas iguales a 0 y sus desviaciones típicas por columnas iguales a 1. Si los datos no verifican directamente estas condiciones de centrado y escalado (lo que a nivel práctico suele ser el caso más habitual) entonces se deben calcular las medias por columnas

$$\overline{x}_j = \frac{1}{I} \sum_{i=1}^{I} x_{ij} \tag{2.1}$$

y las desviaciones típicas

$$S_j = \sqrt{\frac{1}{I}\sum_{i=1}^{I}(x_{ij} - \overline{x}_j)^2}, \tag{2.2}$$

y reemplazar los valores en x_{ij} por su versión centrada y escalada mediante

$$\frac{x_{ij} - \overline{x}_j}{S_j}.$$

Esta transformación de los datos es útil al poner las J variables en pie de igualdad, de tal forma que el resultado del PCA no esté distorsionado por las escalas o unidades de medida particulares en las variables consideradas.

EL PCA es una de las técnicas más comúnmente utilizada para analizar conjuntos de datos. El material que se presenta a continuación puede ser encontrado en muchos manuales de Análisis de Datos Multivariante y de Aprendizaje Automático (*Machine Learning*). En la elaboración del material en esta memoria se ha utilizado especialmente Flury and Riedwyl [1988], Aluja-Banet and Morineau [1999], James et al. [2023], Peña [2013], y para la parte relativa a su implementación en Python se ha seguido Géron [2022] junto a James et al. [2023]. Una presentación más detallada y extensa del PCA puede encontrarse en Jolliffe [2002].

El PCA trata de reducir la dimensionalidad de la matriz de datos $\mathbf{X}$, pero perdiendo la mínima "información" posible. La forma más sencilla de reducir la dimensionalidad es considerar combinaciones lineales de las columnas de la matriz. Una forma matemáticamente correcta y simplificada de escribir las combinaciones lineales de las columnas de una matriz $\mathbf{X}$ es multiplicar por un vector $\mathbf{p}$ en dimensión J. Para

$$\mathbf{p} = (p_1, p_2, ..., p_J)^t \in \mathbb{R}^J$$

se tiene que

$$\mathbf{Xp} = p_1\mathbf{x}_1 + p_2\mathbf{x}_2 + ... + p_J\mathbf{x}_J \in \mathbb{R},$$

es una combinación lineal de las variables originales $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_J$. Como trabajamos con una matriz de datos centrada por columnas, se tiene que la media muestral del vector $\mathbf{Xp}$ es también igual a 0 y la forma de perder la menor la menor información posible, a nivel matemático, se consigue cuando la varianza muestral de esa combinación lineal $\mathbf{Xp}$ es lo mayor posible. Notesé que un caso límite surge cuando la varianza de $\mathbf{Xp}$ sea exactamente 0, ya que esa situación implicaría que todos los valores de la combinación lineal $\mathbf{Xp}$ son iguales a 0 y la combinación lineal no proporcionaría absolutamente ninguna información.

A nivel matemático, el objetivo del PCA es maximizar la varianza muestral $\text{Var}(\mathbf{Xp})$ pero con la restricción añadida de que la norma de ese vector $\mathbf{p}$ sea igual

a 1, es decir, $\|\mathbf{p}\| = 1$. Esta última restricción surge del hecho de que multiplicar las combinaciones lineales por constantes fijas no debe cambiar la interpretación que se debe dar a dichas combinaciones lineales. Tomar vectores $\mathbf{p}$ con norma 1 además hace que se pueda ver (salvo un signo) la combinación lineal como la proyección ortogonal en la dirección dada por el vector $\mathbf{p}$ (Peña [2013]).

La matriz

$$\frac{1}{I}\mathbf{X}^t\mathbf{X}$$

de tamaño $J \times J$ (donde $\mathbf{A}^t$ denota la traspuesta, intercambiando el papel de filas y columnas, de una matriz $\mathbf{A}$) coincide para datos centrados con la matriz de varianzas-covarianzas muestral (contiene las varianzas de las variables en la diagonal y sus covarianzas fuera de la diagonal). De hecho, esta matriz coincide (salvo una constante) con la matriz de correlaciones cuando las columnas de la matriz $\mathbf{X}$ estén estandarizadas.

Se puede ver que la varianza de la combinación lineal satisface

$$\text{Var}(\mathbf{Xp}) = \frac{1}{I}\mathbf{p}^t\mathbf{X}^t\mathbf{Xp}.$$

Si se elimina la constante $1/I$, el maximizar $\mathbf{p}^t\mathbf{X}^t\mathbf{Xp}$ con la restricción $\|\mathbf{p}\| = 1$ se alcanza al elegir un vector $\mathbf{p}$ que sea autovector de la matriz $\mathbf{X}^t\mathbf{X}$ (es decir, $\mathbf{X}^t\mathbf{Xp} = \lambda\mathbf{p}$ para una constante o autovalor λ).

Adicionalmente, $\mathbf{p}$ no es un autovector cualquiera porque se verifica

$$\mathbf{p}^t\mathbf{X}^t\mathbf{Xp} = \lambda\mathbf{p}^t\mathbf{p} = \lambda\|\mathbf{p}\|^2 = \lambda \cdot 1 = \lambda,$$

y $\mathbf{p}$ debe ser un autovector de norma 1 asociado al autovalor mayor de la matriz $\mathbf{X}^t\mathbf{X}$. Este autovector $\mathbf{p}$ proporcionaría los pesos de la combinación lineal pasando de "forma óptima" de dimensión J a dimensión 1 y los valores de dicha combinación lineal vienen dados por el vector $\mathbf{t} = \mathbf{Xp}$.

Esta reducción de los datos a dimensión 1 puede ser demasiado extrema y quizás se prefiera buscar A combinaciones lineales extrayendo la mayor información posible. La solución a este problema se obtiene de los A autovectores $\mathbf{p}_1$, $\mathbf{p}_2$, y $\mathbf{p}_A$ de la matriz $\mathbf{X}^t\mathbf{X}$, todos con norma igual a 1, y asociados a los autovalores

$$\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_A$$

mayores de $\mathbf{X}^t\mathbf{X}$.

Se tiene que $\mathbf{X}^t\mathbf{X}$ será una matriz $J \times J$ que, en principio, puede tener hasta J autovectores, luego A debe ser obviamente menor que J. No obstante, las componentes asociadas a autovalores iguales a 0 no resultan informativas porque corresponden

a combinaciones lineales de las columnas de la matriz $\mathbf{X}$ que son exactamente iguales a un vector con todos sus componentes iguales a 0.

Las A combinaciones lineales óptimas obtenidas son $\mathbf{t}_1 = \mathbf{X}\mathbf{p}_1$, $\mathbf{t}_2 = \mathbf{X}\mathbf{p}_2$ y $\mathbf{t}_A = \mathbf{X}\mathbf{p}_A$ son conocidas comúnmente como *scores*. Dichos scores puede obtenerse matricialmente como

$$\mathbf{T}_{1:A} = (\mathbf{t}_1|\mathbf{t}_2|...|\mathbf{t}_A) = (\mathbf{X}\mathbf{p}_1|\mathbf{X}\mathbf{p}_2|...|\mathbf{X}\mathbf{p}_A) = \mathbf{X} \cdot \mathbf{P}_{1:A}.$$

La matriz $I \times A$ dada por $\mathbf{T}_{1:A}$ se conoce como matriz de *scores* y cada una de sus filas contendrá los valores (coordenadas o puntuaciones) de las A combinaciones lineales para cada uno de los I individuos del conjunto de datos.

Por otro lado, la matriz $J \times A$ dada por $\mathbf{P}_{1:A}$ es conocida como matriz de *loadings* y sus columnas son los pesos (cargas o loadings) de las A combinaciones lineales óptimas.

Se puede probar que las columnas de la matriz $\mathbf{P}_{1:A}$ son ortogonales (Peña [2013]). Es decir, el producto escalar de vectores $\mathbf{p}_j$ y $\mathbf{p}_{j'}$, denotado por $\langle \mathbf{p}_j, \mathbf{p}_{j'} \rangle$, es 0.

A nivel estadístico se han encontrado unas nuevas variables $\mathbf{t}_1$, $\mathbf{t}_2$, ..., y $\mathbf{t}_A$, combinaciones lineales de las variables originales $\mathbf{x}_1$, $\mathbf{x}_2$, ..., y $\mathbf{x}_J$, que verifican

$$\text{Var}(\mathbf{t}_1) \geq \text{Var}(\mathbf{t}_2) \geq ... \geq \text{Var}(\mathbf{t}_A).$$

Es decir, estas nuevas variables están ordenadas de más a menos informativas. Por otro lado, sus covarianzas satisfacen

$$\text{cov}(\mathbf{t}_a, \mathbf{t}_{a'}) = 0 \text{ para } a \neq a',$$

y, así, las nuevas variables no contienen información duplicada (se tiene incorrelación entre $\mathbf{t}_a$ y $\mathbf{t}_{a'}$).

La combinación de los comentarios anteriores hace que el PCA proporcione una forma notablemente eficiente de resumir la información existente en un conjunto de datos.

Se puede transponer la matriz de datos en la matriz $\mathbf{X}^t$ y tener así J filas en dimensión I. Al aplicar PCA a estos "nuevos" datos, ese análisis complementario basado en autovectores de la matriz $I \times I$ dada por $\mathbf{X}\mathbf{X}^t$, se podría representar también de forma óptima las J variables. Los dos análisis se pueden superponer y obtener representaciones gráficas tipo *biplot* que permiten gráficar el comportamiento de individuos y variables simultameamente. De hecho, los dos problemas complementarios están relacionados y tienen que ver con la conocida como descomposición en valores singulares de la matriz $\mathbf{X}$.

Cuando más grande es el autovalor λ_a mayor es la variabilidad que esa componente principal a recoge, puesto que $\text{Var}(\mathbf{t}_a) = \lambda_a$. Este es el tipo de información que se suele usar para elegir el número de componentes A al utilizar los gráficos conocidos como *scree plots*.

Otra de las aplicaciones del PCA es obtener una aproximación de la matriz $\mathbf{X}$, descartando la información en las componentes principales asociadas a los autovalores mas pequeños. Si los autovalores de $\mathbf{X}^t\mathbf{X}$ satisfacen

$$\frac{\lambda_1 + ... + \lambda_A}{\lambda_1 + ... + \lambda_J} \approx 1,$$

entonces se tiene que

$$\widehat{\mathbf{X}}_{1:A} = \mathbf{T}_{1:A} \cdot (\mathbf{P}_{1:A})^t \approx \mathbf{X}.$$

Es decir, en ese caso de que tengamos autovalores menores próximos a 0, entonces $\widehat{\mathbf{X}}_{1:A}$ resulta una muy buena aproximación de la matriz de datos original $\mathbf{X}$. Esta aproximación puede proporcionar una representación aproximada de $\mathbf{X}$ bastante económica cuando A sea notablemente menor que J, puesto que se precisan almacenar $I \times A + J \times A$ valores, en lugar de los $I \times J$ iniciales.

Una crítica común al PCA es su subjetividad en la elección del número de componentes A [Jolliffe, 2002, James et al., 2023]. No obstante, esto es esperable pues el método PCA es un método de aprendizaje "no supervisado". Alternativamente, en métodos "supervisados" existe una (o varias) variable(s) respuesta(s) a predecir. Los métodos supervisados cuentan con "conjuntos de entrenamiento" para los que se conocen los valores de las variables respuestas, y que son usados para "aprender" reglas de predicción y que son posteriormente utilizadas para predecir los valores de la(s) variable(s) respuesta(s) de nuevos individuos. En este entorno de aprendizaje supervisado es muy común utilizar procedimientos de "validación-cruzada" para seleccionar las reglas de predicción más razonables. La validación-cruzada parte de un conjunto de entrenamiento o "train" y un conjunto de prueba o "test". Con estos conjuntos se analiza qué tal funciona cada regla de predicción entrenadas con el conjunto de entrenamiento al predecir la respuesta de los individuos en conjunto de prueba. La idea clave que el ajuste de la regla no pueda usar los valores de las respuestas del conjunto de test y así evite de forma natural un posible "sobreajuste" de la regla de predicción. El sobreajuste surge cuando la regla se adapta "demasiado" bien al conjunto de entrenamiento, de tal forma que funciona muy bien para este conjunto de entrenamiento, pero podría no funcionar tan bien para predecir lo que sucede para nuevos individuos por su fuerte dependencia en dicho conjunto de entrenamiento. Existen diferentes posibilidad de realizar la validación cruzada, entre las que destaca el método de K-fold y el "leave-one-out".

Como se ha comentado, en PCA no existen variables respuesta a predecir, por este carácter no-supervisado de la técnica. Esto hace que no sea directo aplicar técnicas

de validación cruzada para elegir A salvo que el PCA pueda ser visto como un paso intermedio para predecir alguna variable respuesta.

Existen varios intentos de aplicar los principios de la validación cruzada a elegir el número de componentes A en PCA. Ninguno de los métodos propuestos parece considerarse como universalmente aceptado [Jolliffe, 2002, Josse and Husson, 2012]. De hecho, se reconoce que esta elección es fuertemente dependiente del objetivo final del PCA. Así, es raro seleccionar dimensiones mucho mayores que 3 si el objetivo es usar el PCA para la visualización de datos, puesto que dimensión 3 es la dimensión máxima que el ser humano está bien capacitado a visualizar. Por otro lado, cuando el objetivo del PCA sea reducir la dimensionalidad del problema, por ejemplo, para su almacenamiento en memoria o la aplicación de otras técnicas que requieran dicha reducción de la dimensionalidad, entonces, se puede pensar en elegir valores de A mayores, explicando la suficientemente variabilidad aunque menores que la dimensionalidad original J. Las capacidades de almacenamiento, para esta versión "comprimida" de los datos, puede condicionar severamente el valor A a elegir.

Reconociendo esta dificultad en usar la validación cruzada para elegir A en PCA, con el fin de ayudar al usuario en la elección del número de componentes, se detallará en la Sección 7.4.5 un procedimiento basado en esta filosofía.

Justificaciones rigurosas de todo lo comentado en este capítulo pueden encontrarse en los manuales y referencias citadas. A nivel histórico, como ya ha sido comentado, esta técnica es atribuida a Hotelling [Hotelling, 1933] aunque sus orígenes ya aparecían en la regresión mínimo-cuadrática con residuales ortogonales propuesta en Pearson [Pearson, 1901]. Su máximo desarrollo se ha debido a avances computacionales e informáticos que actualmente permiten su rápida implementación, incluso en casos en los que el número de individuos I o de variables J sean elevados.

Capítulo 3

Planificación

En este capítulo se describirá la planificación que se ha seguido para la realización del proyecto y como esta se ha llevado a cabo. También se presentará la metodología seguida y se realizará un análisis del presupuesto y los riesgos del desarrollo de la aplicación.

3.1. Metodología *waterfall*

El modelo *waterfall* o modelo en cascada es una metodología de gestión de proyectos en la cual el proyecto es dividido en tareas, las cuales se van realizando de manera secuencial. En esta metodología, una tarea empieza cuando termina la anterior, y una vez terminada una tarea no se debería volver a tareas terminadas.

Esta metodología surgió en los sectores de la fabricación y construcción, y fue descrita formalmente para el sector del desarrollo software por Winston W. Royce en 1970 [Royce, 1970]. En esta definición, los proyectos se dividen en las distintas etapas principales del desarrollo de software, que son

- Requisitos.
- Análisis.
- Diseño.
- Implementación.
- Pruebas.

- Mantenimiento

La etapa de requisitos en esta metodología debe ser más rigurosa que en otro tipo de metodologías, ya que por definición no se puede volver a esta fase una vez se avance a las siguientes tareas. Por lo tanto para poder aplicar el modelo *waterfall* todos los requisitos del proyecto se deben poder definir en esta primera fase.

En las fases de análisis y diseño, a veces combinadas en una única tarea, se definen la arquitectura del sistema y los detalles más específicos del software que se desarrolla.

La etapa de implementación es en la que se elabora el código del proyecto. Para esta tarea, se implementa la funcionalidad del sistema siguiendo los pasos elaborados en las fases anteriores.

Durante la etapa de pruebas, el principal objetivo es asegurarse que el proyecto se ha desarrollado correctamente y que cumple las especificaciones de los clientes. En muchos proyectos, esta etapa de pruebas es ignorada, ya que una vez entregado el código a los usuarios cualquier situación que pueda producirse se tratará en la fase de mantenimiento.

Por último, en esta fase de mantenimiento se termina de pulir el sistema, pues cuando los clientes lo utilicen, se podrán detectar problemas derivados de las etapas anteriores o añadir aquellos cambios que no fueron contemplados correctamente durante la fase de requisitos. Debido a la naturaleza de este proyecto, no se dispondrá de fase de mantenimiento.

Este modelo, presenta algunas ventajas con respecto a los marcos de trabajo "ágiles" [Pargaonkar, 2023]:

- El enfoque para abordar el desarrollo es más directo al estar divido en fases claramente diferenciadas.
- La rigidez de la metodología, permite observar de forma sencilla la evolución del proyecto y la etapa en la que se encuentra a lo largo del tiempo.
- Simplifican aquellos proyectos con requisitos claros y fáciles de fijar.
- Los errores en las fases de diseño y análisis se detectan en su fase y por lo tanto no afectan a la etapa en la que se desarrolla el código.
- La duración y los costes del proyecto se pueden estimar más fácilmente.

Sin embargo, también sufre algunas desventajas:

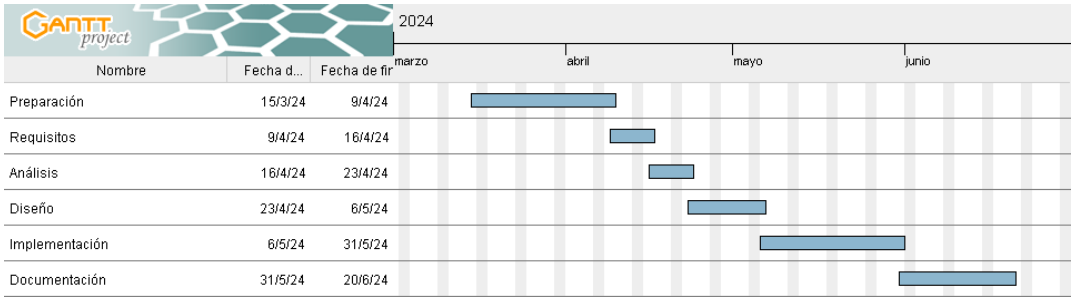


Figura 3.1: Diagrama de Gantt de la planificación del proyecto

- Cualquier retraso en alguna de las fases iniciales pospone el comienzo de las fases posteriores, produciendo un mayor impacto en la duración del proyecto.
- Es una metodología poco tolerante con respecto a cambios en los requisitos, al no poder retroceder en las fases.
- Dado que no se realiza revisión hasta que finaliza la etapa de implementación y se entrega el código a los clientes, cualquier cambio que necesite el código provocara una mayor carga en la tarea de mantenimiento.

La visualización de esta metodología se puede realizar fácilmente por medio de diagramas de Gantt, los cuales permiten mostrar mediante una gráfica de barras horizontal las tareas y su duración a lo largo del eje del tiempo.

3.2. Adaptación de la metodología *waterfall* al proyecto

Debido al ámbito académico en el que se desarrolla esta aplicación y a que los requisitos han estado claramente definidos desde el comienzo de este, se ha optado por seguir la metodología *waterfall*. Sin embargo, al tratarse de un proyecto de Trabajo de Fin de Grado, ha sido necesario modificar este modelo para adaptarse a este contexto. Debido a esto, si bien se han tratado de seguir las etapas originales de la metodología, ha sido necesario modificar el orden de estas, así como añadir alguna tarea inicial de documentarse y prepararse para el proyecto. De esta forma, se ha realizado el diagrama de Gantt de la Figura 3.1 como representación de la planificación inicial.

También se han estado realizando reuniones de seguimiento semanales con el tutor, por lo tanto si que ha habido retroalimentación en todas las etapas. Esto además ha provocado tener que reducir la rigidez entre las fases, permitiendo volver a una tarea anterior para poder realizar las modificaciones que sean necesarias o avanzar partes de alguna tarea sin haber finalizado la anterior.

3.3. Riesgos

En los proyectos de desarrollo de software es necesario realizar un correcto plan de riesgos para poder identificar las posibles situaciones que puedan afectar al desarrollo del proyecto, así como permitir elaborar una serie de medidas en caso de que se produzca alguno de estos riesgos.

Dado que se trata de un análisis elaborado en la fase de planificación de un proyecto, es imposible determinar con exactitud los problemas que van a surgir durante el desarrollo. Debido a esto, el objetivo del análisis de riesgos será predecir y definir todos aquellos riesgos a los que se pueda enfrentar el proyecto, y asignarles un nivel de probabilidad e impacto en función del contexto en el que se realice.

Por último, puesto que los riesgos se tratan de obstáculos que afectan al correcto desarrollo del proyecto, será necesario realizar planes de mitigación para reducir el impacto que puedan tener estos riesgos, así como planes de contingencia para poder afrontarlos en caso de que ocurran.

Las Tablas 3.1 a 3.7 muestran los riesgos que han sido analizados para este proyecto.

3.4. Presupuesto

Otro apartado que no puede faltar en un documento de planificación es el presupuesto del proyecto. Dado que se realiza antes de comenzar el desarrollo, se trata de una estimación calculada a partir de los costes medios de los equipos y los sueldos de los trabajadores con la duración también estimada del proyecto.

Sin embargo, dado que se trata de un proyecto realizado como Trabajo de Fin de Grado en un ámbito académico, el presupuesto real no coincidirá con la estimación realizada. Debido a esto se realizarán ambos análisis en diferentes apartados

3.4.1. Presupuesto simulado

Para este primer caso, se supondrá que el desarrollo del proyecto se realizará en el entorno de una empresa privada dedicada a consultoría. En este contexto es necesario aplicar costes de personal y de los equipos que utilizan para el desarrollo, así como las licencias de los programas que se vayan a utilizar en caso de ser requeridos. Teniendo esto en cuenta, a continuación se detallarán estos costes medios.

Riesgo R-01	
Riesgo	Conocimientos insuficientes de las herramientas y procedimientos utilizados.
Descripción	El alumno no dispone de los conocimientos necesarios para poder utilizar las herramientas necesarias para el desarrollo de la aplicación.
Probabilidad	Media
Impacto	Medio
Plan de mitigación	▪ Realizar una tarea de documentación sobre las herramientas previa al comienzo del desarrollo. ▪ Fijar otra tecnolgía para el desarrollo del proyecto.
Plan de contingencia	▪ Documentarse sobre las funciones de las herramientas utilizadas a medida que se necesiten estas. ▪ Consultar con expertos en el uso de la herramienta. ▪ Solicitar cambiar el proyecto a otra tecnología con la que el equipo este más familiarizado.

Tabla 3.1: Riesgo R-01 - Conocimientos insuficientes.

Riesgo R-02	
Riesgo	Fallo de los equipos informáticos utilizados.
Descripción	El equipo con el que se realiza el proyecto sufre algún tipo de fallo que impide su uso temporalmente.
Probabilidad	Baja
Impacto	Alto
Plan de mitigación	▪ Utilizar dispositivos de poca antiguedad para el desarrollo del proyecto. ▪ Utilizar control de versiones que permita almacenar el código en un servidor remoto.
Plan de contingencia	▪ Reparar el dispositivo estropeado. ▪ Recuperar los archivos necesarios que se puedan perder por la avería. ▪ Utilizar otro dispositivo para continuar el desarrollo.

Tabla 3.2: Riesgo R-02 - Fallo de los equipos informáticos.

Riesgo R-03	
Riesgo	Finalización incorrecta de las tareas.
Descripción	Durante el desarrollo del proyecto alguna de las tareas no se ha realizado correctamente.
Probabilidad	Media
Impacto	Bajo
Plan de mitigación	▪ Definir correctamente los requisitos del sistema. ▪ Realizar revisiones periódicas para detectar los problemas lo antes posible.
Plan de contingencia	▪ Corregir los problemas o realizar las modificaciones que sean necesarias. ▪ Definir de nuevo y con más detalle los requisitos para evitar más fallos.

Tabla 3.3: Riesgo R-03 - Finalización incorrecta de las tareas.

Riesgo R-04	
Riesgo	Interrupción por enfermedad del alumno.
Descripción	Debido a algún problema de salud del desarrollador, el proyecto es interrumpido .
Probabilidad	Media
Impacto	Bajo
Plan de mitigación	▪ Tomar precauciones a la hora de ir a sitios con un número elevado de personas. ▪ Llevar hábitos de vida saludables.
Plan de contingencia	▪ Consultar con un médico la enfermedad para conocer el tratamiento más adecuado. ▪ Reducir la carga de trabajo en lugar de detenerla por completo si la situación lo permite.

Tabla 3.4: Riesgo R-04 - Interrupción por enfermedad del desarrollador.

Riesgo R-05	
Riesgo	Interrupción por enfermedad del tutor.
Descripción	Debido a algún problema de salud del tutor, el proyecto o la revisión de este son interrumpidos.
Probabilidad	Media
Impacto	Bajo
Plan de mitigación	▪ Fijar todas las tareas en las reuniones iniciales. ▪ Establecer vías de comunicación distintas a las reuniones presenciales.
Plan de contingencia	▪ Avanzar aquellas tareas que no requieran ser consultadas inmediatamente con el tutor. ▪ Contactar de manera *online* con el tutor en caso de que sea necesario y que el problema de salud lo permita.

Tabla 3.5: Riesgo R-05 - Interrupción por enfermedad del tutor.

Riesgo R-06	
Riesgo	Problemas con la entrega del proyecto.
Descripción	Debido a algún problema con la plataforma no se puede entregar correctamente el proyecto.
Probabilidad	Baja
Impacto	Bajo
Plan de mitigación	▪ Realizar la entrega con un margen suficiente. ▪ Comprobar el estado de la red antes de realizar la entrega.
Plan de contingencia	▪ Intentar realizar de nuevo la entrega pasado un tiempo si los plazos lo permiten. ▪ Contactar con el servicio de ayuda de la plataforma si el problema persiste.

Tabla 3.6: Riesgo R-06 - Problemas con la entrega del proyecto.

Riesgo R-07	
Riesgo	Incorrecta planificación de la duración del desarrollo.
Descripción	Debido a que se han estimado incorrectamente la duración de alguna de las tareas o del proyecto en general, es posible que no se pueda llegar a la fecha límite de entrega.
Probabilidad	Media
Impacto	Alto
Plan de mitigación	▪ Definir una correcta planificación del tiempo para cumplir las tareas. ▪ Comenzar el desarrollo con tiempo suficiente para finalizar el proyecto en las fechas marcadas. ▪ Aún si se dispone de una segunda convocatoria, realizar la planificación con objetivo de finalizar en la primera.
Plan de contingencia	▪ Aumentar si es posible la carga de trabajo para poder finalizar antes de la fecha límite. ▪ Consultar posibles soluciones si no se ha llegado a ninguna de las fechas límite. ▪ Preparar el proyecto para poder finalizarlo en el próximo curso académico.

Tabla 3.7: Riesgo R-07 - Incorrecta planificación de la duración del desarrollo.

El sueldo medio de un programador en España en el año 2024 es de 2.375€ al mes (https://es.talent.com/salary?job=programador), lo que equivale a 14,62€ la hora. Si bien este valor dependerá de varios factores, como la experiencia y especialidad del trabajador o la ubicación de la empresa, se tomará este salario medio para realizar el cálculo de los costes de este proyecto. Para este cálculo también se considerará que los programadores cobrarán por las horas trabajadas. Siendo un proyecto individual y con una duración fijada previamente en 300 horas, el coste total del programador para este proyecto asciende a 4.386€.

Otro elemento que se debe tener en cuenta a la hora de realizar el presupuesto de un proyecto es la amortización de los equipos. En este caso, solo se dispone de un trabajador, que utiliza un portátil Toshiba Portégé Z930, el cual tiene un precio en la actualidad de 165€. A pesar de ser un dispositivo algo antiguo y que ya fue amortizado hace tiempo, el cálculo de la amortización se realizará suponiendo que se ha comprado por el valor actual. Teniendo en cuenta la antigüedad y precio de este equipo se supondrá que el tiempo de amortización es de 2 años, lo cual dará una amortización de 6,88€ al mes. Dado que el proyecto tendrá una duración aproximada de 4 meses, el coste final de la amortización será de 27,52€.

Por último, será necesario añadir al presupuesto los costes de todas las licencias de los programas que las requieran. En este proyecto, serán necesarias las licencias de Qt, Astah y Balsamiq. Qt ofrece distintas licencias en función del tamaño de la empresa, debido a las dimensiones de este proyecto se seleccionará la versión para pequeños negocios, el cual dado que se compra en planes anuales tendrá un coste de 499€. Para la licencia de Astah, al solo tratarse de un trabajador, se adquirirá la licencia individual, la cual tiene un coste de 11,99€ al mes y supondrá por tanto un gasto final de 47,96€. Por ultimo, Balsamiq ofrece dos tipos de licencia, una temporal con un coste de 9$ (8,43€) cada mes y que ofrece acceso a la versión Cloud alojada en la web, y una permanente que ofrece la versión de escritorio y que solo requiere un único pago de 129$ (120,88€). Puesto que esta herramienta solo se utilizará para diseñar una versión inicial de la interfaz de usuario, lo cual es una tarea que no llevará más de un mes, y dado que se dispone de una conexión a internet estable se adquirirá la licencia temporal con un único pago mensual de 9$ o 8,43€.

En la Tabla 3.8 se muestra un resumen del presupuesto simulado.

3.4.2. Presupuesto real

Para el presupuesto real sin embargo, puesto que se trata de un proyecto académico de la asignatura Trabajo de Fin de Grado, el alumno que desarrolla el proyecto no tendrá remuneración. Por lo tanto del presupuesto simulado no se aplicarán los costes de salario del trabajador.

Presupuesto simulado			
Elemento	**Coste**	**Duración**	**Coste Final**
Salario del desarrollador	14,62€/hora	300 horas	4.386€
Amortización del equipo	6,88€/mes	4 meses	27,52€
Licencia de Qt	499€/año	1 año	499€
Licencia de Astah	11,99€/mes	4 meses	47,96€
Licencia de Balsamiq	8,37€/mes	1 mes	8,37€
Total			**4968,85€**

Tabla 3.8: Resumen del presupuesto simulado.

Con respecto a la amortización del equipo, el dispositivo desde el que se realizó la práctica tiene una antigüedad de más de 10 años, por lo cual ya ha superado cualquier vida útil con la que se hubiera realizado el cálculo de la amortización.

Por último, para las herramientas utilizadas en el proyecto se han utilizado las licencias de código abierto disponibles o aquellas que proporciona la Escuela, por lo cual tampoco han supuesto ningún coste para el proyecto.

Si no contamos otros costes adicionales como la electricidad o la red, la realización del proyecto no ha supuesto ningún coste.

3.5. Seguimiento

Para finalizar este capítulo de Planificación, se ha decidido añadir un ultimo apartado en el que se describirá el seguimiento final del proyecto, es decir, como ha quedado el desarrollo del proyecto con respecto a la planificación inicial.

El primer cambio con respecto a la planificación inicial se ha producido en la tarea de implementación, y es que esta se ha alargado más de lo esperado, con una duración añadida de más de dos semanas. Debido a esto ha sido necesario realizar algunas de las fases al mismo tiempo que se realizaba la de implementación. Por lo tanto también se ha producido un cambio de fechas en el resto de las tareas.

Debido a esto, la organización final de las tareas queda representada en el diagrama de Gantt de la Figura 3.2.

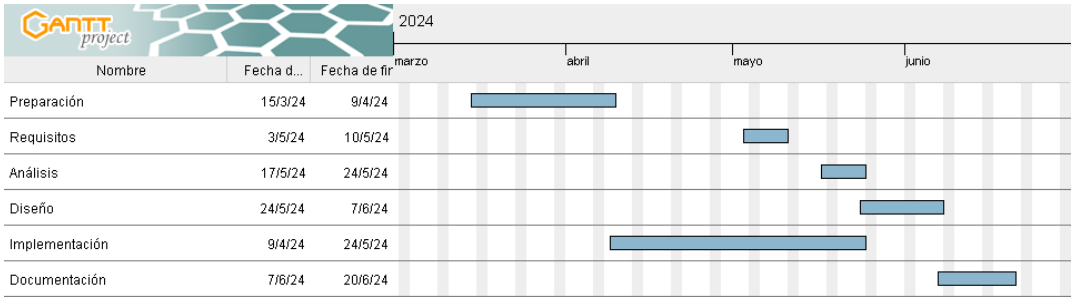


Figura 3.2: Diagrama de Gantt del seguimiento del proyecto

Capítulo 4

Requisitos

En este capítulo se presenta la especificación de requisitos software así como el modelo de casos de uso y la descripción detallada de estos.

Los requisitos consisten en objetivos y restricciones que se imponen al sistema para definir su comportamiento [Arlow and Neustadt, 2005]. Estos requisitos se pueden dividir en dos tipos principales según que parte del sistema definan, que son requisitos funcionales y no funcionales. Los primeros se centran en describir los objetivos principales del sistema, mientras que los no funcionales cubren el conjunto de restricciones y reglas que aportan calidad al software.

4.1. Descripción detallada del sistema

El sistema consistirá en una aplicación de escritorio. La aplicación permitirá la lectura de los datos de un fichero en formato CSV (comma Separated Values) para realizar el PCA de estos y representará una serie de gráficas correspondientes a dicho análisis.

La aplicación se podrá utilizar para ver y modificar los datos de un fichero, para posteriormente guardarlos si se desea en el mismo fichero o en otro distinto. Una vez cargado un fichero, sus datos se utilizarán para realizar el análisis de componentes principales, para ello se ofrecerán como métodos para seleccionar el número de componentes utilizar todas las variables, un número concreto de estas, o calcularlas mediante un porcentaje de varianza mínimo. Antes de dibujar las gráficas también se consultará al usuario si se quiere añadir una gráfica de validación cruzada.

Respecto a las gráficas, se mostrarán en un grupo de pestañas aparte, y represen-

tarán toda la información resultante del análisis. Mediante un diagrama de colores se informará acerca de la contribución de las variables a cada componente principal. Para la varianza habrá dos gráficas, un primer histograma con el porcentaje de varianza explicada por cada componente y un segúndo gráfico de líneas con el porcentaje de varianza explicada acumulada. Por último, del análisis se deberán mostrar una gráfica que compare los *loadings* de dos componentes principales, un diagrama de dispersión que compare los valores obtenidos de dos componentes, un *biplot* que combine los dos últimos gráficos, y una serie de gráficas de puntos que para cada variable compare los datos originales con la reconstrucción de los datos obtenidos mediante el análisis.

4.2. Requisitos funcionales

Los requisitos funcionales se tratan de una serie de necesidades y funcionalidades que debe cubrir el sistema [Arlow and Neustadt, 2005]. Definiendo los requisitos funcionales se determinarán los servicios que debe ofrecer el sistema.

En la Tabla 4.1 se muestra la lista de requisitos funcionales.

4.3. Requisitos no funcionales

Los requisitos no funcionales consisten en una serie de objetivos y restricciones que se imponen al sistema para mejorar su calidad y seguridad. Siguiendo el modelo FURPS+ [Grady and Caswell, 1987] entran en estos requisitos todos aquellos que influyan con la usabilidad, la fiabilidad, el rendimiento y el soporte, entre otros.

En la Tabla 4.2 se muestra la lista de requisitos no funcionales.

4.4. Requisitos de información

Los requisitos de información son aquellos que determinan la información que se va a almacenar en el sistema [Arlow and Neustadt, 2005]. Si bien estos requisitos pertenecen al conjunto de requisitos funcionales, se suelen separar de estos para facilitar la claridad y la creación de los modelos de dominio correspondientes.

Puesto que este sistema se trata de una aplicación de escritorio que únicamente genera el análisis y las gráficas de un conjunto de datos, no hay necesidad de almacenar ningún tipo de información acerca de usuarios, sesiones u otros datos necesarios

para el funcionamiento del sistema. Por lo tanto en este proyecto no se han fijado requisitos de información.

4.5. Modelo de casos de uso

El modelo de caso de uso consiste en una representación de las actividades del sistema y como los diferentes actores que participan en este sistema interactúan con él. Cada caso de uso define una de las posibles acciones que se pueden realizar durante el uso de la aplicación.

En la Figura 4.1 se muestra el diagrama de casos de uso del sistema.

4.6. Especificación de actores

Los actores de un sistema son aquellos que interactúan con él. En algunos tipos de sistemas, definir estos actores es un paso importante, ya que varios usuarios pueden tener diferentes niveles de permisos o necesitar realizar funciones distintas a otro tipo de usuarios.

En este proyecto, al tratarse de una aplicación de escritorio que no almacena ningún tipo de dato ni necesita diferenciar las funciones por roles, solo existe un tipo de actor. Por tanto el actor principal y único de este sistema es "Usuario".

El actor Usuario será quien inicie todos los casos de uso del sistema, representados en el diagrama de casos de uso de la Figura 4.1.

4.7. Especificación de casos de uso

En las especificaciones de los casos de uso se analizará en detalle los escenarios que se siguen para la realización de cada caso de uso.

Las tablas 4.3 a 4.9 muestran las especificaciones de los casos de uso del sistema.

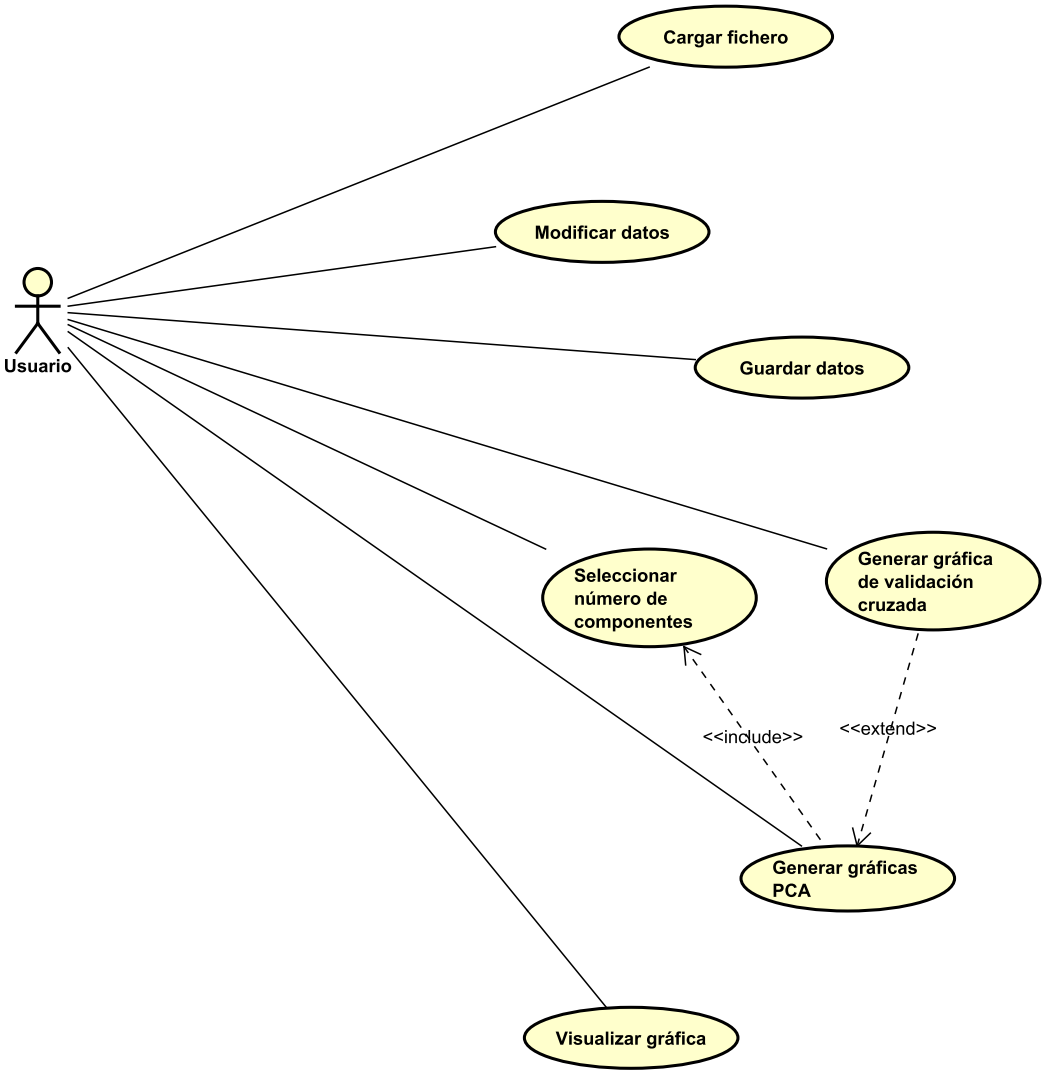


Figura 4.1: Diagrama de casos de uso.

Requisitos funcionales	
Id	Descripción
RF01	El sistema deberá permitir al usuario cargar y mostrar en una tabla los datos de un fichero cargado.
RF02	El sistema deberá permitir al usuario editar sobre una tabla los datos de un fichero.
RF03	El sistema deberá permitir al usuario guardar sobre un fichero los datos de la tabla de datos.
RF04	El sistema deberá permitir al usuario seleccionar una columna de la tabla para marcarla como columna de etiquetas y excluirla del análisis.
RF05	El sistema deberá permitir al usuario seleccionar el número de componentes principales empleado para el análisis.
RF06	El sistema deberá permitir al usuario representar en una gráfica el resultado de la validación cruzada.
RF07	El sistema deberá permitir mostrar al usuario la contribución de las variables a las componentes principales.
RF08	El sistema deberá permitir mostrar al usuario los porcentajes de varianza explicada de cada componente.
RF09	El sistema deberá permitir mostrar al usuario los porcentajes de varianza explicada acumulada de cada componente.
RF10	El sistema deberá permitir mostrar al usuario la posición de los *loadings* sobre dos componentes principales.
RF11	El sistema deberá permitir al usuario seleccionar las dos componentes principales sobre las que se mostrarán los *loadings*.
RF12	El sistema deberá permitir mostrar al usuario la dispersión de las variables sobre dos componentes principales.
RF13	El sistema deberá permitir al usuario seleccionar las dos componentes principales sobre las que se mostrarán la dispersión de las variables.
RF14	El sistema deberá permitir mostrar al usuario un *biplot* con la dispersión de las componentes principales y los *loadings*.
RF15	El sistema deberá permitir al usuario seleccionar las dos componentes principales sobre las que se representa el *biplot*.
RF16	El sistema deberá permitir mostrar al usuario la reconstrucción de cada variable a partir del análisis generado con el número de componentes principales seleccionado.

Tabla 4.1: Tabla de requisitos funcionales.

Requisitos no funcionales	
Id	Descripción
RNF01	El sistema deberá disponer de una interfaz de usuario fácil de usar.
RNF02	El sistema deberá poder utilizarse sin necesidad de formación previa en programación.
RNF03	El sistema deberá funcionar sin caídas ni errores.
RNF04	El sistema deberá tardar menos de un minuto en procesar o mostrar cualquier información solicitada por el usuario.
RNF05	El sistema deberá funcionar con el mínimo impacto sobre los recursos del dispositivo en el que se use.
RNF06	El sistema deberá ser desarrollado en Python.
RNF07	El sistema deberá adaptarse a cualquier tamaño de pantalla.
RNF08	El sistema deberá ser compatible con cualquier sistema operativo.
RNF09	El sistema deberá poseer una interfaz creada con Qt.

Tabla 4.2: Tabla de requisitos no funcionales.

Caso de uso "Cargar Fichero"	
Identificador:	CU01
Descripción:	El actor Usuario selecciona un fichero del cual cargar los datos.
Actores:	Usuario
Precondición:	El formato del fichero cargado deberá ser CSV.
Postcondición:	Los datos del fichero se mostrarán en la aplicación dentro de una tabla.
Secuencia normal	
1. El usuario selecciona la opción abrir fichero del menú. 2. El sistema muestra una ventana de selección de fichero del sistema operativo. 3. El usuario selecciona un fichero para cargar los datos. 4. El sistema valida el formato del fichero seleccionado. 5. El sistema muestra los datos en una tabla. 6. El caso de uso finaliza.	
Flujos alternativos	
3.a) El usuario cancela la selección de fichero y el caso de uso queda sin efecto. 4.a) El sistema comprueba que el formato del fichero no es correcto, informa al usuario y el caso de uso queda sin efecto.	

Tabla 4.3: Tabla de especificación del caso de uso "Cargar Fichero".

Caso de uso "Modificar Datos"	
Identificador:	CU02
Descripción:	El actor Usuario selecciona una casilla de la tabla de datos para modificar su valor
Actores:	Usuario
Precondición:	Debe haberse cargado un fichero de datos.
Postcondición:	La modificación permanecerá en la tabla hasta que se cargue otro fichero o se cierre la aplicación.
Secuencia normal	
1. El usuario selecciona una casilla en la tabla y modifica los datos. 2. El sistema actualiza los datos de la tabla. 3. El caso de uso finaliza.	

Tabla 4.4: Tabla de especificación del caso de uso "Modificar Datos".

Caso de uso "Guardar Datos"	
Identificador:	CU03
Descripción:	El actor Usuario selecciona guardar los datos en un fichero
Actores:	Usuario
Precondición:	Debe haberse cargado un fichero de datos.
Postcondición:	Los datos de la tabla quedarán almacenados en el fichero que seleccione el usuario.
Secuencia normal	
1. El usuario selecciona la opción guardar del menú. 2. El sistema comprueba la opción de guardado seleccionada. 3. El sistema muestra una ventana de guardado de fichero del sistema operativo. 4. El usuario guarda el archivo con el nombre introducido en la ubicación seleccionada. 5. El caso de uso finaliza.	
Flujos alternativos	
2.a) El sistema comprueba que se ha seleccionado la opción de guardar en el fichero actual, guarda los datos actuales en el fichero del que se habían cargado los datos y el caso de uso finaliza. 4.a) El usuario cancela el guardado del fichero y el caso de uso queda sin efecto.	

Tabla 4.5: Tabla de especificación del caso de uso "Guardar Fichero".

Caso de uso "Seleccionar Número De Componentes"	
Identificador:	CU04
Descripción:	El actor Usuario selecciona un método para determinar el número de componentes principales.
Actores:	Usuario
Precondición:	Debe haberse cargado un fichero de datos.
Postcondición:	El número de componentes principales queda fijado para el análisis de componentes principales.
Secuencia normal	
1. El usuario selecciona un método para establecer el número de componentes principales. 2. El sistema obtiene el número de componentes principales mediante el método elegido por el usuario. 2.1 El sistema comprueba que la opción marcada por el usuario es todas las componentes principales, establece el número de componentes como el número de variables. 2.2 El sistema comprueba que la opción marcada por el usuario es un número especifico de componentes principales, establece el número de componentes como el número introducido por el usuario. 2.3 El sistema comprueba que la opción marcada por el usuario es un porcentaje de varianza mínimo que se debe cumplir, el sistema calcula el número de componentes en función del porcentaje introducido por el usuario y el caso de uso finaliza. 3. El caso de uso finaliza.	
Flujos alternativos	
2.a) El sistema comprueba que el valor introducido por el usuario el paso 2.b) no es válido, informa al usuario y el caso de uso queda sin efecto. 2.b) El sistema comprueba que el valor introducido por el usuario el paso 2.c) no es válido, informa al usuario y el caso de uso queda sin efecto.	

Tabla 4.6: Tabla de especificación del caso de uso "Seleccionar Número De Componentes".

Caso de uso "Generar Gráfica de Validación Cruzada"	
Identificador:	CU05
Descripción:	El actor Usuario selecciona que se represente la gráfica de validación cruzada.
Actores:	Usuario
Precondición:	Debe haberse cargado un fichero de datos.
Postcondición:	La opción de generar o no la gráfica de validación cruzada quedará registrada.
Secuencia normal	
1. El usuario marca la casilla para generar la gráfica de validación cruzada. 2. El sistema guarda el estado de la casilla para generar validación cruzada. 3. El caso de uso finaliza.	

Tabla 4.7: Tabla de especificación del caso de uso "Generar Gráfica de Validación Cruzada".

Caso de uso "Generar Gráficas PCA"	
Identificador:	CU06
Descripción:	El actor Usuario selecciona la opción de generar las gráficas del análisis de componentes principales.
Actores:	Usuario
Precondición:	Debe haberse cargado un fichero de datos.
Postcondición:	Se han generado las gráficas correspondientes al análisis de componentes principales con el número de componentes fijado.
Secuencia normal	
1. El usuario selecciona una columna de etiquetas si lo considera conveniente. 2. El sistema marca la columna de etiquetas para excluirla del conjunto de gráficas que se van a generar. 3. El usuario selecciona la opción generar análisis. 4. El sistema muestra la ventana de selección del número de componentes principales. 5. «Punto de inclusion» Se realiza el caso de uso Seleccionar Número De Componentes especificado en la Tabla 4.6. 6. «Punto de extensión» Si el usuario escoge la opción de validación cruzada, se realiza el caso de uso Generar Gráfica De Validación Cruzada especificado en la Tabla 4.7. 7. El usuario selecciona la opción generar. 8. El sistema genera las gráficas del análisis de componentes principales. 9. El caso de uso finaliza.	
Flujos alternativos	
7.a) El usuario cierra la ventana de selección del número de componentes y el caso de uso queda sin efecto.	

Tabla 4.8: Tabla de especificación del caso de uso "Generar Gráficas PCA".

Caso de uso "Visualizar Gráfica"	
Identificador:	CU07
Descripción:	El actor Usuario selecciona la pestaña de la gráfica que desea visualizar.
Actores:	Usuario
Precondición:	Deben haberse generado las gráficas del análisis de componentes principales.
Postcondición:	El sistema muestra la gráfica seleccionada por el usuario.
Secuencia normal	
1. El usuario selecciona la pestaña de gráficas del análisis generado. 2. El sistema muestra las pestañas de las gráficas generadas para el análisis. 3. El usuario selecciona la gráfica que desea visualizar. 4. El sistema muestra la gráfica seleccionada por el usuario. 5. El usuario selecciona mediante los desplegables las dos componentes a comparar. 6. El sistema actualiza la gráfica en función de las dos componentes que se desean comparar. 7. El caso de uso finaliza.	
Flujos alternativos	
5.a) Si la gráfica no compara dos componentes principales, el caso de uso finaliza.	

Tabla 4.9: Tabla de especificación del caso de uso "Visualizar Gráfica".

Capítulo 5

Análisis

El objetivo de este capítulo será detallar el *workflow* de análisis del proyecto, proporcionando un "Modelo de Dominio" y un "Modelo de Análisis" del sistema. También se incluirán comentarios sobre casos de uso del análisis.

5.1. Modelo de Dominio

El "Modelo de Dominio" trata de proporcionar una representación de los conceptos que definen el problema, y cuyo principal objetivo es identificar las clases y los atributos que lo componen [Arlow and Neustadt, 2005]. También se encarga de modelar las relaciones entre estas clases.

La clase DataSet contiene los datos cargados del fichero, y está a su vez compuesta por otras dos clases, Fila y Columna. Cada una de estas clases contiene respectivamente una fila y una columna del DataSet.

La clase ModeloPCA consiste en el PCA realizado con el número de componentes seleccionado. Este a su vez estará compuesto por los *loadings* y los *scores*.

Por último, las clases de tipo Gráfica serán las encargadas de contener las representaciones del análisis. Debido a que cada gráfica se encarga de preparar los datos de diferente manera al resto, se ha elegido crear una clase para cada una de estas. Además, dado que algunas de las gráficas comparan dos componentes entre si, ha sido necesario crear la clase CustomPlot para poder almacenar como atributos las dos componentes que se van a comparar.

En la Figura 5.1 se muestra el "Modelo de Dominio".

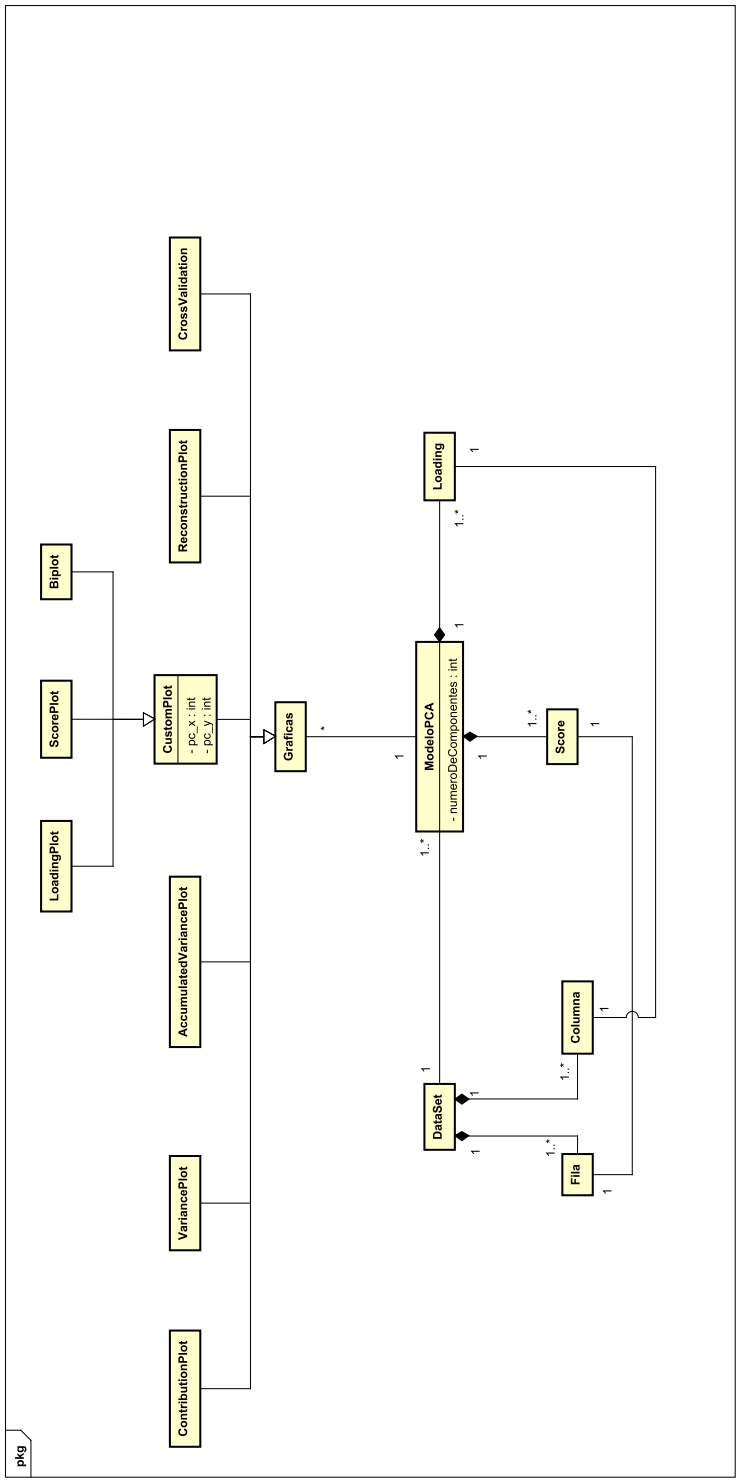


Figura 5.1: Modelo de Dominio.

5.2. Modelo de Análisis

El “Modelo de Análisis” se ocupará de definir con más detalle las clases representadas en el diagrama “Modelo de Dominio”, añadiéndose a este las operaciones u otras clases que sean necesarias. El ‘Modelo de Análisis” está formado por las clases de análisis dotadas de operaciones y la realización de los casos de uso [Arlow and Neustadt, 2005].

En la Figura 5.2 se muestra el ”Modelo de Análisis”.

Los casos de uso en análisis permiten visualizar el seguimiento de como se desarrollan los distintos casos de uso del sistema por medio de las interacciones entre clases, los actores y el propio sistema. Esta visualización se realiza mediante diagramas de secuencia, los cuales permiten mostrar a lo largo del tiempo las operaciones que se han producido para llevar a cabo el caso de uso.

Dado que la realización de varios de los casos de uso de este proyecto no conllevan demasiadas interacciones, se ha decidido mostrar únicamente uno de los diagramas de secuencia que define en más detalle el uso principal de la aplicación. Solo se ha dotado de operaciones a las clases que se usarán en los casos de uso realizados en este capitulo.

En la Figura 5.3 se muestra la realización del caso de uso “Generar Gráficas PCA” especificado en la Tabla 4.8.

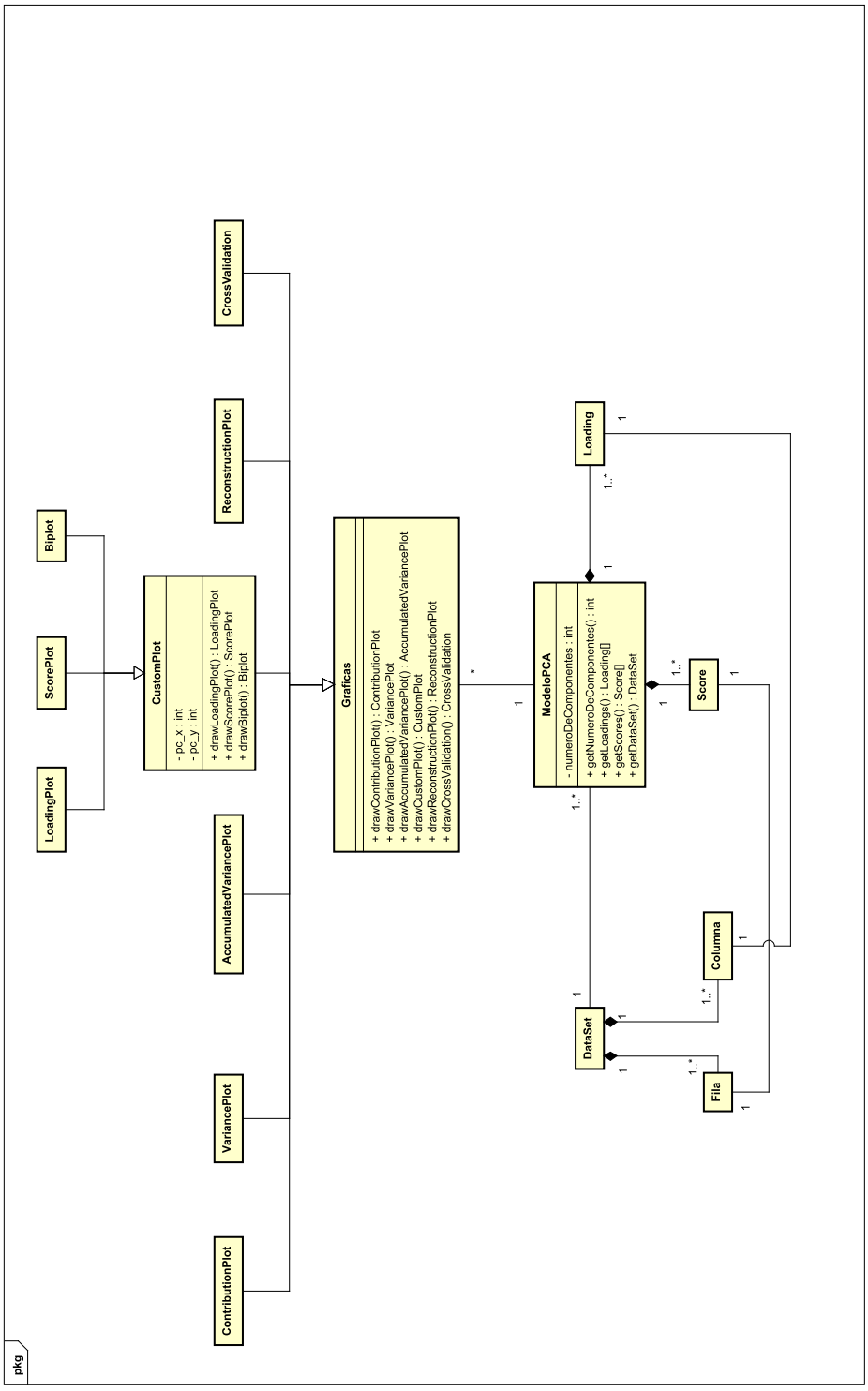


Figura 5.2: Modelo de Análisis.

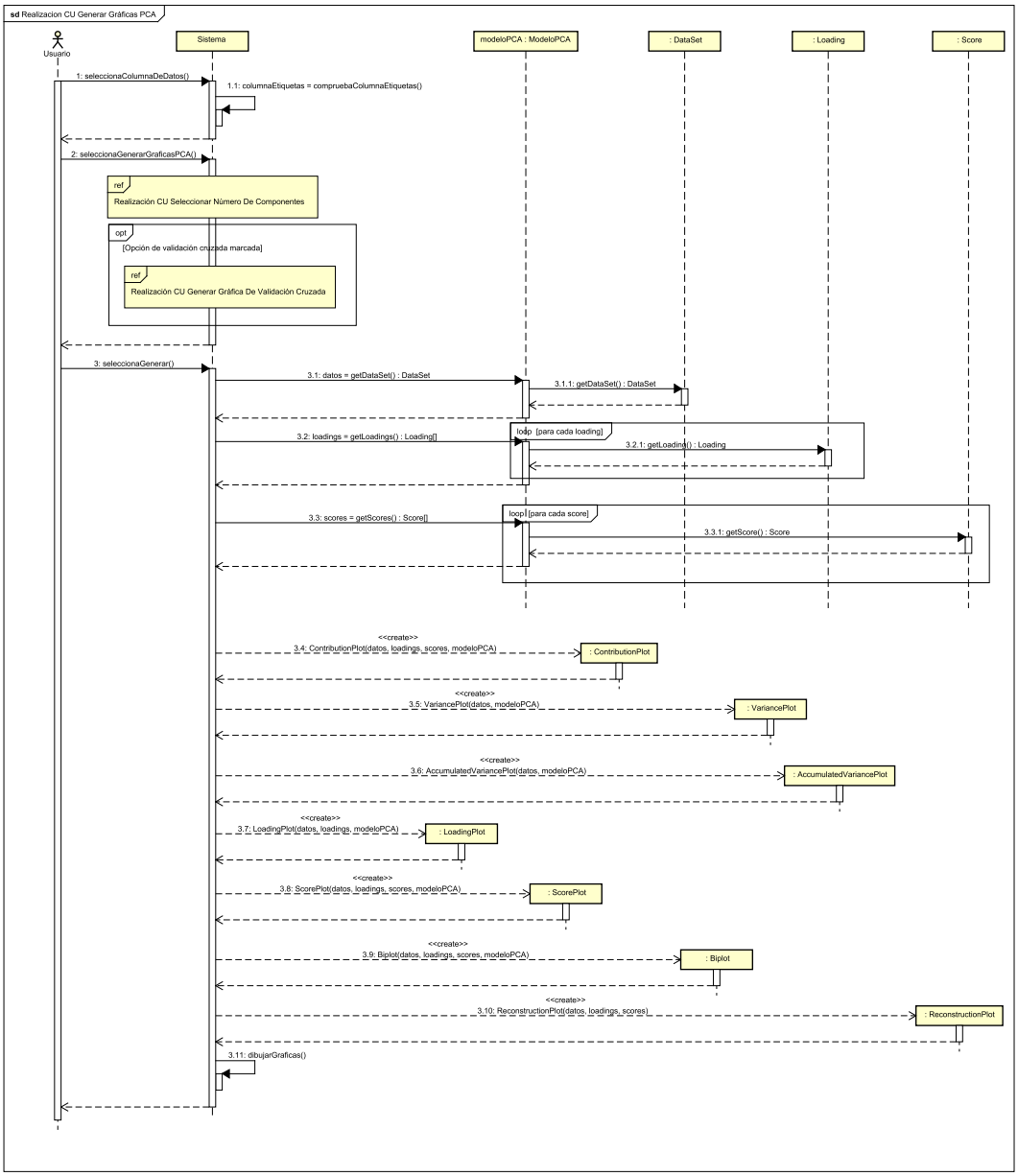


Figura 5.3: Realización del caso de uso “Generar Gráficas PCA”.

Capítulo 6

Diseño

En este capítulo se describirá el *workflow* de diseño del proyecto, que incluirá la "Arquitectura Física" y la "Arquitectura Lógica" del sistema, incluyendo la definición de las capas de la aplicación, la realización de casos de uso en diseño y, por último, el diseño de la interfaz de usuario.

6.1. Arquitectura física

La arquitectura física del sistema está compuesta por los distintos componentes y dispositivos hardware, sobre los cuales se despliegan los elementos software desarrollados. Su representación se realizará típicamente mediante diagramas de despliegue de UML. Dado que en este proyecto se ha desarrollado una aplicación de escritorio, el diagrama resultante es bastante sencillo.

En la Figura 6.1 se muestra la arquitectura física del sistema, en la que se han incluido las principales dependencias de la aplicación.

6.2. Arquitectura lógica

La arquitectura lógica del sistema puede seguir algún patrón arquitectónico que permite dividir la estructura del sistema en elementos más sencillos. El patrón capas o *layered pattern* se trata de una de las posibles maneras en las que se puede realizar el diseño de un sistema [Buschmann et al., 2001]. En este patrón como su nombre indica, se separan los componentes del sistema en capas, de tal forma que cada uno

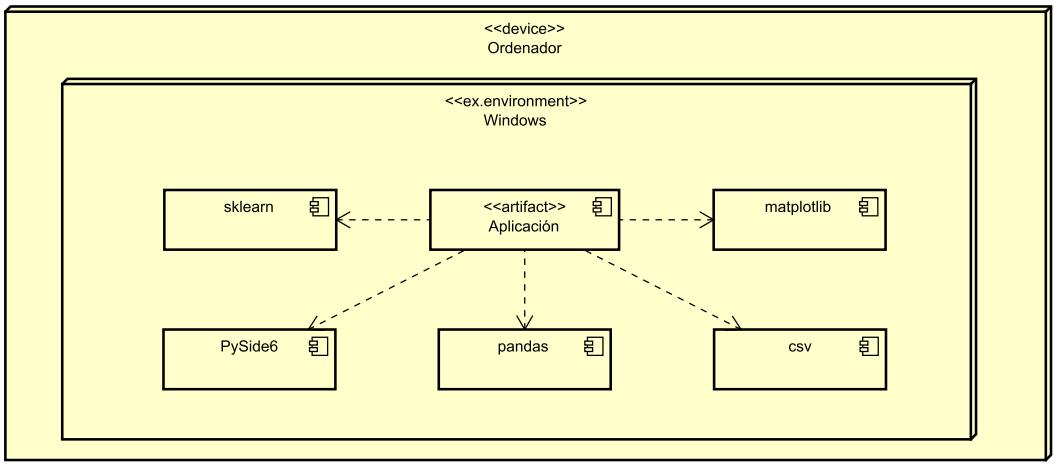


Figura 6.1: Arquitectura física del sistema.

de los componentes de cada capa solo se comunique con los demás componentes de su capa o de las capas inferiores.

En la Figura 6.2 se muestra la arquitectura lógica del sistema.

6.2.1. Capa de Presentación

La capa de presentación contendrá los elementos necesarios para la interfaz de la aplicación, siendo estos los archivos de interfaz de usuario .ui con sus respectivos controladores. Como se verá en el diagrama de clases, en esta capa existirán dos tipos principales de clases.

Por un lado las clases MainWindow, PcSelectionView y CustomPlotView se tratan de las vistas creadas en archivos .ui que contienen todos los detalles de los elementos y su ubicación en la interfaz de usuario. Por tanto, ya que serían demasiados, no se mostrarán los atributos que poseen estas clases.

Por otro lado, están las clases Ui_MainWindow, Ui_PcSelectionView y Ui_CustomPlotView las cuales son los controladores de las clases anteriores. Estas son creadas automáticamente durante la compilación por Qt, y contienen los métodos utilizados para dibujar las vistas en la aplicación.

En la Figura 6.3 se muestra el diagrama de clases de la capa de presentación.

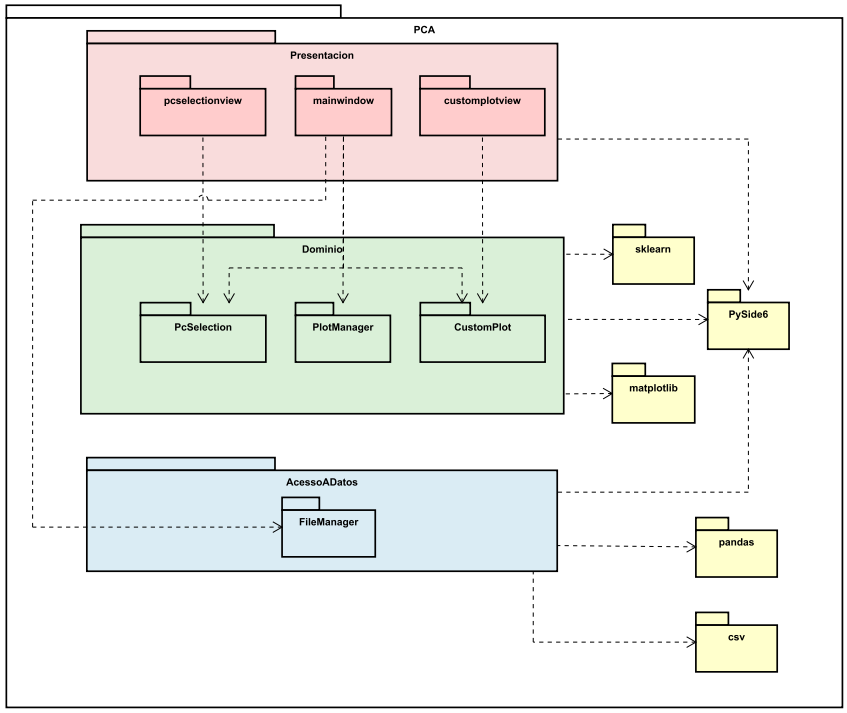


Figura 6.2: Arquitectura lógica del sistema.

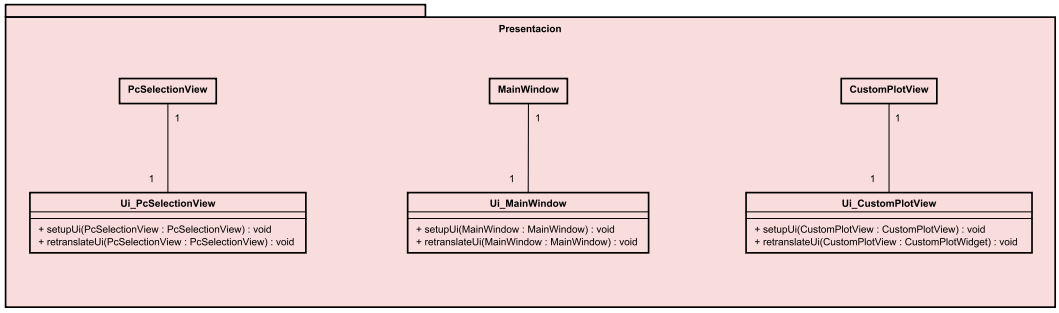


Figura 6.3: Diagrama de clases de la capa de presentación.

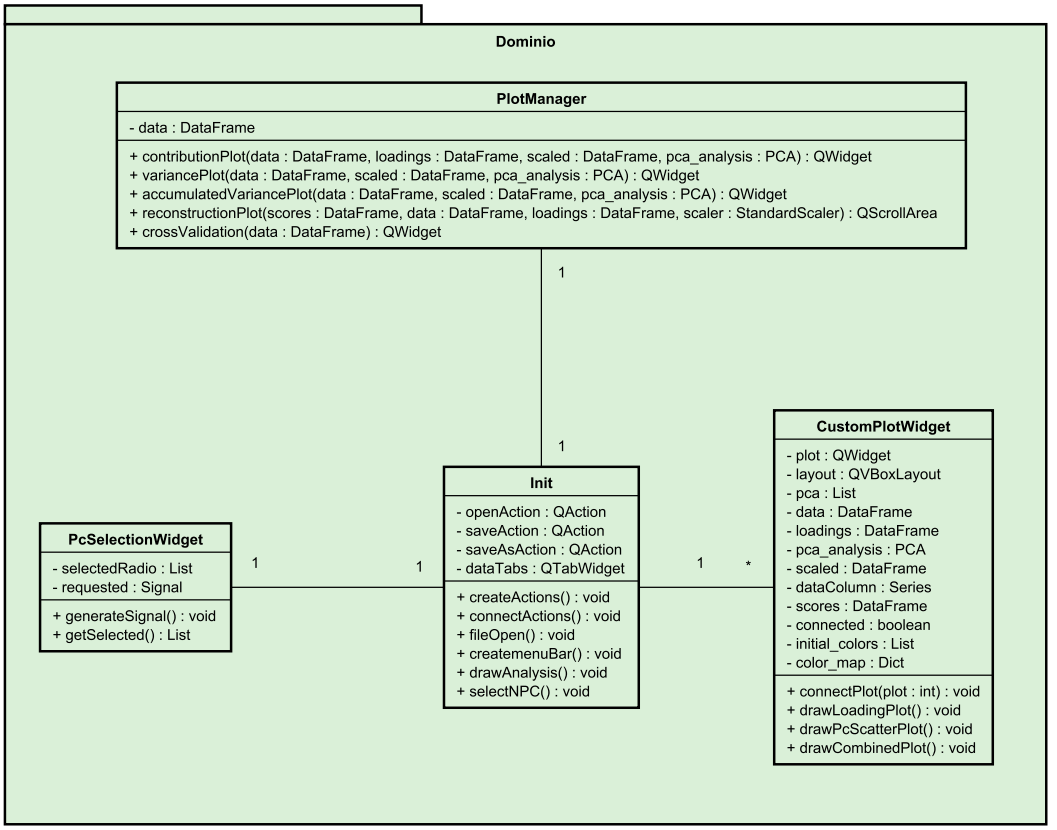


Figura 6.4: Diagrama de clases de la capa de dominio.

6.2.2. Capa de Dominio

En la capa de Dominio se encuentran las clases que se encargaran de realizar la lógica de la aplicación. Comúnmente, esta capa contiene las clases modelo del patrón MVC y los controladores de los casos de uso. Debido a que el modelo en esta aplicación se gestiona por medio de librerías, las únicas clases que se representarán en el diagrama de esta capa serán los controladores que se encargan de la lógica del programa.

En la Figura 6.4 se muestra el diagrama de clases de la capa de dominio.

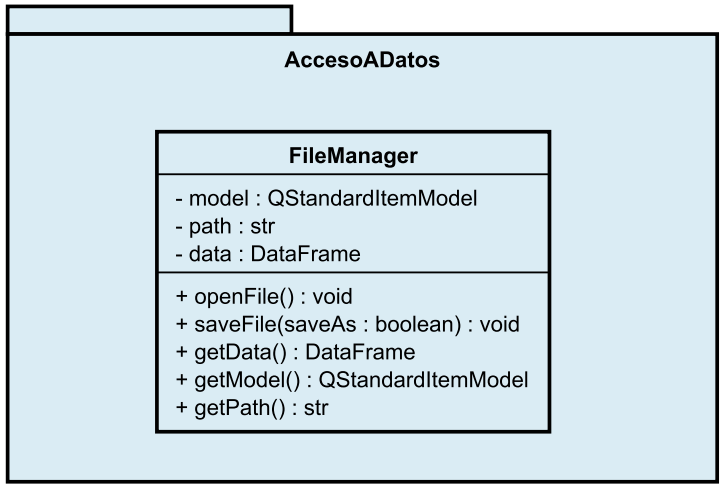


Figura 6.5: Diagrama de clases de la Capa de Acceso A Datos.

6.2.3. Capa de Acceso a Datos

La capa de acceso a datos como su nombre sugiere será la encargada de cargar los datos de los ficheros que seleccione el usuario, así como de guardar las modificaciones que se realicen durante la ejecución de la aplicación cuando el usuario lo indique. Debido a que las operaciones necesarias para realizar esta lectura y guardado son pocas y sencillas, el acceso a datos se realiza por medio de una única clase llamada FileManager.

En la Figura 6.5 se muestra el diagrama de clases de la capa de acceo a datos

6.3. Realización Casos de Uso de Diseño

Por brevedad, se mostrará únicamente uno de los casos de uso del sistema, distinto al que se mostró en la Figura 5.3.

En la Figura 6.6 se muestra la realización del caso de uso “Cargar Fichero”.

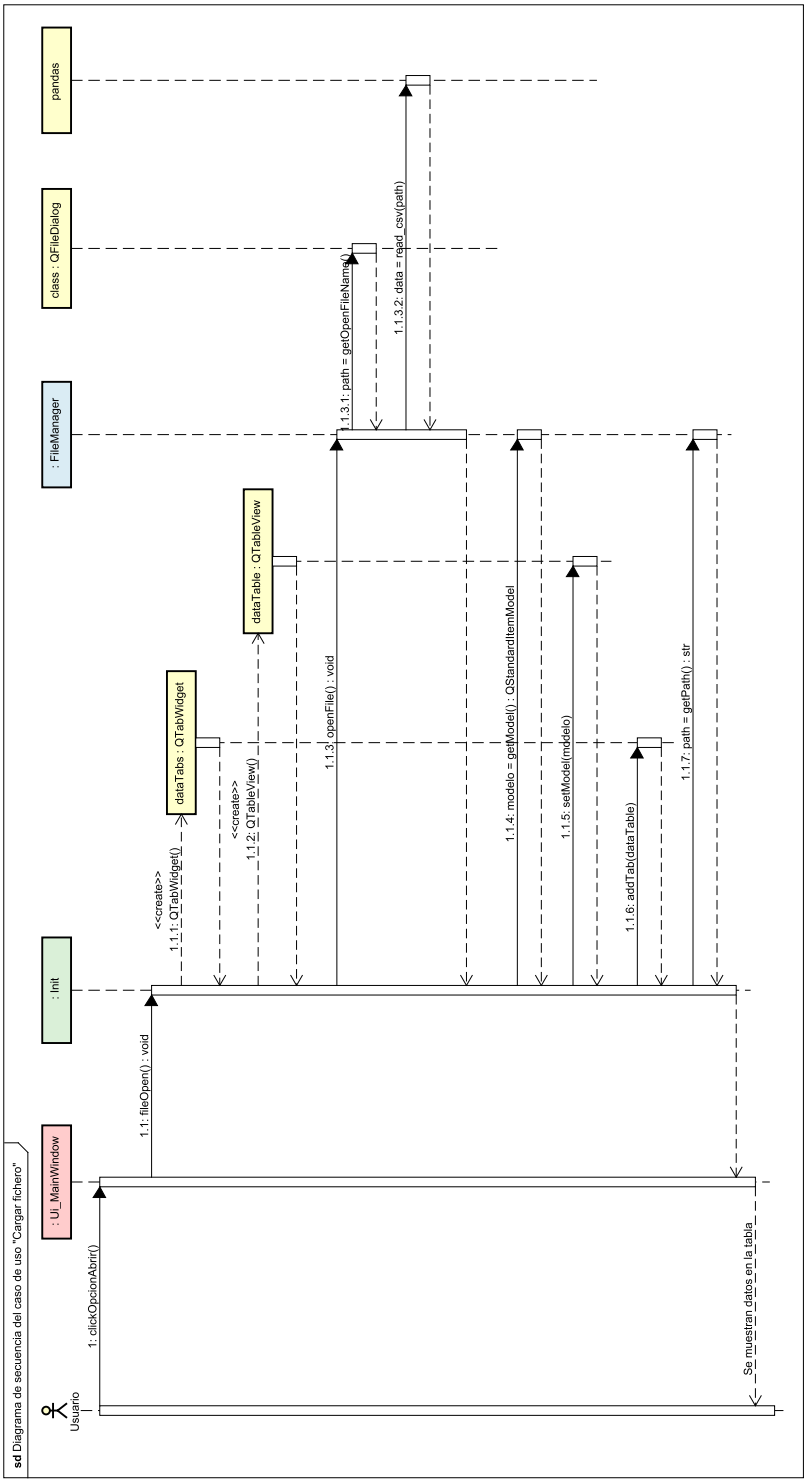


Figura 6.6: Realizacion CU Cargar Fichero en Diseño.

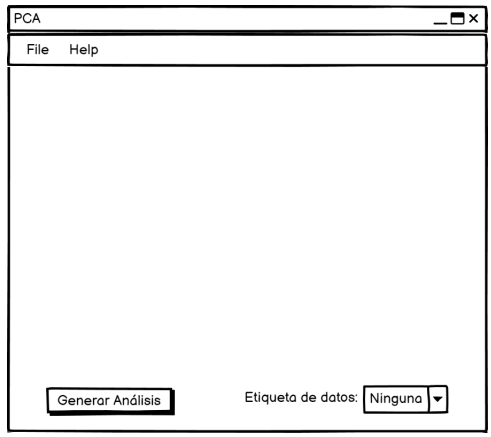


Figura 6.7: Boceto de interfaz de usuario al iniciar la aplicación.

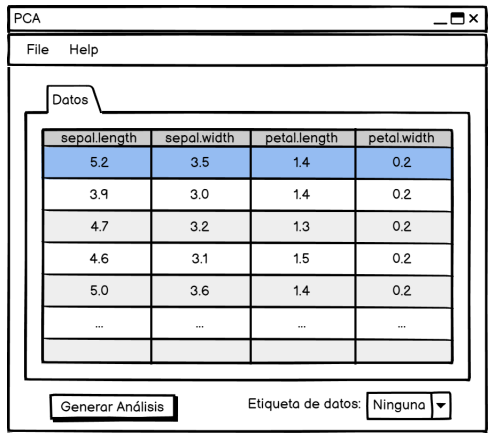


Figura 6.8: Boceto de interfaz de usuario al cargar los datos.

6.4. Diseño Interfaz Gráfica

Antes de comenzar con la implementación de la aplicación, se realizó una breve tarea en la que se realizaron los bocetos de la interfaz de usuario que deberá tener la aplicación. Los bocetos de esta interfaz se realizaron con la herramienta Balsamiq [Balsamiq Studios, LLC, 2022] y sirven como orientación a la hora de desarrollar la interfaz de usuario con Qt.

En las Figuras 6.7 a 6.12 se muestran los bocetos de la interfaz de usuario realizados.

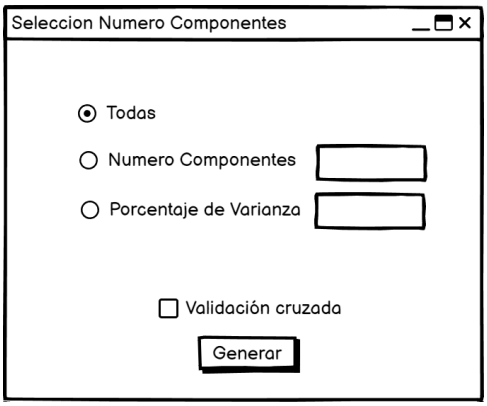


Figura 6.9: Boceto de interfaz de usuario para seleccionar el número de componentes.

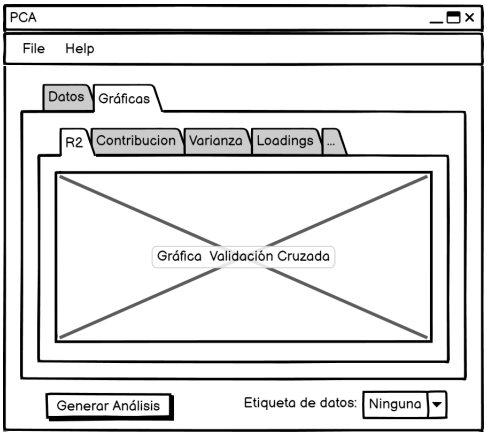


Figura 6.10: Boceto de interfaz de usuario para mostrar las gráficas.

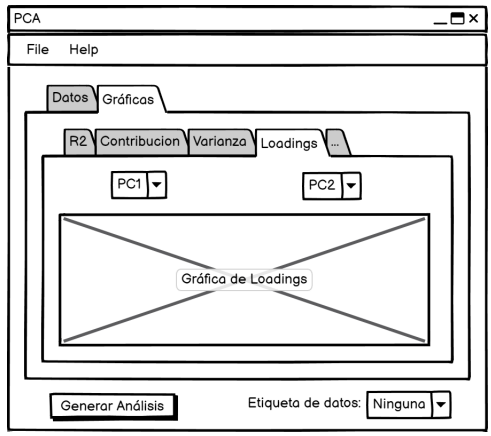


Figura 6.11: Boceto de interfaz de usuario para mostrar las gráficas que comparan componentes principales.

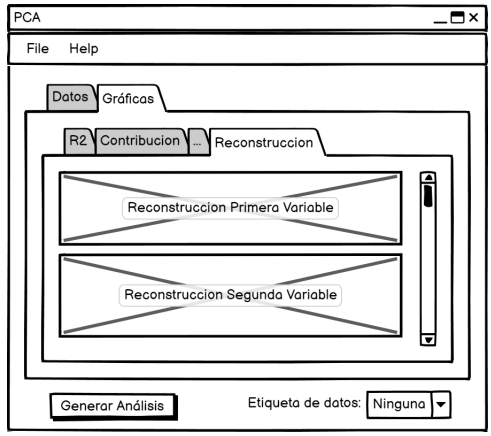


Figura 6.12: Boceto de interfaz de usuario para mostrar las gráficas de reconstrucción de las variables.

Capítulo 7

Implementación

En este capítulo se presenta el *workflow* de implementación final del proyecto. Se verá, primeramente, cuáles han sido las tecnologías utilizadas en la elaboración del mismo. Se comentará también cómo se ha organizado el código de la aplicación y se proporciona una breve explicación de cómo funciona el patrón Modelo-Vista-Controlador (MVC) en Qt. El capítulo incluye también una descripción de cómo se debe usar la aplicación y las distintas salidas que ofrece al usuario. Esta descripción está ilustrada con dos conjuntos de datos reales, Iris y Decathlon, adjuntados a la aplicación como conjuntos de datos de ejemplo. Se aprovecha esta descripción para comentar brevemente cómo debe el usuario interpretar las salidas gráficas y numéricas proporcionadas. Se podrá especial énfasis en la propuesta gráfica de validación cruzada, como procedimiento que ayuda al establecimiento, por parte del usuario, del número de componentes principales a retener.

7.1. Tecnologías utilizadas

En este proyecto se han utilizado distintas tecnologías para la realización de la aplicación, las cuales se describirán a continuación.

7.1.1. Python

Python [Python Software Foundation, 2024] es el lenguaje de programación que se ha utilizado para este proyecto. Se trata de un interprete gratuito que ha aumentado su popularidad en los últimos años debido a su facilidad tanto de uso como de aprendizaje. También resulta muy cómodo para utilizar librerías, ya que incluye

una gran variedad de ellas bastante útiles dentro del interprete, como por ejemplo *pandas*, *csv*, *os* o *sys*, las cuales han sido utilizadas todas en este proyecto.

Para la obtención del PCA se han añadido las librerías *numpy*, *matplotlib* and *scikit-learn*.

7.1.2. Qt

Qt [Qt Company, 2024] es un *framework* de desarrollo escrito en C++ para el desarrollo de aplicaciones de escritorio. A pesar de estar escrito en C++, también ofrece versiones o *bindings* con los que desarrollar aplicaciones en otros lenguajes, siendo una de estas versiones Python, con la cual se ha realizado el proyecto.

Qt tambien ofrece un entorno de desarrollo propio llamado QtCreator, el cual también se ha usado para desarrollar la aplicación y que incorpora QtDesigner, el cual es la herramienta con la que se añade la interfaz de usuario.

7.1.3. Gitlab

Gitlab [GitLab Inc., 2011] es un servicio web utilizado principalmente para el control de versiones el cual está basado en Git. Se ha utilizado para poder almacenar en un repositorio remoto el código de la aplicación.

7.1.4. Astah

Astah [Change Vision, Inc, 2020] es un software que ofrece una gran variedad de herramientas de modelado y que permite realizar los diagramas necesarios para las fases de análisis y diseño del proyecto. En concreto se ha utilizado Astah Proffesional debido a que es la versión mas completa y es la licencia que ofrece la Universidad de Valladolid.

7.1.5. Balsamiq

Balsamiq [Balsamiq Studios, LLC, 2022], ya sea en su versión de escritorio *Wireframes* o su versión web *Cloud*, es una herramienta que permite realizar de manera sencilla los bocetos de interfaz de usuario de cualquier tipo de aplicación, permitiendo seleccionar y ordenar elementos previamente dibujados con una estética hecha a mano hasta formar la interfaz deseada.

7.1.6. Overleaf

Overleaf [Hammersley and Lees-Miller, 2012] es un editor online LaTeX que simplifica la escritura de documentos de manera colaborativa. En el editor de overleaf también se encuentra un compilador que permite actualizar el documento de manera sencilla, facilitando la visualización de los cambios realizados.

7.1.7. Inkscape

Inkscape [Equipo Inkscape, 2023] es una herramienta que permite crear y modificar gráficos vectoriales. Se ha utilizado para convertir las imágenes en gráficos vectoriales, permitiendo así poder ampliarlas sin perder la resolución.

7.1.8. GanttProject

GanttProject [Thomas and Barashev, 2003] es una herramienta que permite realizar diagramas de Gantt, los cuales se han utilizado para realizar la planificación inicial y el seguimiento en el capítulo de planificación.

7.2. Organización del código

El código estará organizado de tal manera que las carpetas se correspondan con las capas que se definieron en el patrón arquitectónico realizado durante la fase de diseño. Hay una carpeta "Presentación" la cual contiene todos los ficheros utilizados para mostrar la interfaz de usuario, una carpeta "Dominio" que contendrá los controladores y las clases que se encargarán de la funcionalidad de la aplicación, y una carpeta "AccesoADatos" en la que se encuentra la clase encargada de gestionar la apertura y guardado de los datos en un fichero.

Para crear la vista de las carpetas del proyecto se ha utilizado el comando de *Windows tree*, el cual sirve para mostrar la estructura de directorios de la ruta en la que se ejecuta. Para esta visualización solo se han incluido las carpetas y los ficheros que contienen las clases que se han implementado, excluyendo otras como el entorno de Qt o las carpetas utilizadas para el repositorio Git.

En la Figura 7.1 se muestra la organización del código.

```
C:.
    Init.py
    PCA.pyproject

---AccesoADatos
        FileManager.py

---Dominio
        CustomPlot.py
        PcSelection.py
        PlotManager.py

---Presentacion
        customplotview.ui
        mainwindow.ui
        pcselectionview.ui
        ui_customplotview.py
        ui_mainwindow.py
        ui_pcselectionview.py
```

Figura 7.1: Organización del código.

7.3. El patrón Modelo-Vista-Controlador en Qt

Para la implementación de la interfaz gráfica, se ha utilizado la librería de desarrollo de interfaces graficas Qt [Qt Company, 2024]. Esta, ofrece una serie de clases por medio de las cuales utilizar el patrón MVC. En la Figura 7.2 se muestra la representación del MVC que utiliza Qt.

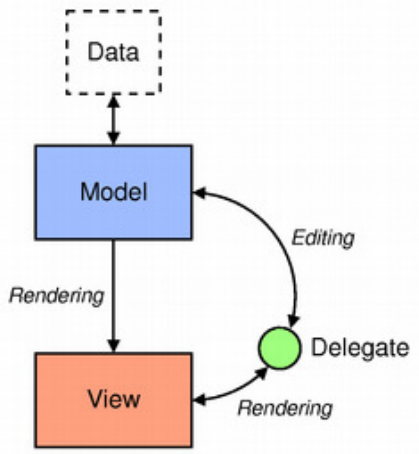


Figura 7.2: Representación del patrón MVC de Qt (https://doc.qt.io/qt-6/model-view-programming.html).

En el modelo, todas las clases se basan en QAbstractItemModel, la cual se encarga de almacenar y gestionar los datos. De los diferentes modelos ya preparados que ofrece Qt y que se basan en esta clase, para este proyecto se utilizó QStandardItemModel por ser el que permite trabajar con conjuntos de datos más complejos y personalizados, que en este caso serán los datos de los ficheros .CSV (Comma Separated Values).

En la vista, Qt ofrece clases ya implementadas y que permiten mostrar los datos como se desee. Una de las clases que se ha utilizado para este proyecto es QTableView que se encargará de representar los datos en una tabla.

Por último, también se ofrecen clases que realizan la función de controlador y que se basan en la clase QAbstractItemDelegate. Sin embargo, en este proyecto no se han utilizado, ya que se ha decidido implementar las interacciones entre el modelo y la vista de manera independiente.

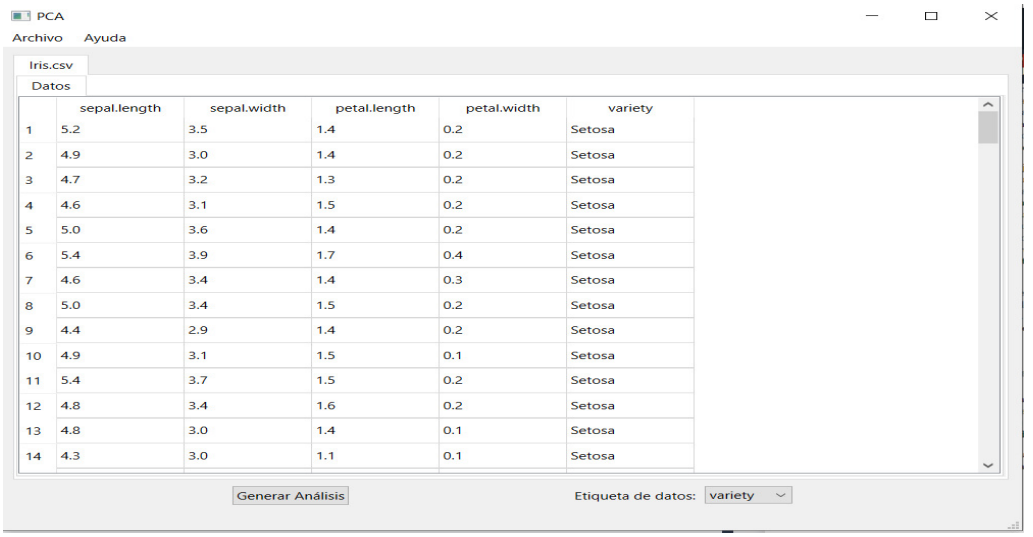
7.4. Implementación final y uso de la aplicación

7.4.1. Manejo de los datos

Siguiendo la notación introducida en el Capítulo 2, tras cargar los datos desde un fichero CSV, se genera una matriz de datos $\mathbf{X}$ donde cada fila i de la matriz corresponderá a un individuo o caso ($i = 1, ..., I$) y cada columna j será una de las variables medidas sobre los individuos ($j = 1, ..., J$). De esta forma, $\mathbf{X}$ será una matriz $I \times J$ y se denota por x_{ij} al elemento en la posición (i, j) de dicha matriz, como el valor que toma el individuo i en la variable j. El programa permite la visualización de la tabla de datos, la modificación de datos desde la misma aplicación y guardar los cambios realizados en el conjunto de datos en el mismo fichero u otro distinto. El conjunto de datos debe estar completo y no incluir valores perdidos o *missing* puesto que no se incorpora ningún tratamiento a esta situación.

Como y se comentó en la Sección 2, en PCA es común trabajar con las variables centradas y estandarizadas, de tal forma que las medias por columnas sean 0 y las desviaciones típicas por columnas sean 1. Esta estandarización es realizada internamente por la aplicación reemplazando los datos originales x_{ij} por su versión centrada y escalada $(x_{ij} - \overline{x}_j)/S_j$, donde $\overline{x}_j$ es la media de la variable j, que fue introducida en (2.1), y S_j la desviación típica de la variable j, introducida en (2.2). Esta transformación de los datos pone las J variables en una misma escala y, de esta forma, PCA no será dependiente de la escala concreta en la que fueron medidas las variables. Esta estandarización es realizada por *Scikit-Learn* mediante `StandardScaler(with_std=True,with_mean=True)`.

La Figura 7.3 muestra la pantalla de entrada de datos.



	sepal.length	sepal.width	petal.length	petal.width	variety
1	5.2	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
3	4.7	3.2	1.3	0.2	Setosa
4	4.6	3.1	1.5	0.2	Setosa
5	5.0	3.6	1.4	0.2	Setosa
6	5.4	3.9	1.7	0.4	Setosa
7	4.6	3.4	1.4	0.3	Setosa
8	5.0	3.4	1.5	0.2	Setosa
9	4.4	2.9	1.4	0.2	Setosa
10	4.9	3.1	1.5	0.1	Setosa
11	5.4	3.7	1.5	0.2	Setosa
12	4.8	3.4	1.6	0.2	Setosa
13	4.8	3.0	1.4	0.1	Setosa
14	4.3	3.0	1.1	0.1	Setosa

Figura 7.3: Pantalla de datos.

Otra característica de la aplicación desarrollada es que permite introducir una variable categórica de "etiquetas" y que será usada en los distintos gráficos identificando con colores distintos los distintos grupos de individuos creados por dicha variable categórica. Esta variable puede ser elegida, cuando esté disponible, seleccionando dicha variable en el desplegable de "Etiqueta de datos" previamente a "Generar el análisis".

7.4.2. Conjuntos de datos de ejemplo

La aplicación incluye a modo de ejemplo y para poder hacer pruebas de su aplicabilidad dos conjuntos de datos: Iris y Decathlon. En este capítulo se usarán dichos conjuntos de datos para ilustrar el uso de la aplicación y las distintas salidas gráficas que genera.

Uno de los conjuntos de datos incluidos es el conjunto de datos Iris. Este es uno de los conjuntos de datos más famosos para analizar datos multivariantes y fue introducido por Fisher Fisher [1936]. El conjunto de datos Iris contiene medidas para $I = 150$ flores de tipo "iris" divididas en tres especies: "setosa", "versicolor" y "virginica". Para cada flor, se han medido $J = 4$ variables: longitud del sépalo (sepal.length), ancho del sépalo (sepal.width), longitud del pétalo (petal.length) y ancho del pétalo (petal.width), todas ellas medidas en centímetros.

Otro conjunto de datos que será también usado es Decathlon (Lê et al. [2008]), que

contiene medidas de $I = 41$ atletas en las 10 pruebas que conforman la competición de atletismo de "decathlon" (tiempo en segundos en recorrer 100 metros (100m); salto de longitud en metros (long_jump); lanzamiento de peso (Shot_put) en metros; salto de altura en metros (High_jump); ...) junto a una variable de puntuación (Points) con la puntuación otorgadas al atleta con esos resultados en las 10 pruebas realizadas. Se tiene así un conjunto en dimensión $J = 10 + 1 = 11$.

7.4.3. Selección del número de componentes

Una vez que los datos están estandarizados y la etiqueta seleccionada (si de decide usar esa variable de etiqueta) se debe decidir cuántas componentes principales extraer y ese número seleccionado de componentes se denotará en lo que sigue por A. La elección del número de componentes A es sin duda uno de los problemas más complejos en PCA y no está para nada resuelto. Su determinación es bastante dependiente del objetivo final de aplicación del PCA, como ya se comentó en la Sección 2

El programa permite al usuario seleccionar el número de componentes al presionar el botón "Generar análisis". Por defecto, determina tantas componentes como variables se tengan en el conjunto de datos al marcar "Todas". Es decir, si tenemos J variables, el programa selecciona $A = J$ (lo que es factible si no exista una relación lineal exacta entre las columnas de la matriz $\mathbf{X}$).

La Figura 7.4 muestra una captura de pantalla con las posibilidades existentes para seleccionar el número de componentes.

Para la realización del PCA se ha usado `sklearn.decomposition` con `pca=PCA()`, y de tal forma que el resultado de aplicar PCA se almacena en un objeto en Python, que denominaremos `pca`. Así, `pca.components_` proporciona los A autovectores de la matriz $\mathbf{X}^t\mathbf{X}$ (ver Capítulo 2) y que (introducidos por columnas) constituirán la matriz de *loadings*

$$\mathbf{P}_{1:J} = (\mathbf{p}_1|\mathbf{p}_2|...|\mathbf{p}_J).$$

Estos autovectores están asociados a autovalores $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_J$ de la matriz $\mathbf{X}^t\mathbf{X}$ y todos esos autovectores con norma igual a 1.

El usuario también puede elegir un número de específico de componentes A (quizás bastante) menor que J. Este número de componentes A es elegido por el usuario al presionar el botón "Generar análisis" y marcar "Número de componentes". El usuario estaría así seleccionando el parámetro `n_components` en `PCA()`.

En la selección del número de componentes es habitual quedarse con los componentes asociados a autovalores mayores que 1, siguiendo la recomendación en Kaiser

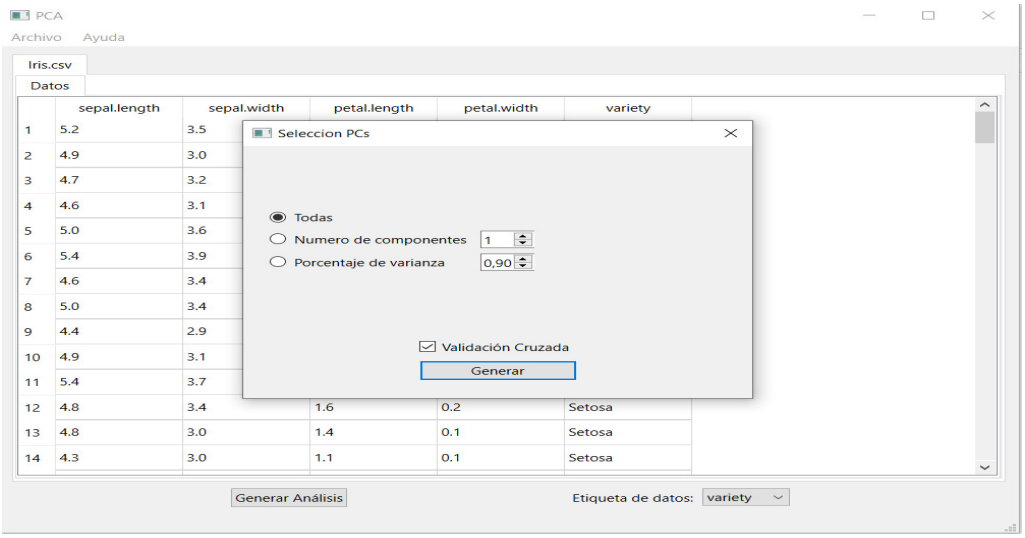


Figura 7.4: Selección del número de componentes.

[1960], por lo que esta forma de proceder suele conocerse como "regla Kaiser". No obstante, esta selección de A es bastante discutible y una práctica quizás más recomendable es elegir el menor A para el que se consigue una proporción de varianza explicada (ver Capítulo 2) prefijada. Se puede ver que la variabilidad explicada si se eligen A componentes coincide con

$$\lambda_1 + \lambda_2 + ... + \lambda_A.$$

De hecho, dado que la variabilidad total coincide con $\lambda_1+\lambda_2+...+\lambda_J$, se puede elegir A de tal forma que se logre alcanzar un determinado porcentaje de la variabilidad total explicada. Al presionar el botón "Generar análisis", el usuario puede elegir un "Porcentaje de varianza" en "tanto por ciento" a explicar. Es habitual seleccionar un 80 % o 90 %, aunque algunas aplicaciones del PCA puedan requerir porcentajes mayores. El programa considera por defecto un porcentaje del 90 % pero el usuario puede también elegir cualquier otro porcentaje q % de variabilidad explicada. En ese caso, el programa selecciona como número de componentes al menor A tal que

$$\frac{\lambda_1 + ... + \lambda_A}{\lambda_1 + ... + \lambda_J} \geq q \cdot \frac{1}{100}.$$

Esta selección se puede realizar proporcionando en `n_components` de *Scikit-Learn* un valor $q/100$ en el intervalo $(0, 1)$ en lugar de un valor $A \in \mathbb{N}$.

Como fue comentado en el Capítulo 2, una crítica bastante común al PCA es su subjetividad en la elección del número de componentes A o del porcentaje q. En

PCA no existen variables respuesta a predecir y esto hace que no sea inmediato el poder aplicar técnicas de validación cruzada para elegir A.

Reconociendo esta dificultad en usar la validación cruzada para elegir A, con el fin de guiar al usuario en la elección de A, el programa ha incluido un procedimiento de visualización de las variabilidades explicadas que sí tiene en cuenta la imprecisión inherente a la estimación de estas variabilidades y desea prevenir sobre-optimismo por sobreajuste. Este procedimiento será detallado en la Sección 7.4.5. La técnica de validación cruzada proporcionada allí es computacionalmente costosa y solo se ejecutará si el usuario marca la opción "Validación cruzada" en "Generar análisis".

7.4.4. Gráficos proporcionados

Una vez elegido el número de componentes principales A y generado el análisis aparece una pestaña "Gráficas A componentes". Esta pestaña se despliega a su vez en otras sub-ventanas de gráficas que se describen a continuación:

1. **R2 K-fold:** Este gráfico solo es disponible si se marca la opción "Validación cruzada" en "Generar análisis". El procedimiento de validación cruzada implementado será posteriormente descrito.

2. **Gráfico de contribución de las variables:** Este gráfico muestra "gráfico de de calor" o *heatmap* con los valores de las columnas de la matriz de loadings.

   $$\mathbf{P}_{1:A} = (\mathbf{p}_1|\mathbf{p}_2|...|\mathbf{p}_A).$$

   El gráfico muestra los pesos que se han usado para obtener cada una de las A componentes principales. Los coeficientes están forzosamente en el intervalo $(-1, 1)$ ya que, por definición, tienen norma unitaria $\|\mathbf{p}_a\| = 1$. Se usan colores más azules intensos para valores negativos y más amarillentos para positivos. Por esta codificación de colores los colores verdosos corresponden a pesos próximos a cero y, al estar estandarizadas las variables sobre las que multiplican estos pesos, estos colores verdosos indican que esa variable no ha jugado un papel importante al determinar la componente principal a.

   Si se fija $A = 4$ para los datos Iris ($I = 150$ y $J = 4$) obtenemos el gráfico mostrado en la Figura 7.5.

3. **Varianzas explicadas:** Este gráfico es también conocido como *scree plot* y proporciona el porcentaje de la variabilidad que cada componente principal explica. Estos valores surgen de `pca.explained_variance_ratio_` desde *Scikit-Learn* y `pca=PCA()`. Se representan estos valores en forma de gráfico de barras

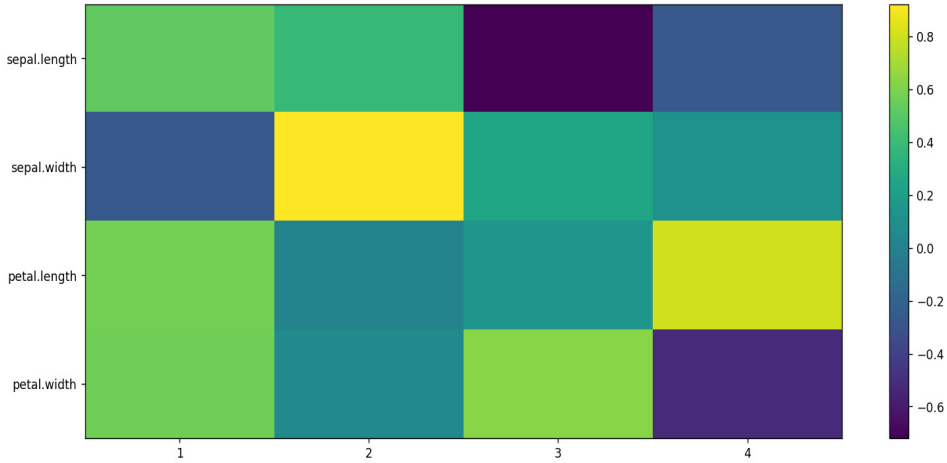


Figura 7.5: Gráfico de contribución de las variables para los datos Iris.

y encima de cada barra aparece el porcentaje de variabilidad explicada por cada componente, que coincide con

$$\frac{\lambda_a}{\lambda_1 + ... + \lambda_J},$$

para $a = 1, ..., A$. Cuando más próxima la fracción correspondiente a a esté a 1 entonces mayor será el porcentaje de variabilidad que recoge esa componente a. Se sabe que $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_A$ por lo que, por definición, las componentes están ordenadas en forma decreciente de variabilidad recogida.

Una idea razonable para seleccionar A es buscar un "codo" en este gráfico. Buscar un codo significa encontrar un valor de A pare el decrecimiento en variabilidades explicadas sea rápido hasta justo llegar a ese A (indicando que eliminar la información de esa última componente principal aún resulta en una pérdida significativa de información) pero a partir de ese A el decrecimiento es más moderado (y ya no se pierde tanto al ir eliminando componentes).

El gráfico de variabilidades explicadas para los datos Iris se muestra en Figura 7.6. Se ve que la primera componente principal es capaz por sí sola de explicar un 73 % de la variabilidad en estos datos.

4. **Varianzas explicadas acumuladas:** Se proporciona un gráfico de líneas con las varianzas acumuladas que se van explicando al aumentar el A mediante la representación de

$$\frac{\lambda_1 + ... + \lambda_a}{\lambda_1 + ... + \lambda_J},$$

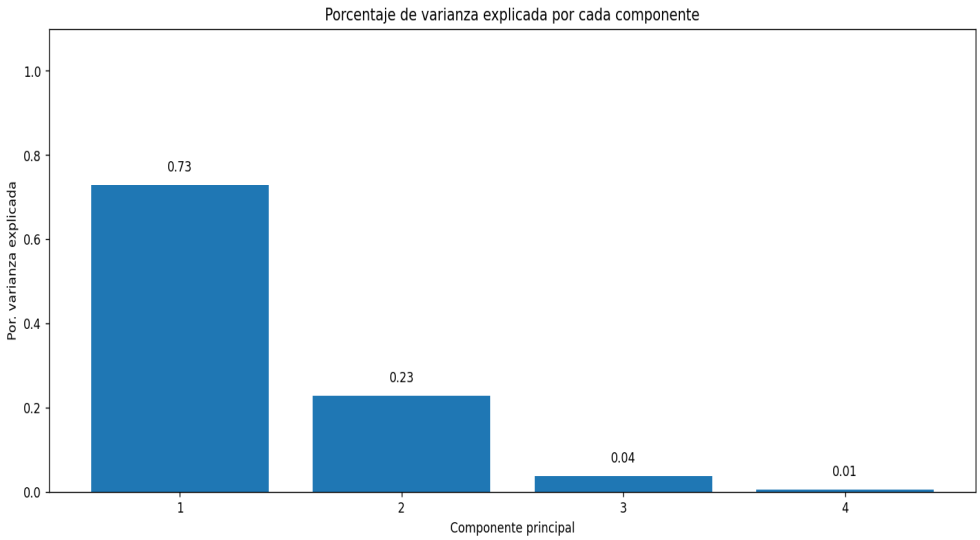


Figura 7.6: Varianzas explicadas para los datos Iris.

para $a = 1, ..., A$.

Para los datos Iris se obtiene el gráfico mostrado en la Figura 7.7, donde se puede ver que las dos primeras componentes principales (que denotamos por PC1 y PC2) explican un 96 % de la variabilidad observada en ese conjunto de datos.

Para obtener el gráfico anterior se ha usado `np.cumsum()`, proporcionando las sumas acumuladas en *Numpy*.

Como se ha comentado, el alcanzar una proporción de variabilidad explicada prefijada es uno de los criterios más habituales para elegir A y es común fijarse en cuando esta variabilidad explicada excede un determinado porcentaje, por ejemplo, del 80 % o 90 %.

5. **Gráfico de loadings:** Se representan pares de columnas j y j' de la matriz de *loadings* $\mathbf{P}_{1:A}$, donde ese par (j, j') es especificado por el usuario mediante dos menús desplegables. Estos loadings se obtienen en `pca.components_` y, por defecto, el gráfico muestra PC1 y PC2, correspondiente al caso más informativo en el que $(j, j') = (1, 2)$.

   En lugar de representar directamente las columnas $\mathbf{p}_a$ de la matriz, se ha optado por representar $\sqrt{\lambda_a}\mathbf{p}_a$. Cuando los datos están estandarizados, el gráfico de *loadings* puede interpretarse como mostrando unas "coordenadas" para cada una de las J variables originales. El hecho de multiplicar por la raíz del autovalor consigue que esta coordenada representada por cada variable pueda ser

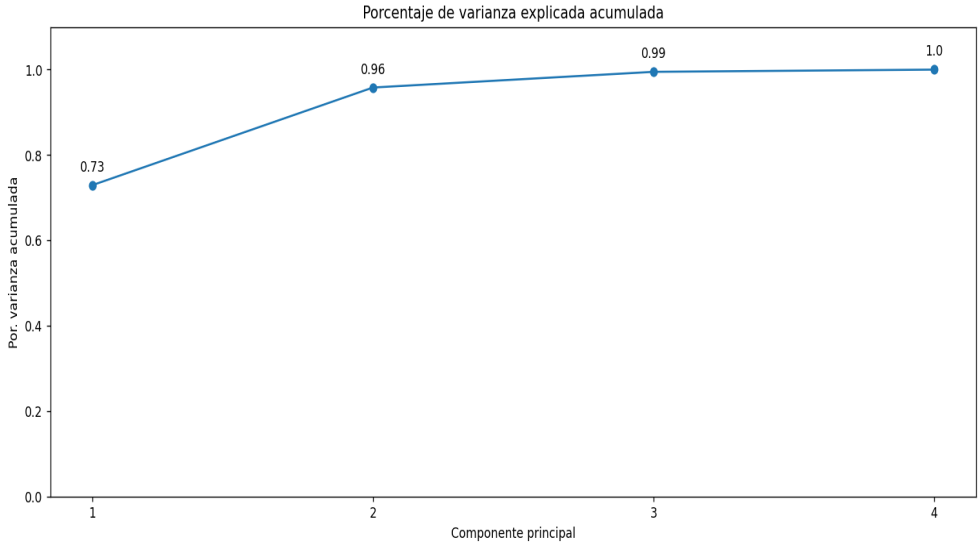


Figura 7.7: Varianzas explicadas acumuladas para los datos Iris.

interpretar como la correlación de cada variable con la componente principal a. La corrección usando autovalores ha sido programada directamente en Python puesto que no es un salida directa desde *Scikit-Learn*.

Cuando más grande (en valor absoluto) es la coordenada de una variable en este gráfico mayor peso tendrá dicha variable en determinar la componente principal. Esta representación permite visualizar cómodamente la estructura de dependencia y correlaciones de las variables originales. Si dos variables $\mathbf{x}_j$ y $\mathbf{x}_{j'}$ tienen coordenadas muy próximas en este gráfico de loadings entonces quiere decir que están muy correladas positivamente ($\text{corr}(\mathbf{x}_j, \mathbf{x}_{j'}) \approx 1$). Una alta correlación positiva indica que valores grandes de la variable $\mathbf{x}_j$ nos lleva a valores grandes de $\mathbf{x}_{j'}$ y, al revés, valores pequeños de $\mathbf{x}_j$ nos lleva a valores pequeños de $\mathbf{x}_{j'}$. Si dos variables están muy alejadas en este gráfico entonces las dos variables también están también muy correladas, pero ahora negativamente, con $\text{corr}(\mathbf{x}_j, \mathbf{x}_{j'}) \approx -1$. En este caso, valores grandes de $\mathbf{x}_j$ llevan a valores pequeños de $\mathbf{x}_{j'}$. Situaciones intermedias indican un continuo de grados de correlación. Esta explicación es la que hace que las coordenadas en este gráfico hayan sido representadas por flechas, y no por puntos, ya que implican más bien "direcciones" de correlación. Si estas flechas son perpendiculares (ángulo de 90^{o}) entonces las variables $\mathbf{x}_j$ y $\mathbf{x}_{j'}$ están próximas a ser incorreladas con $\text{corr}(\mathbf{x}_j, \mathbf{x}_{j'}) \approx 0$.

La Figura 7.8 muestra el gráfico de *loadings* asociado a las dos primeras componentes principales de los datos Iris.

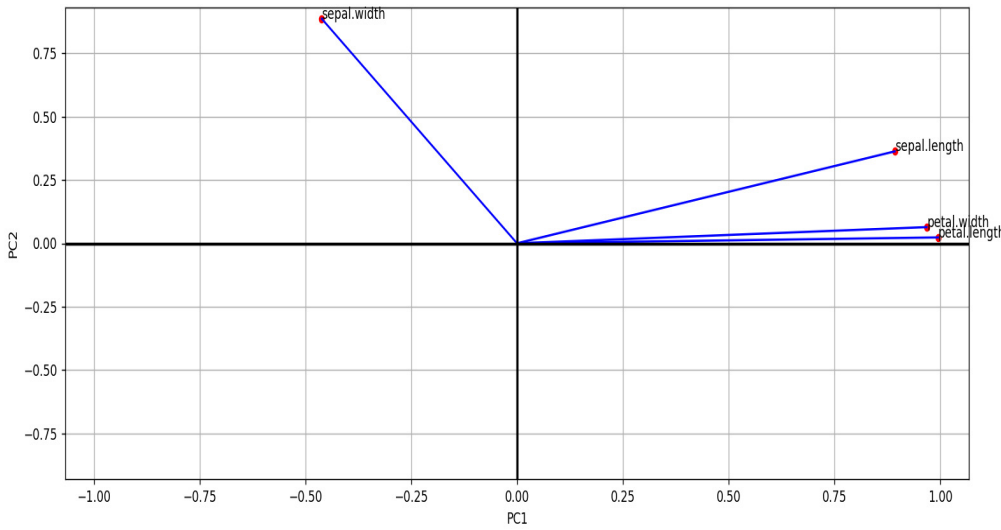


Figura 7.8: Gráfico de loadings para los datos Iris.

En la Figura 7.8 se muestra que las variables petal.length y petal.width tiene una correlación próxima a 1. Por tanto, flores con pétalos "grandes" tienen simultaneamente pétalos largos y anchos. Estas flores con pétalos grandes tienden también a tener sépalos largos (setal.length grande). No obstante, el tamaño de los pétalos no proporciona tanta información sobre el ancho de los sépalos (setal.width), ya que para nada se observan fuertes correlaciones positivas. Se verá que este ancho de los sépalos estará más asociado con poder diferenciar una especie particular de iris, las iris "setosa", respecto a las otras dos especies.

Se muestra en la Figura 7.9 un segundo ejemplo, con datos de pruebas de atletismo, desde el conjunto de datos Decathlon.

Se puede ver qué tiempos bajos en realizar los 100 metros lisos están asociados a tiempos también bajos en realizar los 110 metros vallas y en saltos de longitud largos (de hecho, no es extraño atletas que sobresalen simultáneamente en estas tres disciplinas). Las pruebas de lanzamiento de jabalina, peso y disco están correladas positivamente, indicando que si un atleta es capaz de lanzar lejos alguno de estos objetos también lo suele hacer con los otros. Se puede ver que la puntuación (points) está justo en la dirección de valores positivos de la primera componente principal PC1. Esto muestra que la primera componente PC1 ha sido capaz de diferenciar (de forma no-supervisada) los atletas mejores de los peores. Los atletas buenos (con PC1 positivos y altos) lanzan y saltan lejos y consiguen hacer tiempos bajos en pruebas en las que se mide el tiempos en recorrer distancias. Por otro lado, la segunda componente podría

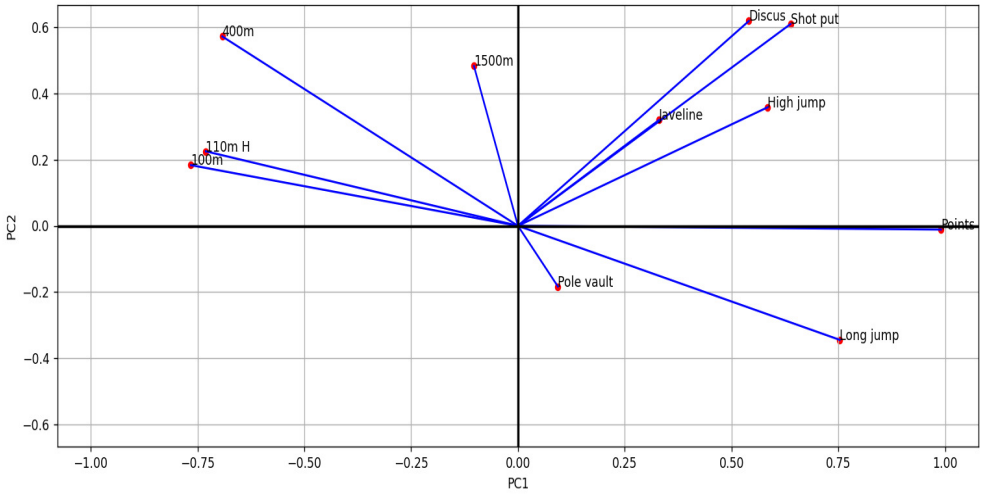


Figura 7.9: Gráfico de loadings para los datos Decathlon.

estar detectando atletas más de "fuerza" (puesto que valores de PC2 positivos y grandes están asociados a buenos lanzamientos de objetos pero tiempos mayores en hacer las pruebas de correras más largas (como pueden ser los 1500 metros). Por otro lado, atletas con menor masa muscular y menos peso pueden estar asociados a valores de PC2 negativos.

6. **Gráfico de scores:** Este gráfico representa las coordenadas o *scores* de los I individuos en los pares de componentes principales que selecciona el usuario (por defecto PC1 y PC2). Este gráfico se obtiene desde `pca.fit_transform()` de *Scikit-Learn*.

   La información proporcionada por estos scores es muy útil. Si dos individuos i e i' están próximos en sus coordenadas en las primeras componentes principales entonces estos individuos también toman valores parecidos en las J variables originales. De esta forma, estamos visualizando datos en dimensión J en una representación bidimensional. Este gráfico utiliza colores que tienen que ver con la "Etiqueta de datos" y permite ver hasta qué punto las observaciones con una misma etiqueta son similares y analizar variabilidades y separaciones entre niveles de esa etiqueta.

   Las coordenadas se obtienen mediante la expresión

$$\mathbf{T}_{1:A} = (\mathbf{t}_1|\mathbf{t}_2|...|\mathbf{t}_A) = \mathbf{X} \cdot \mathbf{P}_{1:A}.$$

   La Figura 7.10 muestra las coordenadas de las $I = 150$ flores correspondientes a los datos Iris. Se usan colores distintos para las distintas especies: "setosa",

"versicolor" y "virginica". Se ve que las flores de una misma especie toman valores parecidos en las $J = 4$ variables relativas al tamaño de sus pétalos y sépalos. Se puede ver que la especie "setosa" está mejor diferenciada de las otras dos, mientras que las flores "versicolor" y "virginica" son más parecidas entre sí.

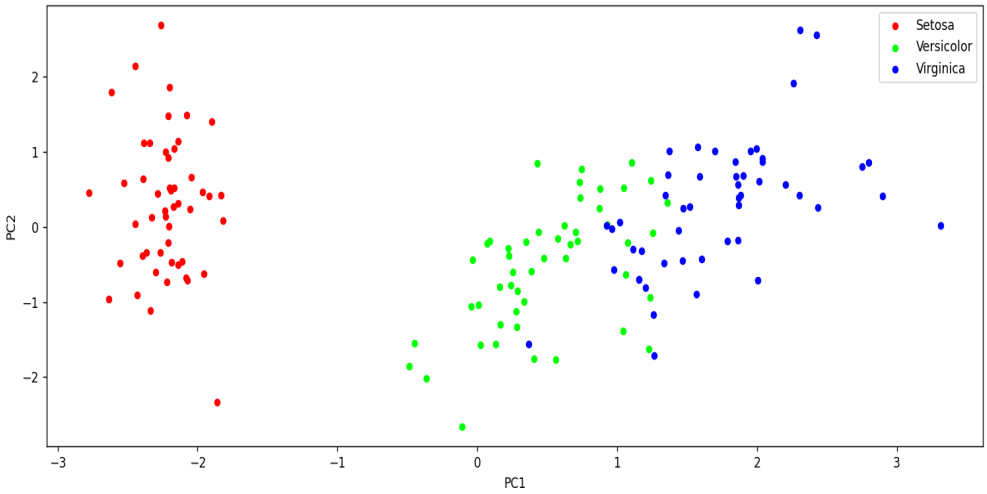


Figura 7.10: Gráfico de scores para los datos Iris.

Se podría haber hecho un *matrix plot*, donde se representa en un único gráfico todos los gráficos por pares "dos a dos" de variables, o utilizar un gráfico tridimensional de tres componentes principales. No obstante, estos gráficos no son fáciles de visualizar cuando A sea grande y el tridimensional requeriría añadir una funcionalidad de rotación de ejes para su correcta visualización.

7. **Biplot:** Es un gráfico muy útil para extraer de forma muy cómoda gran parte de la información en un conjunto de datos, incluso cuando el número de variables J sea moderadamente elevado, y salvando parcialmente las dificultades que el ser humano tiene para visualizar conjuntos de datos en dimensiones superiores a 3.

   En el gráfico *biplot* se superponen los loadings presentados en el punto 5) junto al gráfico de *scores* presentado en el punto 6) y, evidentemente, eligiendo las mismas componentes principales en ambos gráficos. El usuario puede seleccionar las componentes que desea representar, aunque la representación más útil es la que se proporciona en el programa por defecto al usar PC1 y PC2. Al representar los *scores* es posible representar los I individuos con colores distintos dependiendo de la variable de etiqueta seleccionada.

La interpretación del gráfico *biplot* es razonablemente sencilla. Si un individuo i queda en la dirección que marca la flecha correspondiente a la variable $\mathbf{x}_j$ entonces se puede decir que ese individuo i toma un valor "relativamente" alto en dicha variable $\mathbf{x}_j$. Eso no quiere decir que ese valor en la variable $\mathbf{x}_j$ sea muy elevado a nivel absoluto, pero sí que ese valor es relativamente más alto que el que toman otros individuos en nuestro conjunto de datos en esa variable $\mathbf{x}_j$. Si, por el contrario, el individuo i aparece en dirección contraria a la que marca la variable $\mathbf{x}_j$ entonces tomará un valor relativamente bajo. Nótese que esto es coherente con la representación de las correlaciones comentada en el punto 5).

En el biplot para los datos Iris representado en la Figura 7.11 se puede ver que la especie "setosa" toma valores relativamente altos en sepal.width, por lo que serán flores con sépalos más anchos. Las especies "versicolor" y "virginica" suelen tener flores con pétalos más grandes. Se puede distinguir ligeramente entre estas dos últimas especies viendo que las flores de la especie "virgina" tiene los pétalos ligeramente más grandes y, también, sépalos algo más largos que las flores de la especie "versicolor".

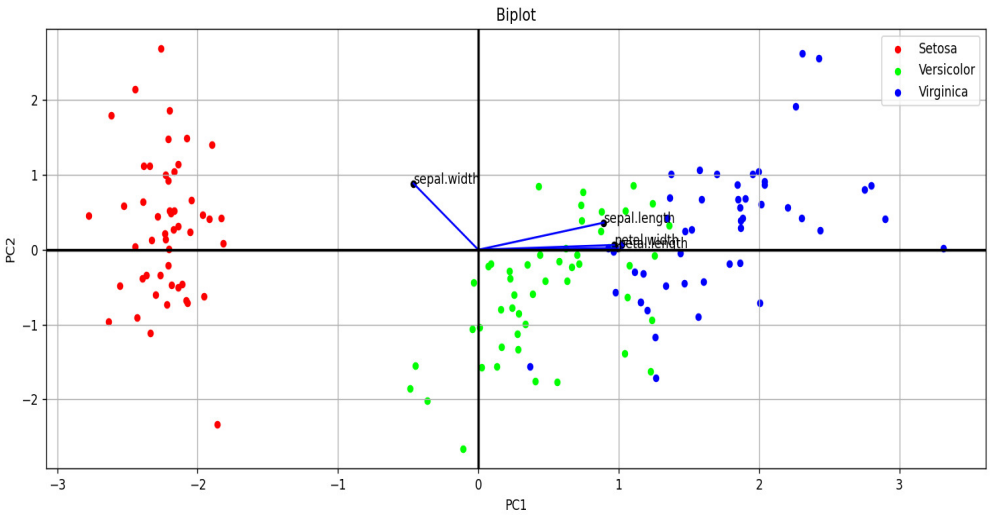


Figura 7.11: Biplot para los datos Iris.

En este ejemplo hemos visualizando en dimensión $A = 2$ unos datos que están originalmente estaban en dimensión $J = 4$, con una pérdida de información mínima (PC1 y PC2 recogen conjuntamente un 96 % de la información).

8. **Reconstrucción de datos:** Una forma alternativa de ver el PCA es pensar en reconstruir los datos originales (en dimensión J) por una representación en un subespacio de dimensión menor A. Cada una de las observaciones $\mathbf{x}_i =$

$(x_{i1}, x_{i2}, ..., x_{iJ})$ es aproximada por una predicción

$$\widehat{\mathbf{x}}_{i;1:A} = (\widehat{x}_{i1;1:A}, \widehat{x}_{i2;1:A}, ..., \widehat{x}_{iJ;1:A}).$$

Esta predicción $\widehat{\mathbf{x}}_{i;1:A}$ se obtiene mediante una combinación lineal adecuada de las columnas de la matriz $\mathbf{P}_{1:A}$. De hecho, los valores por los que se debe multiplicar estas columnas de $\mathbf{P}_{1:A}$ vienen dados por la fila i-ésima de la matriz $\mathbf{T}_{1:A}$ y esto justifica el nombre de matriz de scores o puntuaciones de $\mathbf{T}_{1:A}$.

Si $\widehat{\mathbf{X}}_{1:A}$ denota la matriz $I \times J$ con las reconstrucciones al usar A componentes (sus filas serían $\widehat{\mathbf{x}}_{i;1:A}$ para $i = 1, ..., I$), se tiene

$$\widehat{\mathbf{X}}_{1:A} = \mathbf{T}_{1:A} \cdot (\mathbf{P}_{1:A})^t.$$

Con la expresión anterior, se estaría reconstruyendo los datos centrados y escalados (ya que para el PCA se ha partido de dichos datos estandarizados). Por tanto, para obtener las predicciones de los datos originales, hay que deshacer esta estandarización sumando las medidas por columnas y multiplicando por la desviación típica de las las variables originales.

En el programa, se han representado los valores observados (círculos) para las J variables originales, en los I individuos, junto a sus valores reconstruidos (aspas) cuando se consideran A componentes principales. Se ha usado un panel *scrolleable* en la pestaña "Reconstrucción de datos" para proporcionar estos gráficos.

Para los datos Iris, la Figura 7.12 muestra la reconstrucción de los datos con $A = 2$. La reconstrucción resulta bastante razonablemente a pesar de que solo requiere almacenar 150×2 scores y 4×2 loadings, en lugar de 150×4 valores.

En un caso general, sería necesario almacenar $I \times A$ scores y $J \times A$ loadings (con A generalmente mucho menor que J) y alcanzar un ratio de comprensión aproximada igual a A/J. La calidad de esta reconstrucción coincide con el porcentaje acumulado visto en el punto 4).

Si se elige $A = J$ entonces los valores de x_{ij} y sus predicciones $\widehat{x}_{ij;1:A}$ coinciden exactamente y la calidad de la representación es del 100 %. Esto puede verse en la Figura 7.13 donde se ve que la reconstrucción es perfecta al tomar $A = J = 4$.

7.4.5. Implementación de la validación cruzada K-fold

Con el material presentado en los apartados anteriores, se está en condiciones de explicar el "gráfico R2 K-fold" que ayuda a elegir el número de componentes A usando validación cruzada y que se obtiene al marcar la opción "Validación cruzada" en "Generar análisis".

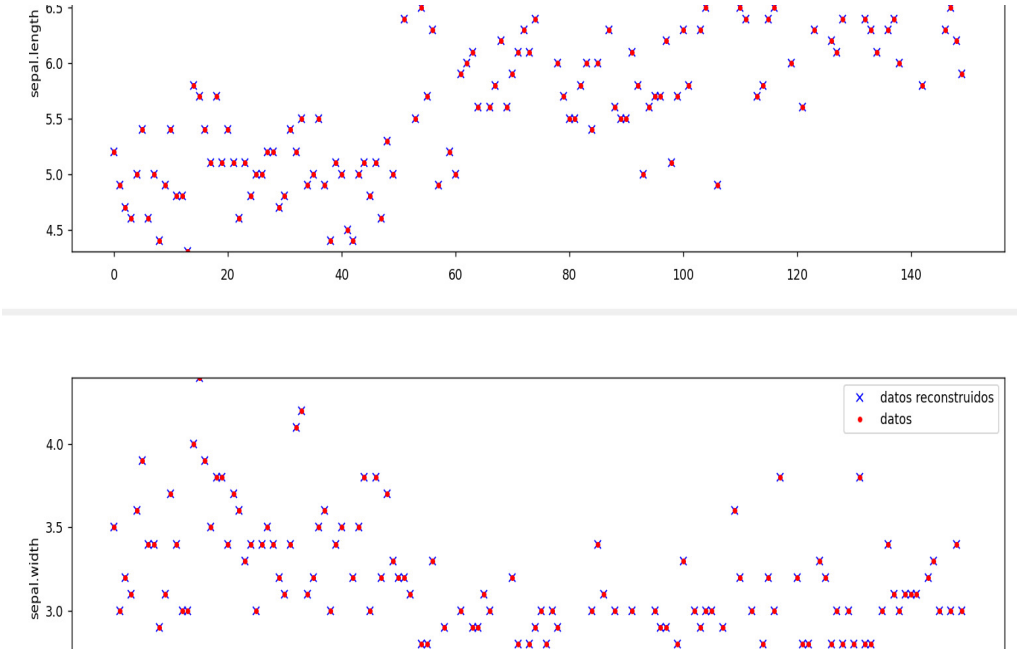


Figura 7.12: Parte del gráfico para las variables en los datos Iris junto a su reconstrucción usando $A = 2$ componentes.

Como técnica de validación cruzada, se ha seguido un procedimiento K-fold con $K = 5$. Aunque $K = 5$ y $K = 10$ son los valores de K más habituales, sería fácil añadir la funcionalidad de que el usuario pudiera elegir otros valores de K. La idea del K-fold es particionar el conjunto de datos $\mathbf{X}$ en K subconjuntos o "hojas" (*folds*) e intercambiar secuencialmente el papel de conjunto de entrenamiento y test basado en estas particiones. La partición del conjunto de datos $\mathbf{X}$ en K hojas conviene realizarse de forma aleatoria y se ha usado, con este propósito, la orden `KFold()` de la librería *Scikit-Learn*. Todo el procedimiento que se va a describir para implementar la validación cruzada K-fold ha sido programado en Python en este Trabajo Fin de Grado.

En la presentación del procedimiento de validación cruzada se usa el superíndice k al referirse a la hoja (*fold*) para $k = 1, ..., 5$, el subíndice "20" representa un 20 % de observaciones en el conjunto de prueba ($1/K = 1/5 = 0{,}20$) y el subíndice "80" representa el 80 % de observaciones en el conjunto de entrenamiento ($1 - 1/K = 1 - 1/5 = 0{,}80$). Así, $\mathbf{X}^k_{80}$ será el conjunto de entrenamiento en la hoja k y $\mathbf{X}^k_{20}$ el conjunto de prueba en dicha hoja. Nótese que $\mathbf{X}$ es la unión de $\mathbf{X}^k_{80}$ y $\mathbf{X}^k_{20}$ y que $\mathbf{X}$ es también la unión de $\mathbf{X}^1_{20}$, $\mathbf{X}^2_{20}$, ..., $\mathbf{X}^5_{20}$.

Con esta notación, un pseudo-código para implementar el procedimiento valida-

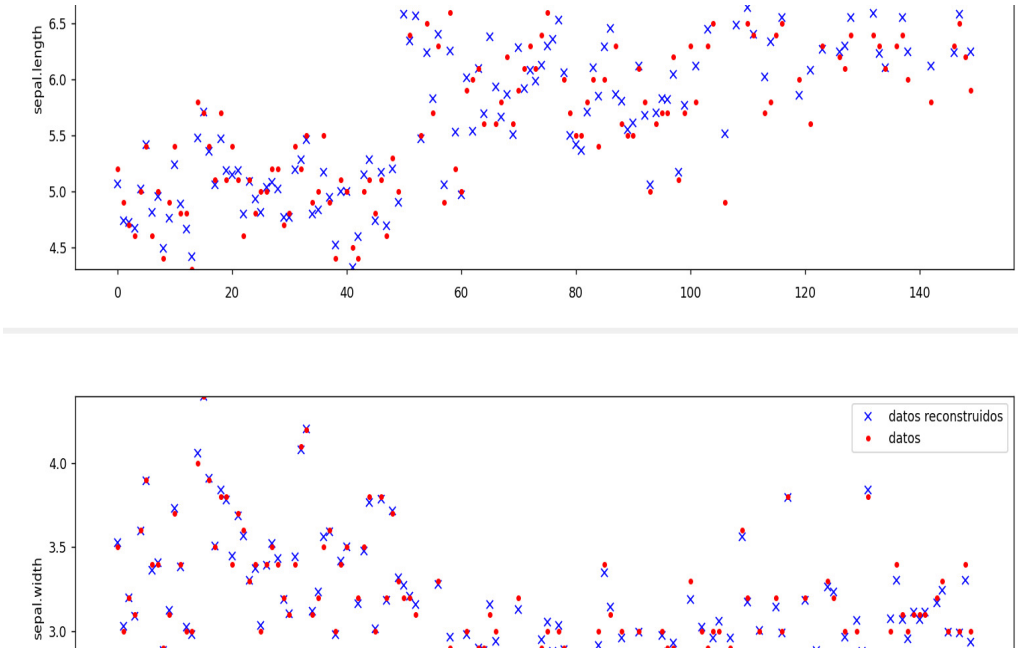


Figura 7.13: Parte del gráfico para las variables en los datos Iris junto a su reconstrucción usando $A = J = 4$ componentes.

ción cruzada es el siguiente:

1: Considerar $\{\mathbf{X}^k_{80}, \mathbf{X}^k_{20}\}_{k=1}^5$ con $\cup_{k=1}^5 \mathbf{X}^k_{20} = \mathbf{X}$ y $\mathbf{X}^k_{20} \cup \mathbf{X}^k_{80} = \mathbf{X}$
2: **for** k=1:5 **do**
3: $\quad \mathbf{P}^k_{80;1:A}$ (autovectores usando $\mathbf{X}^k_{80}$ y A componentes)
4: $\quad$ **for** A=1:J **do**
5: $\quad\quad \mathbf{T}^k_{20;1:A} = \mathbf{X}^k_{20}\mathbf{P}^k_{80;1:A}$ (*scores* del conjunto test en la hoja k)
6: $\quad\quad \widehat{\mathbf{X}}^k_{20;1:A} = \mathbf{T}^k_{20;1:A}(\mathbf{P}^k_{80;1:A})^t$ (predicciones conjunto test en hoja k)
7: $\quad\quad$ Calcular la medida de eficiencia:

$$R^{2,k}_A = 1 - \frac{\|\mathbf{X}^k_{20} - \widehat{\mathbf{X}}^k_{20;1:A}\|^2}{\|\mathbf{X}^k_{20}\|^2}$$

8: $\quad$ **end for**
9: **end for**

Como medida de eficiencia se ha usado un análogo al coeficiente de determinación R^2 en Regresión. El R^2 es el cociente entre "variabilidad explicada" y "variabilidad total" o, análogamente, uno menos el cociente entre "variabilidad no-explicada" y

la "variabilidad total". Dada una matriz $\mathbf{Y}$ de tamaño $I \times J$ con términos y_{ij}, que suponemos centrada por columnas, la variabilidad total es

$$\|\mathbf{Y}\|^2 = \sum_{i=1}^{I} \sum_{j=1}^{J} y_{ij}^2,$$

y, por tanto, $\|\mathbf{X}_{20}^k - \widehat{\mathbf{X}}_{20;1:A}^k\|^2$ es medida de lo bien que $\widehat{\mathbf{X}}_{20;1:A}^k$ sirve para "aproximar" $\mathbf{X}_{20}^k$ en la hoja k si elegimos A componentes. Para cada uno de los valores de A probados, con $A \leq J$, tendremos $K = 5$ valores del R^2 y que son denotados como $R_A^{2,1}$, $R_A^{2,2}$, ..., y $R_A^{2,5}$. Podríamos resumir estos $K = 5$ valores del K-fold para un A fijo mostrando su mediana o su media muestral pero es más conveniente utilizar un diagrama de cajas para dicho resumen. El diagrama de cajas está basado esencialmente en la mediana y el primer y tercer cuartil de los datos. La caja central de un diagrama de cajas representa el 50 % de los datos más centrales y unas barras (bigotes) marcan el rango de valores donde se pueden encontrar las observaciones menos atípicas, mientras que las observaciones detectadas como atípicas por este método se marcan con puntos aislados.

La Figura 7.14 muestra los gráficos de caja para los valores $R_A^{2,1}$, $R_A^{2,2}$, ..., y $R_A^{2,5}$ en los datos Iris cuando se prueban valores de A entre 1 y 4. Los gráficos de cajas nos dan una idea de los valores entre los que se puede mover este R^2 al aplicar principios de validación cruzada.

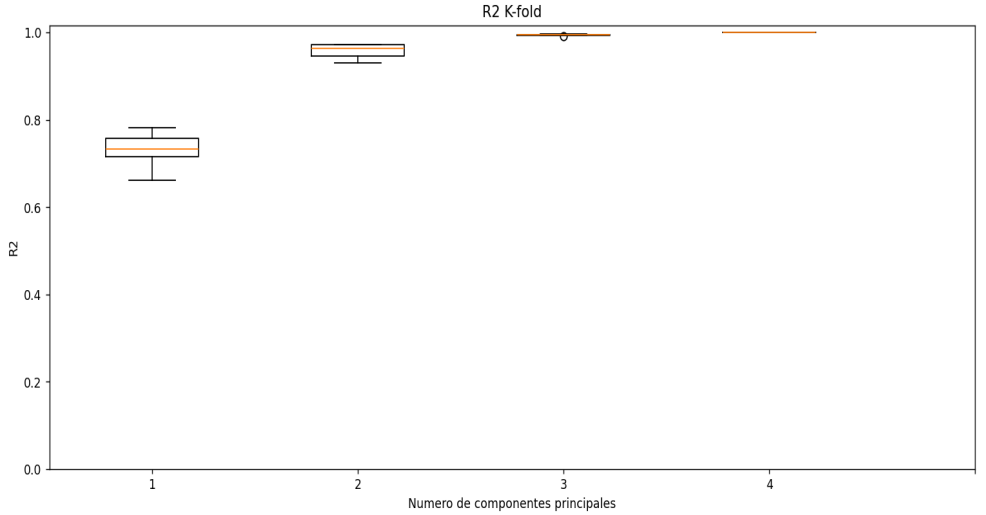


Figura 7.14: Gráfico "R2 K-fold" para los datos Iris.

El mismo procedimiento se ha aplicado a los datos Decathlon con valores de A entre 1 a 11. Los gráficos de caja asociados se muestran en la Figura 7.15.

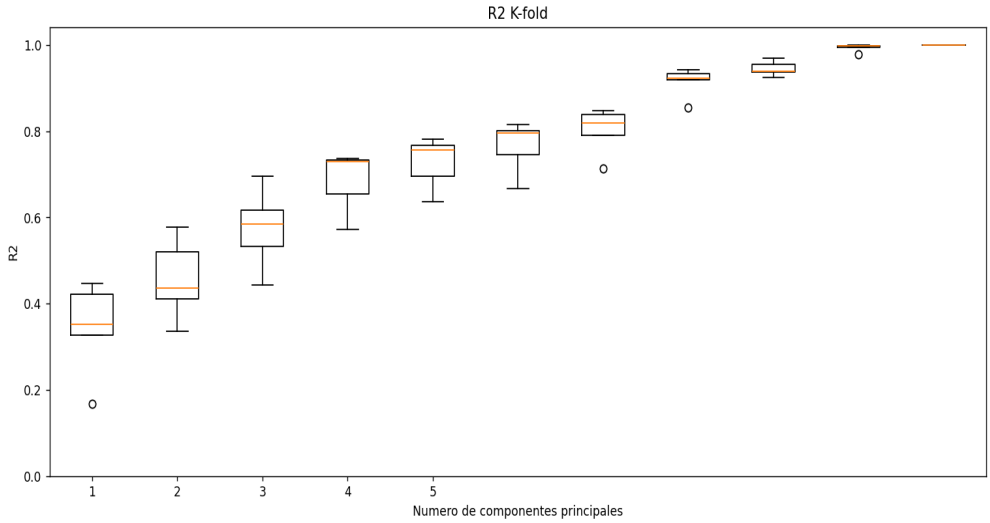


Figura 7.15: Gráfico "R2 K-fold" para los datos Decathlon.

Al contrario de lo que pasa en métodos supervisados, el procedimiento descrito no es un procedimiento perfecto de validación cruzada. De hecho, es complicado observar patrones decrecientes en estos diagramas de cajas al aumentar el A, ya que siempre se "explica" más al añadir más componentes puesto que

$$\|\mathbf{X}_{20}^k - \widehat{\mathbf{X}}_{20;1:A'}^k\|^2 \leq \|\mathbf{X}_{20}^k - \widehat{\mathbf{X}}_{20;1:A}^k\|^2$$

para $A \leq A'$. Además, al obtener $\widehat{\mathbf{X}}_{20;1:A}^k = \mathbf{T}_{20;1:A}^k(\mathbf{P}_{80;1:A}^k)^t$ se considera $\mathbf{T}_{20;1:A}^k = \mathbf{X}_{20}^k\mathbf{P}_{80;1:A}^k$ y se está "usando" el conocimiento de $\mathbf{X}_{20}^k$ para predecir $\mathbf{X}_{20}^k$. No obstante, este procedimiento de validación cruzada, aunque no permita elegir el A (por ser un método no supervisado), sí que proporciona una buena idea de la variabilidad experimental al estimar las variabilidad explicadas y es más apropiado que lo visto en el apartado 4) donde también se usaban los datos en $\mathbf{X}_{20}^k$ para obtener los scores.

Capítulo 8

Pruebas

En este capítulo se comentan brevemente los tipos de pruebas que existen en el desarrollo de software y se describirán las pruebas específicas que se han realizado para este trabajo.

8.1. Introducción

En los proyectos de desarrollo de software existen una serie de pruebas que se realizan con el fin de que el software cumpla con las condiciones especificadas en los requisitos, que no tenga fallos y que se "adapte" a las expectativas de los clientes. Estas pruebas pueden ser clasificadas por tipos o por niveles, en función de qué condiciones comprueben [Sommerville, 2005].

De esta manera, existen dos tipos principales de pruebas, las pruebas de validación cuyo objetivo es comprobar que el software cumple las características deseadas por el cliente, y las pruebas de defectos que sirven para detectar cualquier problema que presente el software.

También se pueden clasificar las pruebas en función de los niveles en los que se comprueba el software. Así, empezando desde las unidades mínimas de código, están las pruebas unitarias, las cuales verifican que las clases y los métodos realizan su función correctamente. A continuación, están las pruebas de integración, las cuales comprueban que todas estas unidades mínimas interactúen correctamente entre sí. Después de estas están las pruebas de sistema, que se realizan sobre el programa completamente implementado y cuyo objetivo es asegurarse que se cumplen todos los objetivos. Por último, están las pruebas de aceptación, que se tratan de pruebas

realizadas con los clientes para asegurar que el código no solo funciona correctamente, sino que cumple con todas las características que necesitaban.

Para este proyecto se han realizado únicamente los dos últimos niveles de pruebas, es decir, las pruebas de sistema y de aceptación.

8.2. Pruebas de sistema

Como ya se ha descrito anteriormente, las pruebas de sistema se tratan de comprobaciones que se realizan sobre el sistema una vez este ha sido implementado por completo. Para poder realizar el seguimiento de estas pruebas se van a detallar a continuación en tablas para cada caso de prueba. En cada caso de prueba, se especificarán las entradas y salidas esperadas del sistema, los pasos que se deberán seguir para realizar el caso de prueba y por último una referencia al caso de uso que se está probando.

En las Tablas 8.1 a 8.7 se muestran los casos de prueba del sistema.

Caso de prueba "Cargar Fichero"	
Identificador	CP-01
Precondiciones	Se deberá disponer de un fichero de datos con formato CSV, que posea cabecera y que no contenga campos vacíos o con valores no válidos.
Datos de entrada	Fichero de datos.
Salida esperada	Datos del fichero cargados y mostrados en una tabla.
Escenario	El usuario seleccionará la opción *archivo* del menú superior y a continuación *Abrir...* en el desplegable de este. Después se mostrará una ventana de selección de fichero desde la cual se seleccionara el fichero de datos que se desee cargar. Una vez se haya seleccionado el archivo los datos se mostraran en una tabla dentro de la aplicación, finalizando así el caso de prueba.
Trazabilidad	Este caso de prueba cubrirá la realización del caso de uso Cargar Fichero especificado en la Tabla 4.3.

Tabla 8.1: Tabla del caso de prueba CP-01 correspondiente al caso de prueba "Cargar Fichero".

Caso de prueba "Modificar Datos"	
Identificador	CP-02
Precondiciones	Se deben haber cargado los datos de un fichero.
Datos de entrada	Modificaciones en uno o varios campos de la tabla.
Salida esperada	Datos del fichero modificados sobre la tabla.
Escenario	El usuario seleccionará una celda de la tabla que quiera modificar, al hacerlo se habilitará esa celda como entrada de texto, sobre la cual se podrá editar el valor actual o borrarlo e introducir uno nuevo. Este caso de prueba se puede repetir tantas veces como modificaciones se quieran hacer.
Trazabilidad	Este caso de prueba cubrirá la realización del caso de uso Modificar Datos especificado en la Tabla 4.4.

Tabla 8.2: Tabla del caso de prueba CP-02 correspondiente al caso de prueba "Modificar Datos".

Caso de prueba "Guardar Datos"	
Identificador	CP-03
Precondiciones	Se deben haber cargado los datos de un fichero.
Salida esperada	Datos guardados en el fichero del que se cargaron los datos o en un fichero nuevo.
Escenario	El usuario seleccionará la opción *archivo* del menú superior y a continuación *Guardar* o *Guardar como...* en el desplegable de este. Si se ha seleccionado la opción *Guardar*, se guardaran los datos en el fichero del que se cargaron. Si se selecciona lo opción *Guardar como...* se mostrará una ventana de guardar fichero del sistema operativo en la cual se podrá seleccionar la ubicación y el nombre del nuevo archivo.
Trazabilidad	Este caso de prueba cubrirá la realización del caso de uso Guardar Datos especificado en la Tabla 4.5.

Tabla 8.3: Tabla del caso de prueba CP-03 correspondiente al caso de prueba "Guardar Datos".

Caso de prueba "Seleccionar Número De Componentes"	
Identificador	CP-04
Precondiciones	Se deben haber cargado los datos de un fichero y seleccionado la columna de etiquetas si la hay.
Salida esperada	Número de componentes principales para el análisis seleccionado.
Escenario	El usuario seleccionará el botón *Generar Análisis* en la parte inferior de la aplicación. Se mostrará entonces una nueva ventana para la selección del número de componentes, en la cual se marcará el método que se prefiera para calcularlo. Algunas de las opciones permiten tambien introducir un valor ya sea para un número específico de componentes o un porcentaje de varianza. Se guardará la opción marcada y finalizará el caso de prueba.
Trazabilidad	Este caso de prueba cubrirá la realización del caso de uso Seleccionar Número De Componentes especificado en la Tabla 4.6.

Tabla 8.4: Tabla del caso de prueba CP-04 correspondiente al caso de prueba "Seleccionar Número De Componentes".

8.3. Pruebas de aceptación

Como ya se ha comentado anteriormente, las pruebas de aceptación consisten en mostrar a los usuarios la aplicación implementada para validar que cumple con las características deseadas. En este proyecto, estas pruebas se han realizado durante las reuniones semanales con el tutor, en las cuales se verificaban las características que se implementaban cada semana del desarrollo.

Caso de prueba "Generar Gráfica de Validación Cruzada"	
Identificador	CP-05
Precondiciones	Se deben haber cargado los datos de un fichero y seleccionado la columna de etiquetas si la hay.
Salida esperada	Opción para generar la gráfica de validación cruzada seleccionada.
Escenario	En la ventana de selección del número de componentes principales accedida en el caso de prueba CP-04 representado en la Tabla 8.4 hay una opción marcada por defecto con la etiqueta *Validación Cruzada*. Si esta opción se deja marcada, se indicará que se desea incluir al conjunto de gráficas una dedicada a validación cruzada. Se haya dejado marcada o no el caso de prueba finalizará cuando el usuario quiera continuar utilizando la aplicación.
Trazabilidad	Este caso de prueba cubrirá la realización del caso de uso Generar Gráfica De Validación Cruzada especificado en la Tabla 4.7.

Tabla 8.5: Tabla del caso de prueba CP-05 correspondiente al caso de prueba Generar "Gráfica de Validación Cruzada".

Caso de prueba "Generar Gráficas PCA"	
Identificador	CP-06
Precondiciones	Se deben haber cargado los datos de un fichero y seleccionado la columna de etiquetas si la hay.
Salida esperada	Ventana con las gráficas del PCA.
Escenario	En la ventana de selección del número de componentes principales una vez seleccionados el método elegido para obtener estos y si se desea una gráfica de validación cruzada, el usuario seleccionará el botón *Generar*. Una vez hecho esto, se habrá añadido una pestaña que contiene lás gráficas del PCA. Si se repite el proceso seleccionando de nuevo el número de componentes principales, las gráficas del nuevo análisis se mostrarán en una nueva pestaña.
Trazabilidad	Este caso de prueba cubrirá la realización del caso de uso Generar Gráficas PCA especificado en la Tabla 4.8.

Tabla 8.6: Tabla del caso de prueba CP-06 correspondiente al caso de prueba "Generar Gráficas PCA".

Caso de prueba "Visualizar Gráfica"	
Identificador	CP-07
Precondiciones	Se deben haber generado las gráficas del PCA.
Salida esperada	Visualización de la gráfica seleccionada.
Escenario	Una vez generadas las gráficas, el usuario seleccionará la nueva pestaña creada que contiene las gráficas de PCA con el número de componentes seleccionados. La primera gráfica que se mostrará por defecto es la de contribución de las variables, salvo que se haya decidido generar la gráfica de validación cruzada, en cuyo caso será esta última la que aparecerá como predeterminada. Si el usuario quiere visualizar otra gráfica seleccionara la pestaña de la gráfica que desee ver. Si la gráfica compara dos componentes especificos, se podrá seleccionar estos componentes en una de las listas desplegables correspondientes a los ejes de las gráficas. La gráfica se actualizará al seleccionar cualquier componente del desplegable.
Trazabilidad	Este caso de prueba cubrirá la realización del caso de uso Visualizar Gráfica especificado en la Tabla 4.9.

Tabla 8.7: Tabla del caso de prueba CP-07 correspondiente al caso de prueba "Visualizar Gráfica".

Capítulo 9

Conclusiones y líneas de trabajo futuras

En este capítulo se presentarán las conclusiones de este proyecto, analizando el grado de consecución de los objetivos iniciales. También se hará una breve reflexión sobre posibles líneas de trabajo futuras a abordar partiendo del desarrollo de este proyecto.

9.1. Conclusiones

Al principio de esta memoria, se presentaron una serie de objetivos que han dirigido el desarrollo de este trabajo. En esta sección se describirá el grado de consecución de cada uno de estos objetivos, realizando la distinción entre los de desarrollo y los académicos.

9.1.1. Conclusiones de desarrollo

Como conclusión general de los objetivos de desarrollo, se ha desarrollado la aplicación que fue propuesta en el punto de partida de este proyecto, superando con éxito todos los objetivos iniciales del mismo. Para detallar estas conclusiones, se listarán siguiendo el orden en el que se presentaron los objetivos.

- Se ha desarrollado una aplicación que permite la visualización y modificación de los datos de un fichero sobre una tabla.

- La aplicación desarrollada permite establecer un modelo PCA seleccionando columnas para etiquetar los datos y permitiendo elegir el número de componentes con el que se realiza el análisis.
- La aplicación desarrollada muestra las gráficas del PCA y enfrenta correctamente las componentes seleccionadas en los gráficos que las requieren.
- Se ha documentado el proceso de desarrollo de la aplicación.

9.1.2. Conclusiones académicas

Como conclusiones académicas, este proyecto ha sido una muy buena experiencia de aprendizaje al tener que realizar las diferentes etapas del desarrollo de una aplicación software y documentar el proceso. De los objetivos académicos que se fijaron:

- Se aprendió a usar la librería Qt para desarrollar interfaces gráficas.
- Se han aplicado los conocimientos aprendidos en las asignaturas del Grado para desarrollar la aplicación y documentar el proceso.
- Se ha superado con éxito el desafío añadido de formarse previamente a la realización del proyecto en la metodología PCA.
- Se ha mejorado en el uso de Python para el desarrollo de aplicaciones.
- Se han aplicado los *frameworks* típicos de Ingeniería del Software para realizar el Análisis y el Diseño de este proyecto.

9.2. Líneas de trabajo futuro

Un extensión bastante natural de la aplicación desarrollada sería incorporar entre sus funcionalidades la posibilidad de implementar el método de *Partial Least Squares* (PLS). El PLS también busca combinaciones lineales de las columnas de la matriz de datos $\mathbf{X}$, reduciendo su dimensionalidad, pero no exactamente a como lo hace el PCA ya que sí que usa una variable (o varias variables respuestas) en la obtención de las combinaciones lineales. En este sentido, el PLS se trata de un método "supervisado" y no un método "no supervisado", como era el PCA. El método de PLS fue desarrollado por el estadístico Herman Wold [Wold, 1982]. Wold estaba interesado en modelos de regresión que pudieran manejar multicolinealidad, es decir, alta correlaciones entre

variables predictoras al predecir una variable respuesta, lo que hacían fallar la técnica de regresión lineal típicamente aplicada con ese fin. Posteriormente, fue su hijo Svante Wold y colaboradores [Wold et al., 2001], los que popularizarían el uso del PLS especialmente en la Quimiometría. La Quimiometría consiste en la aplicación de métodos estadísticos y matemáticos a la Química y es donde, sin duda, la técnica PLS es más utilizada hoy en día [Varmuza and Filzmoser, 2016].

Como se ha comentado, el PLS trabaja con combinaciones lineales de las variables originales y también se denominan por *scores* a los valores que toman estas componentes y *loadings* a los coeficientes utilizados en dichas combinaciones lineales, análogamente a los considerados en esta memoria. La representación gráfica de los *scores* y *loadings* resultantes del PLS es también muy útil para comprender de forma visual aspectos interesantes en los datos, cuando el objetivo sea predecir estas variables respuestas. El PLS y los gráficos asociados al PLS podrían ser incorporados a la aplicación desarrollada como posible línea de trabajo futura.

Otra posible línea de trabajo a considerar sería incorporar funcionalidades asociadas a técnicas estadísticas multivariantes para el Control de Calidad [Montgomery and Verbeeck, 2004]. Las técnicas estadísticas multivariantes, como el PCA (y el PLS), han probado ser de utilidad en la monitorización de procesos industriales y en la detección temprana de fallos [García Álvarez, 2013]. Estas técnicas resultan imprescindibles cuando el (alto) número de variables monitorizadas no permita el uso de las bien conocidas técnicas univariantes. Nótese que un muy elevado número de variables a monitorizar hace extremadamente complejo la monitorización a nivel individualizado de variables y que, además, se olvida la información relativa a su estructura de dependencia o la existencia de variables latentes. Sería interesante, pues, como posible línea futura de trabajo el incluir en la aplicación funcionalidades, como serían ciertos gráficos de control, que permitan visualizar cómodamente toda la información resultante del PCA, y otras técnicas estadísticas multivariantes como el PLS, en la monitorización de procesos industriales.

Apéndice A

Enlace al código

El código correspondiente a este Trabajo Fin de Grado es accesible desde:

- Repositorio del código: https://gitlab.inf.uva.es/garciat/tfg-alvaro-garcia-torres.

Bibliografía

T. Aluja-Banet and A. Morineau. *Aprender de los Datos: el Análisis de Componentes Principales. Una Aproximación desde el Data Mining.* Universidad de Barcelona, 1999.

J. Arlow and I. Neustadt. *UML 2 and the unified process: practical object-oriented analysis and design.* Pearson Education, 2005.

Balsamiq Studios, LLC. Balsamiq 4.5.2. https://balsamiq.com/, 2022. Accessed: 2024-7-02.

F. Buschmann, R. Meunier, H. Rohnert, P. Sornmerlad, and M. Stal. *Pattern-oriented software architecture: a system of patterns. Volume 1.* Wiley, 2001.

Change Vision, Inc. Astah professional. https://tecnicos.blogs.inf.uva.es/astah-professional/, 2020. Accessed: 2021-5-20.

Equipo Inkscape. Inkscape 1.3.2. https://tecnicos.blogs.inf.uva.es/astah-professional/, 2023. Accessed: 2021-4-5.

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

B. Flury and H. Riedwyl. *Multivariate Statistics: a Sractical Approach.* Chapman & Hall, 1988.

D. García Álvarez. *Monitoring, fault detection and estimation in processes using multivariate statistical techniques.* PhD thesis, Universidad de Valladolid, 2013.

A. Géron. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow.* O'Reilly Media, Inc., 2022.

GitLab Inc. Gitlab. https://gitlab.com/, 2011. Accessed: 2024-7-02.

R. B. Grady and D. L. Caswell. *Software metrics: establishing a company-wide program.* Prentice-Hall, Inc., 1987.

J. Hammersley and J. Lees-Miller. Overleaf. https://es.overleaf.com/, 2012. Accessed: 2024-7-02.

H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24:417, 1933.

G. James, D. Witten, T. Hastie, R. Tibshirani, and J. Taylor. *An Introduction to Statistical Learning: with Applications in Python*. Springer Nature, 2023.

I. Jolliffe. *Principal Component Analysis, 2nd edition*. Springer, 2002.

J. Josse and F. Husson. Selecting the number of components in principal component analysis using cross-validation approximations. *Computational Statistics & Data Analysis*, 56:1869–1879, 2012.

H. F. Kaiser. The application of electronic computers to factor analysis. *Educational and Psychological Measurement*, 20:141–151, 1960.

S. Lê, J. Josse, and F. Husson. FactoMineR: an R package for multivariate analysis. *Journal of Statistical Software*, 25:1–18, 2008.

D. C. Montgomery and D. V. Verbeeck. *Control Estadístico de la Calidad*. Limusa Wiley, 2004.

S. Pargaonkar. A comprehensive research analysis of software development life cycle (SDLC) agile & waterfall model advantages, disadvantages, and application suitability in software quality engineering. *International Journal of Scientific and Research Publications*, 13(08):345–358, 2023.

K. Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2: 559–572, 1901.

D. Peña. *Análisis de Datos Multivariantes*. McGraw-Hill España Cambridge, 2013.

Python Software Foundation. Python 3.12.2. https://www.python.org/, 2024. Accessed: 2024-7-02.

Qt Company. Qt 6.7. https://www.qt.io/, 2024. Accessed: 2024-7-02.

W. W. Royce. Managing the development of large software systems. *Proceedings of IEEE WESCON*, 26:328–388, 1970.

I. Sommerville. *Ingeniería del software*. Pearson educación, 2005.

A. Thomas and D. Barashev. Ganttproject 3.3.3309. https://www.ganttproject.biz/, 2003. Accessed: 2024-7-05.

K. Varmuza and P. Filzmoser. *Introduction to Multivariate Statistical Analysis in Chemometrics*. CRC press, 2016.

H. Wold. Soft modelling: the basic design and some extensions. *Systems under indirect observation, Part II*, pages 36–37, 1982.

S. Wold, M. Sjöström, and L. Eriksson. PLS-regression: a basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems*, 58:109–130, 2001.

Lista de Figuras

Lista de Tablas