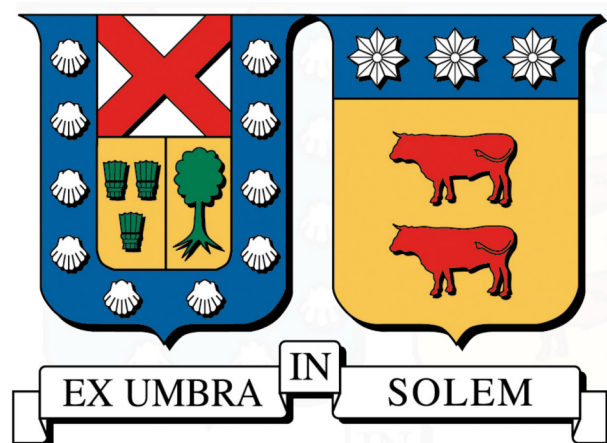


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAISO - CHILE



**PIPELINE PARA ANÁLISIS DE ESPECTROS ELECTROMAGNÉTICOS DE ENANAS
BLANCAS UTILIZANDO REDUCCIÓN DE DIMENSIONALIDAD Y CLUSTERING EN
ESPACIOS LATENTES DE REDES NEURONALES.**

CRISTÓBAL GABRIEL ORTIZ LARA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELECTRÓNICO

PROFESOR GUÍA : SR. MAURICIO ARAYA L.
PROFESOR CORREFERENTE : SR. WERNER CREIXELL F.

JULIO 2023

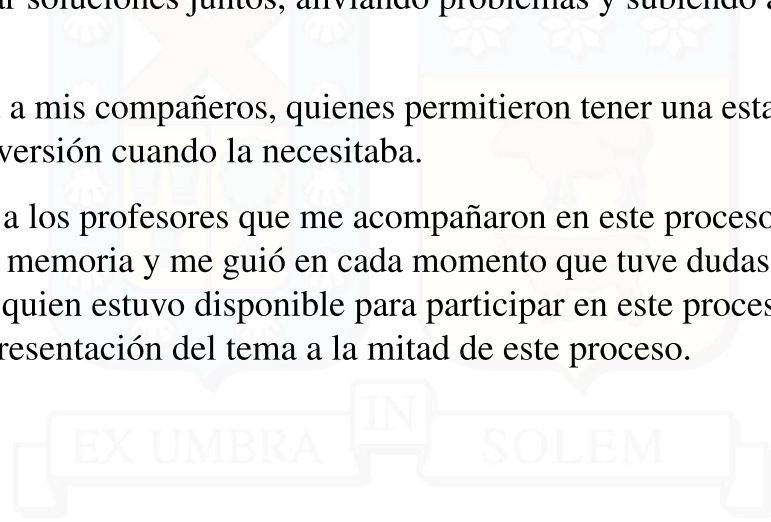
AGRADECIMIENTOS

Me gustaría expresar mis agradecimientos a todas las personas que formaron parte del proceso del desarrollo de esta memoria. Donde el apoyo de ellos ha sido fundamental para sacar adelante los momentos difíciles.

También quisiera reconocer y agradecer a mi familia, por su colaboración y comprensión cuando tuve complicaciones a lo largo del este proyecto y aún más a lo largo de la carrera, por su constante ayuda y dispuestos a escuchar cada uno de mis problemas para buscar soluciones juntos, aliviando problemas y subiendo ánimos cuando realmente lo necesité.

Además, extender esta gratitud a mis compañeros, quienes permitieron tener una estancia grata en la universidad, brindando ayuda cuando la pedía y diversión cuando la necesitaba.

Finalmente agradecer también a los profesores que me acompañaron en este proceso, mi profesor guía Mauricio Araya quien me brindó el tema de esta memoria y me guió en cada momento que tuve dudas respecto al proyecto, y a mi profesor correferente Werner Creixell quien estuvo disponible para participar en este proceso desde el primer momento y brindando retroalimentación en la presentación del tema a la mitad de este proceso.



Índice de Contenidos

1. Introducción	1
2. Trabajo a desarrollar y resultados esperados	3
2.1. Objetivos del trabajo	3
2.2. Etapas del trabajo	4
2.2.1. Definición del tipo de entrada de datos	4
2.2.2. Elección del método de reducción de dimensionalidad y clustering	4
2.2.3. Estudio sobre sinergia entre los métodos posibles	4
2.2.4. Proponer un tipo de salida.	5
2.3. Evaluaciones a realizar	5
2.4. Resultados esperados	5
3. Estado del Arte	6
3.1. Temas relevantes a tratar en el trabajo de título	6
3.1.1. Espectroscopía astronómica	6
3.1.1.1. Espectro de flujo vs Longitud de Onda	6
3.1.2. Reducción de dimensionalidad	7
3.1.2.1. Lineales	8
3.1.2.2. No lineales	8
3.1.3. Clustering	8
3.2. Análisis de trabajos desarrollados por otros autores	9
3.2.1. ASPECT: A spectra clustering tool for exploration of large spectral surveys	9
3.2.2. Spectral classification of quasars in the sloan digital sky survey: eigenspectra, redshift, and luminosity effects	10
4. Alternativas de solución del trabajo	11
4.1. Etapa de reducción de dimensionalidad	11
4.1.1. Principal Component Analysis (PCA)	11
4.1.2. T-distributed Stochastic Neighbor Embedding (t-SNE)	11
4.2. Etapa de clustering	12
4.2.1. K-Means	12
4.2.2. DBSCAN	13
4.2.3. Mean-Shift	14
4.2.4. Agglomerative Nest (AGNES)	14
4.2.5. Spectral clustering	15
4.3. Visualización de la salida	15
4.3.1. Gráfico de dispersión en 2 dimensiones	15
4.3.2. Gráfico de barras	16
5. Solución del proyecto	17
5.1. Relevancia de las soluciones en base a características	17

5.1.1.	Reducción de dimensionalidad	17
5.1.2.	Clustering	17
5.1.3.	Representación gráfica.	18
5.2.	Solución a implementar	18
6.	Desarrollo de la solución	20
6.1.	Ambiente utilizado para el desarrollo de la solución	20
6.2.	Métricas a evaluar durante el desarrollo	20
6.3.	Desarrollo de las etapas planteadas	21
6.3.1.	Extracción de datos del espacio latente	21
6.3.2.	Reducción de dimensionalidad vía PCA	23
6.3.3.	Clustering	25
6.3.3.1.	K-Means	25
6.3.3.2.	DBSCAN	27
6.3.3.3.	Spectral Clustering luego de DBSCAN	30
6.3.4.	t-SNE	32
6.3.5.	Representación gráfica	34
6.3.6.	Tiempos finales de procesamiento	36
7.	Conclusiones y posibles trabajos a futuro	37
7.1.	Conclusiones	37
7.2.	Posibles trabajos a futuro	38
	Bibliografía	39

Índice de Tablas

6.1. Librerías utilizadas para la implementación de las soluciones.	20
6.2. Alternativas desarrolladas.	21
6.3. Tiempos finales de procesamiento para cada solución desarrollada.	36

Índice de Figuras

1.1. Secuencia principal para el espectro de estrellas.	1
1.2. Diagrama general del pipeline a desarrollar.	2
3.1. Gráfico de Flujo vs Longitud de Onda señalando distintos elementos según la magnitud del flujo a determinada longitud de onda obtenido de CAS SDSS.	7
3.2. Función de aprendizaje del radio (t) y función de tasa de aprendizaje (t).	9
4.1. Aplicando PCA a MNIST.	12
4.2. Aplicando t-SNE a MNIST.	12
4.3. Procedimiento del algoritmo K-Means.	13
4.4. Procedimiento del algoritmo DBSCAN.	13
4.5. Procedimiento del algoritmo Mean Shift.	14
4.6. Procedimiento del algoritmo AGNES.	15
4.7. Comparación entre K-Means y Spectral clustering en espectros complejos.	15
6.1. Capas del modelo pre-entrenado con el cual se realizan las pruebas.	22
6.2. Gráficos y datos luego de la etapa de PCA en la capa <i>dense_13</i>	24
6.3. Gráficos y datos luego de la etapa de PCA en la capa <i>dense_14</i>	24
6.4. Gráficos y datos luego de la etapa de PCA en la capa <i>dense_15</i>	25
6.5. K-means para datos de salida de PCA en la capa <i>dense_13</i>	26
6.6. K-means para datos de salida de PCA en la capa <i>dense_14</i>	26
6.7. K-means para datos de salida de PCA en la capa <i>dense_15</i>	27
6.8. DBSCAN luego de la etapa de PCA en la capa <i>dense_13</i>	28
6.9. DBSCAN luego de la etapa de PCA en la capa <i>dense_14</i>	29
6.10. DBSCAN luego de la etapa de PCA en la capa <i>dense_15</i>	29
6.11. Spectral clustering para datos de salida de DBSCAN en la capa <i>dense_13</i>	30
6.12. Spectral clustering para datos de salida de DBSCAN en la capa <i>dense_14</i>	31
6.13. Spectral clustering para datos de salida de DBSCAN en la capa <i>dense_15</i>	31
6.14. t-SNE a la salida de cada algoritmo de Clustering en la capa <i>dense_13</i>	32
6.15. t-SNE a la salida de cada algoritmo de Clustering en la capa <i>dense_14</i>	33
6.16. t-SNE a la salida de cada algoritmo de Clustering en la capa <i>dense_15</i>	33
6.17. Gráficos de barra con cantidad de datos para cada cluster generado en la capa <i>dense_15</i>	35
6.18. Matrices de confusión para algoritmos con DBSCAN y K-Means en la capa <i>dense_15</i>	35
6.19. Matriz de confusión para solución con Spectral clustering en la capa <i>dense_15</i>	36

RESUMEN

Este proyecto forma parte de un trabajo más grande, el cual consiste en la clasificación de estrellas, más específicamente del tipo de enanas blancas, a través de un proceso que trabaja con aprendizaje supervisado, es decir, trabaja con datos previamente etiquetados provenientes de la base de datos de SDSS, donde esto permite entender la evolución de las estrellas, su composición química y como influye en la clasificación, su estructura interna, entre otras características relevantes sobre ellas.

Así, todo este proceso se ha trabajado previamente a través de clases que ya se han definido gracias al trabajo supervisado. Sin embargo, en busca de una validación para este proceso nació la necesidad de este proyecto, donde se necesita realizar un pipeline para esta clasificación a través de aprendizaje no supervisado para obtener resultados que puedan ser comparados con los del proyecto mayor y permitir a los usuarios finales sacar conclusiones al respecto. Este trabajo se enfoca principalmente en el uso de distintos tipos de clustering y reducciones de dimensionalidad para conformar el pipeline, donde se busca que este sistema sea lo más completo, efectivo e integral posible, a la vez de cumplir con los objetivos que más adelante se expliquen.

A través de este desarrollo del proyecto se obtienen resultados dentro de los objetivos que se proponen, con un sistema que toma cuestión de segundos en ejecutarse incluso para grandes cantidades de datos, clasificando en la gran mayoría de las veces un número cercano al que se conocen por el proyecto mayor que ronda entre los 10 y 12 clusters para los casos relevantes, y también un uso práctico para el usuario final.

Estos resultados permiten ampliar el análisis a realizar sobre la información original, cuestionar ciertas clasificaciones de estrellas en base a lo obtenido en los diversos gráficos y buscar el porqué de cada clasificación de las distintas estrellas. Además, en base a estos resultados se puede apreciar que si bien son procesos que atacan la misma problemática inicial desde distintos puntos de vista, son sistemas que obtienen resultados similares de los cuáles se le permite al usuario sacar conclusiones al respecto con mayor cantidad de información.

ABSTRACT

This project is part of a larger work, which involves the classification of stars, specifically white dwarfs, through a process that uses supervised learning, meaning it works with pre-labeled data from the SDSS database. This allows understanding the evolution of stars, their chemical composition, how it influences the classification, their internal structure, and other relevant characteristics.

Thus, this entire process has been previously worked on through classes that have already been defined thanks to supervised learning. However, in search of validation for this process, the need for this project arose, where a pipeline for this classification through unsupervised learning is needed to obtain results that can be compared with those of the main project and allow end-users to draw conclusions about it.

This work mainly focuses on the use of different types of clustering and dimensionality reduction to build the pipeline, aiming to make this system as comprehensive, effective, and integrated as possible, while also achieving the objectives outlined later.

Through the development of this project, results are obtained within the proposed objectives, with a system that takes only a few seconds to execute even for large amounts of data, classifying, in the vast majority of cases, a number close to what is known from the main project, ranging between 10 and 12 clusters for relevant cases, and also providing practical use for the end-users.

These results allow expanding the analysis to be carried out on the original information, questioning certain star classifications based on the findings in various graphs, and searching for the reasons behind each classification of the different stars. Furthermore, based on these results, it can be appreciated that while they are processes that tackle the same initial problem from different points of view, they are systems that obtain similar results, allowing the user to draw conclusions about them with even more information available.

1 | Introducción

Actualmente el conocimiento sobre el universo es acotado, dejando grandes ambigüedades por la infinitud del tamaño de este. Sin embargo, podemos establecer un punto de partida o referencia para comenzar a analizar este vasto espectro y todo lo relacionado a ello: la materia, la energía y la relación e interacción entre ellas. Así, nos podemos encontrar distintos cuerpos que son creados a partir de la interacción entre la materia y la energía que producen, las estrellas y sus tipos. Gracias a la conocida Secuencia Principal podemos observar y clasificar a las estrellas según su temperatura y luminosidad como se aprecia en la figura 1.1:

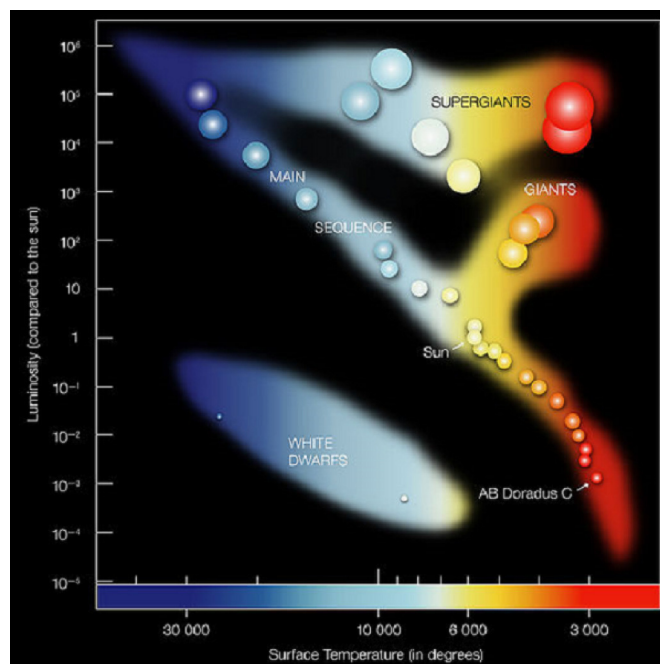


Figura 1.1: Secuencia principal para el espectro de estrellas.

En la secuencia principal, podemos observar diversas estrellas que se encuentran a lo largo del universo, incluyendo grupos como las gigantes, las supergigantes y las enanas blancas, siendo este último el enfoque principal. Asimismo, las estrellas han logrado ser clasificadas según distintos tipos de parámetros, como temperatura, luminosidad, distancia, entre otros. De esta forma, para las enanas blancas y la investigación que se está llevando a cabo será en base a los componentes químicos que las conforman. Estos pueden ser apreciados en base a un espectro que nos permite apreciar emisiones/absorciones de luz, las cuales nos indican un determinado elemento químico que está presente en el cuerpo en cuestión.

Este espectro corresponde justamente a la entrada de este proyecto y del trabajo mayor, donde este entra como un archivo que contiene las distintas magnitudes de flujo para las distintas longitudes de onda en distintos *Arrays*.

Con ello, podemos clasificar las enanas blancas a través de una red neuronal que, donde entregándole como input una imagen de un espectro obtenida de bases de datos como SDSS, entregará una clasificación final expresando al tipo de enana blanca más probable al que pertenece.

Por lo tanto, se plantea la necesidad de desarrollar un pipeline para analizar los espectros electromagnéticos mediante técnicas de reducción de dimensionalidad y clustering, centrándose en el enfoque de aprendizaje no supervisado. La idea de implementar un pipeline surge en función de su objetivo y destinatarios, ya que busca facilitar la comparación de resultados, validar los resultados previamente obtenidos y garantizar la eficiencia del proceso.

De esta manera, se presenta el diagrama de secuencia que seguirá el pipeline es:



Figura 1.2: Diagrama general del pipeline a desarrollar.

El sistema consistirá en las siguientes etapas secuenciales:

- Primero se deberán extraer los datos de la red neuronal principal desde su espacio latente, donde será necesario poder observar la dimensión de los datos y la cantidad de ellos para determinar con qué se trabajará.
- Comenzamos con una etapa crucial de reducción de dimensionalidad, cuyo objetivo es reducir la cantidad de características presentes en los datos ingresados al sistema, permitiéndonos trabajar únicamente con la información más relevante para nuestro propósito.
- Después la etapa de clustering, donde a través de la salida de la etapa anterior se agruparan los datos según distintos parámetros dependiendo del método que se utilice, lo que se explicará en profundidad más adelante.
- Finalmente, luego de estas etapas lo último es representar esto gráficamente y entregar los resultados relevantes del sistema, mostrando de la manera más clara, cómoda y eficiente el trabajo realizado a lo largo de la ejecución de este.

Cada una de estas etapas se explicará en más profundidad en la sección 2.

2 | Trabajo a desarrollar y resultados esperados

2.1. Objetivos del trabajo

La finalidad de este proyecto es ofrecer una alternativa al modelo actual con el cuál se clasifican las enanas blancas desde otra perspectiva, por lo que, para poder cumplir con este objetivo se deben cumplir las siguientes condiciones:

- **Tiempos de procesamiento cortos:** El tiempo desde que se inicia la lectura de los datos en la entrada del sistema hasta que este entrega una visualización de los resultados sea en la escala de los segundos, ya que este proceso debe ser veloz como fue solicitado en un inicio al corresponder a una alternativa *dinámica* del modelo principal. Lo ideal es que el tiempo de procesamiento se encuentre en la escala de los segundos o bien en los pocos minutos. Es difícil establecer un tiempo de procesamiento máximo en particular ya que esto también dependerá con la cantidad de datos que se quieran procesar o las características del sistema donde se ejecute el pipeline, y esto se escapa del alcance de este proyecto.
- **Pipeline correctamente particionado y general:** Se busca que esta alternativa sea lo más ordenada posible, definiendo claramente sus etapas (Extracción de datos - Red. dimensionalidad - Clustering - Resultados), lo que permitirá entender más rápidamente el proceso, y además de facultar la detección de errores y así poder atacarlos lo más pronto posible y de la manera más directa sin la necesidad de cambiar otros fragmentos del sistema. Además, el hecho de realizar un sistema ordenado facilitará su uso para el usuario final al que va destinado este pipeline.
- **Cantidad de clusters detectados:** El clasificador principal de enanas blancas consta de 12 tipos distintos. Si bien no se requiere obtener exactamente esa cantidad, se busca obtener una cifra cercana para obtener resultados coherentes. Por lo tanto, se consideraría aceptable un margen de error de ± 3 clusters con respecto al modelo principal. Optar por un número menor sería excesivamente preciso, ya que el objetivo es obtener información a través del proceso de este pipeline. Sin embargo, seleccionar un número mayor permitiría demasiada flexibilidad, lo que podría conducir a resultados extremos con muy pocos clusters o, por el contrario, a la detección de demasiados clusters sin sentido. Es importante destacar que esto dependerá de la solución desarrollada y variará según los algoritmos utilizados.

2.2. Etapas del trabajo

Como se mencionó anteriormente, el proyecto sigue un flujo como se aprecia en el diagrama 1.2, en el cual se observan las distintas etapas del pipeline para las cuales se seguirán los siguientes pasos a desarrollar:

2.2.1. Definición del tipo de entrada de datos

Para determinar que tipo de entrada requerirá el sistema, se definen los distintos casos en los cuales se podrá a prueba el sistema. Esto es específicamente como se explica en el mismo título del proyecto en cuestión, en el espacio latente del modelo del que formará parte, por lo que se debe investigar una forma de incorporar los datos que se extraerán en dicha zona del espacio latente. Para esto se buscarán soluciones centradas en la generalidad del sistema, es decir, que permita al sistema ser más flexible ante distintas entradas y no se enfoque solo en una capa o zona en específico.

2.2.2. Elección del método de reducción de dimensionalidad y clustering

Para poder decidir que método utilizar para cada una de estas etapas principales del pipeline a desarrollar, debemos considerar los distintos tipos de reducción de dimensionalidad y clustering factibles para nuestro caso, donde se buscará priorizar los que cumplan con nuestros objetivos explicados en la sección 2.1.

Con esto, nos centraremos principalmente en la efectividad, complejidad y tiempo de procesamiento que tome cada alternativa para poder definir la factibilidad y rendimiento de cada una de ellas, lo que se explicará más adelante (ver sección 5.1).

Por lo tanto, se implementarán distintos métodos para cada etapa y se probará su rendimiento en cuanto a lo mencionado anteriormente y se realizará una comparación de estos, con el fin de elegir el método que más se acomode a los objetivos propuestos o bien, entregar un sistema lo más completo posible.

2.2.3. Estudio sobre sinergia entre los métodos posibles

Dado que nuestro sistema cuenta con etapas bien definidas, es crucial que trabajen en conjunto para generar la mejor respuesta o solución posible para el proyecto. Por lo tanto, no podemos buscar la mejor solución para cada etapa de manera independiente, ya que eso no beneficiaría al sistema. En cambio, debemos enfocarnos en entender cómo funcionan las distintas combinaciones de alternativas y buscar la razón o información detrás de la sinergia entre ellas.

En lugar de realizar pruebas exhaustivas para cada combinación posible, nuestro enfoque será comprender cómo y por qué estas alternativas se complementan o interactúan de manera efectiva. Esto nos permitirá obtener y comprender las soluciones que mejor se ajustan a nuestros objetivos.

Esto se puede corroborar a través de la calidad y cantidad de clusters generados luego del proceso de ambas etapas, analizando si son acordes a lo deseable y conservando la relevancia de la salida en la etapa final.

2.2.4. Proponer un tipo de salida.

La salida es una de las partes más sencillas de implementar pero una de las más importantes del sistema. Si bien la etapa de clustering entregará una salida predeterminada (Gráfico de dispersión en 2 dimensiones) se debe adecuar para que sea lo más cómodo, fácil de entender y útil para el usuario final. Con esto, se debe buscar y proponer otros tipos de salida que permitan entregar información valiosa que no considera el gráfico de dispersión en 2 dimensiones.

2.3. Evaluaciones a realizar

Las evaluaciones para determinar el rendimiento de cada alternativa y comprobar su correcto funcionamiento se probarán inicialmente con un dataset de prueba que corresponderá a MNIST(1), esto nos permitirá tener una idea de la rapidez del sistema con las distintas alternativas y escalarlo a la hora de utilizar los datos reales que utiliza el modelo principal del cual formará parte este pipeline.

Cabe destacar que las pruebas se harán por cada parte independiente e integradas como solución completa en dicho orden, con la idea de comprobar el funcionamiento correcto de cada sección del pipeline que se desarrollará y en caso de que se nos presenten errores poder solucionarlos a la brevedad.

Sin embargo, es importante aclarar que las pruebas realizadas utilizando el dataset de MNIST serán meramente experimentales y no se mostrarán como resultados en este documento. Únicamente se presentarán los resultados obtenidos a través del dataset específicamente destinado para este propósito el cuál se detalla en la sección 6.1.

2.4. Resultados esperados

La idea principal de este proyecto es ofrecer una alternativa diferente para la clasificación de enanas blancas mediante el uso de aprendizaje no supervisado. Aunque este método difiere del modelo principal de clasificación, se espera obtener resultados similares, especialmente en cuanto a la cantidad de clusters, como se mencionó en los objetivos del trabajo (ver sección 2.1). La generación de clusters, su cantidad y formación serán factores cruciales para la utilidad de esta alternativa para el usuario final. Además, la salida del pipeline debe proporcionar información relevante para comparar con los resultados del modelo de aprendizaje supervisado.

Dicho esto, se esperan resultados similares aunque no necesariamente iguales al modelo principal para permitir a los usuarios finales llegar a analizar y obtener información desde otro punto de vista que no poseían anteriormente.

3 | Estado del Arte

El presente proyecto aborda una amplia gama de temas, desde machine learning en general hasta la clasificación de enanas blancas. A continuación, se introducirán los aspectos relevantes que se consideran en el trabajo de título, así como la información proporcionada por otros autores que resulta importante para este proyecto.

3.1. Temas relevantes a tratar en el trabajo de título

3.1.1. Espectroscopía astronómica

La espectroscopia corresponde al estudio de la interacción entre la radiación electromagnética y la materia, y cuando lo asociamos a la parte astronómica nos centramos en el efecto que causa esta radiación hecha por las estrellas y otros objetos celestes, que permiten determinar su composición química, movimiento, distancia, entre otras características, esto mediante el efecto Doppler (2). La espectroscopia nos permite conocer las distintas variables de las estrellas, lo que nos entrega información para investigar diversos fenómenos ocurridos por la radiación electromagnética de las estrellas, como lo es el corrimiento al rojo (3).

En la espectroscopía astronómica nos podemos encontrar con diversas maneras de representar resultados o características de los cuerpos celestes y uno de ellos es el espectro de flujo vs longitud de onda, el cuál tomará gran relevancia en este trabajo de título que se explica a continuación.

3.1.1.1. Espectro de flujo vs Longitud de Onda

El espectro de flujo vs longitud de onda es una representación gráfica que nos permite apreciar cuanta energía pasa sobre una determinada unidad de área llamado flujo para una determinada longitud de onda. A través del gráfico se puede determinar la composición química de una estrella dada por lo *peaks* o ciertas magnitudes de flujo que este gráfico presente a ciertas longitudes de onda, como se puede ver en el siguiente ejemplo:

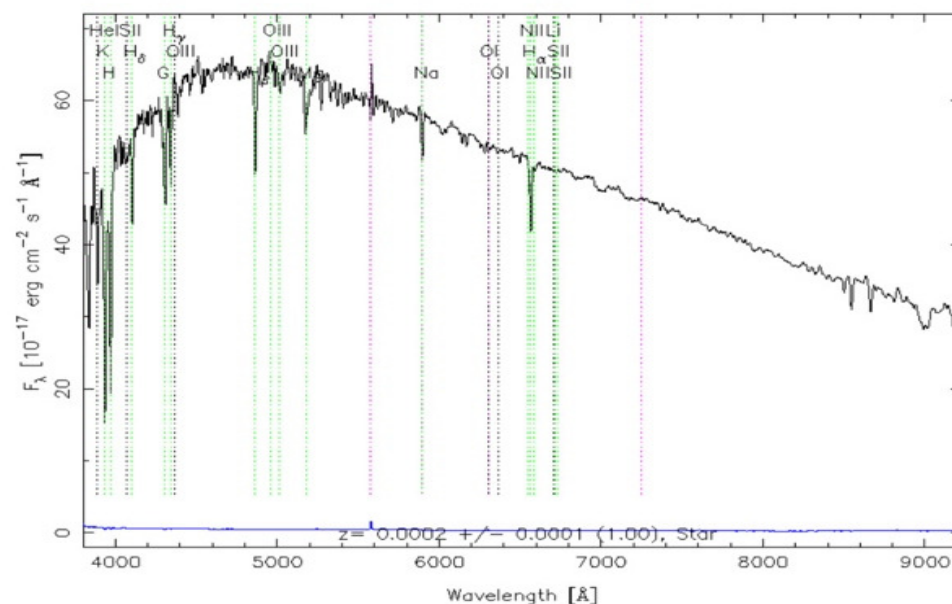


Figura 3.1: Gráfico de Flujo vs Longitud de Onda señalando distintos elementos según la magnitud del flujo a determinada longitud de onda obtenido de CAS SDSS.

Se puede apreciar en el gráfico 3.1 las unidades de medida de los ejes, donde en el eje y es $\frac{10^{-17} \text{ erg}}{\text{cm}^2 \cdot \text{s} \cdot \text{\AA}}$, unidades que se utilizan para medir el flujo o energía de radiación electromagnética que pasa a través de un área. Por su parte, el eje x se mide en $\text{\AA}$ que corresponde a la unidad ángstrom, equivalente a 0.1 [nm].

3.1.2. Reducción de dimensionalidad

La reducción de dimensionalidad nos permite trabajar con modelos que involucran conjuntos de datos de gran magnitud, haciéndolos más manejables y comprensibles. Al reducir la dimensionalidad, podemos apreciarlos de una forma más clara y accesible, incluso llevándolos a 2 o 3 dimensiones, lo que facilita la representación gráfica y mejora su comprensión.

Así, en los modelos tendremos diversas variables a las cuales llamamos características. De esta manera, la funcionalidad de la reducción de dimensionalidad es detectar y eliminar las características redundantes/poco relevantes que disminuyen el rendimiento del modelo de aprendizaje automático. La reducción de dimensionalidad otorga ciertas ventajas importantes en un modelo:

1. El hecho de eliminar características redundantes y/o poco relevantes posibilita la reducción de espacio de almacenamiento para almacenar datos.
2. Al tener menos características, es más sencilla la visualización de resultados al poseer menos dimensiones.
3. Además, al tratar con menos datos al reducir características el proceso tendrá un menor tiempo de procesamiento lo que ayuda en la fluidez del sistema.

Así, se encuentran dos clasificaciones para la reducción de dimensionalidad que definen en como trabajará el algoritmo las características del conjunto de datos:

- **Extracción de características:** Este tipo se caracteriza por trabajar bien en datasets pequeños y conocidos, donde podemos saber que características estamos trabajando y por ende saber cual

no vamos a considerar por ser irrelevante, tener menos importancia que las demás, velocidad del proceso, etc.

- **Transformación de características:** Por otra parte, este método de tratado de características funciona mejor para conjuntos de datos grandes y complejos, combinando así diversas características para generar nuevas pero que contengan la información relevante de las anteriores, haciéndolas más significativas y compactas con respecto a lo que se tenía previamente.

Para el caso del presente trabajo de título, estará centrado en los métodos que transformen las características más que los que extraigan algunas, ya que se sabe de antemano que el conjunto de datos con el que se trabajará no será menor y además este será complejo, por lo que la transformación de características calza bien en la descripción del problema actual. En consecuencia, se puede apreciar subdivisiones en esta clasificación que dependerá netamente de la naturaleza de los datos:

3.1.2.1. Lineales

La reducción de dimensionalidad lineal es una técnica que busca proyectar los datos de alta dimensión en un espacio de menor dimensión, donde cada punto en el nuevo espacio es una combinación lineal de las características originales. Esto se logra a través de métodos como Principal Component Analysis (PCA) (4), donde se busca encontrar una transformación lineal que preserve la mayor cantidad posible de información en los datos originales. Aunque estos métodos son rápidos y fáciles de implementar, pueden no ser capaces de capturar la complejidad de la estructura subyacente o "patrones no evidentes" de los datos.

3.1.2.2. No lineales

A diferencia de la reducción de dimensionalidad lineal, la reducción de dimensionalidad no lineal utiliza transformaciones no lineales para crear una representación más rica y compleja de los datos. Esta representación puede ser utilizada para visualizar los datos en un espacio de menor dimensión, así como para mejorar el rendimiento de algoritmos de aprendizaje automático que pueden tener dificultades para trabajar con datos de alta dimensión.

Aunque los métodos de reducción de dimensionalidad no lineal son capaces de capturar relaciones más complejas entre los atributos, aunque su implementación puede ser más compleja y costosa que los métodos de reducción de dimensionalidad lineal.

3.1.3. Clustering

El clustering es una técnica de aprendizaje no supervisado que se utiliza para descubrir patrones y estructuras en conjuntos de datos no etiquetados. Su objetivo es agrupar objetos similares en base a sus características.

De esta forma, nos encontramos con varios métodos de clustering según como se creen pero en el presente proyecto de título solo se mencionarán los tres más relevantes para el caso:

- **Por distancia:** A través de la distancia euclidiana de los datos se buscan los que se encuentran más cercanos entre ellos para asignarles un mismo cluster.

- Por densidad: Este tipo de algoritmo permite clasificar de manera muy efectiva los *outliers* (5) que se puedan presentar en el dataset, ya que al trabajar con densidades y estos datos estar relativamente alejados de los demás, le es más fácil y es más efectivo a la hora de detectarlos. Sin embargo, el hecho de trabajar con densidades, si los datos se encuentran repartidos de alguna manera uniforme a lo largo del espacio este tipo de algoritmo se le puede complicar la clasificación (6).
- Aglomerativos: Los algoritmos aglomerativos van generando varios clusters que de a poco se van uniendo/combinando para generar otros clusters. Esto permite tener diversos pasos de un algoritmo que permitirá visualizar distintas etapas con distinta cantidad de clusters.

3.2. Análisis de trabajos desarrollados por otros autores

3.2.1. ASPECT: A spectra clustering tool for exploration of large spectral surveys

En 2012 H.Meusinger, P.F. Schalldach y M. Newholm explican como se desarrolla una herramienta para realizar clustering de espectros a un estudio de grandes estudios espectrales.

Al igual que en el proyecto en que se basa el siguiente proyecto de memoria, este estudio también obtiene los datos a través de Sloan Digital Key Survey (SDSS), ya que contiene una gran cantidad de datos astronómicos sobre estrellas, galaxias, entre otros. Así, se comenta como con el siguiente software desarrollo se organizan grandes cantidades de datos espectrales a través de un mapa topológico y la similitud que tuvieran los datos en este. Este documento se basa principalmente en el trabajo con mapas SOM (self-organising maps) para su organización y evaluación.

De esta forma, al trabajar con SOM, que funciona con similitud/disimilitud con respecto a la distancia euclidiana, se prueba variando los parámetros de la red con tal de obtener resultados pertinente para la función de aprendizaje $\alpha(t)$ y la función radio $\theta(t)$ como se puede ver en la figura 3.2.

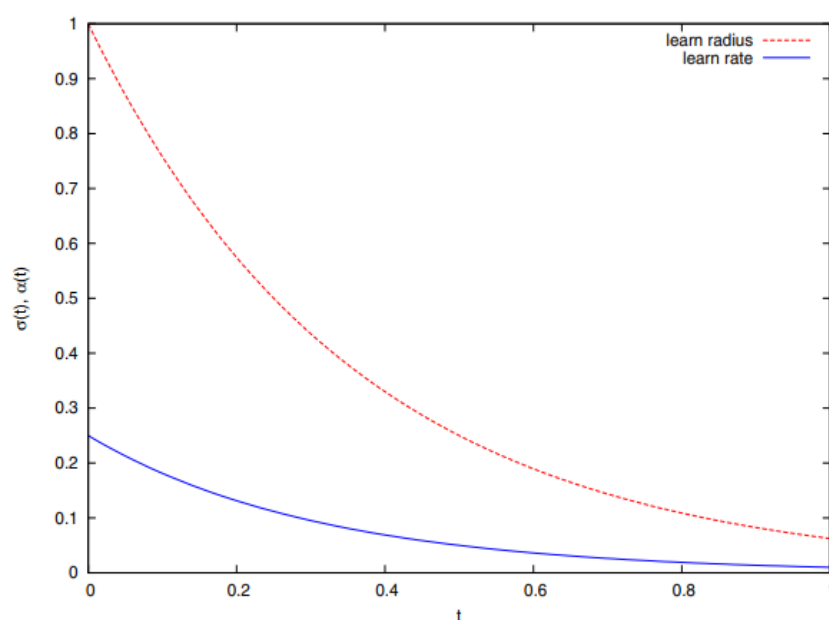


Figura 3.2: Función de aprendizaje del radio (t) y función de tasa de aprendizaje (t).

Este documento explica temas relacionados a los que se tratan en este documento, como el corrimiento al rojo, la relación señal-ruido, lo importante de la representación visual, y además de

tratar con datos de entradas de la misma naturaleza, lo que permite entender de mejor manera la composición de los datos que se tratarán más adelante.

3.2.2. Spectral classification of quasars in the sloan digital sky survey: eigenspectra, redshift, and luminosity effects

Un trabajo realizado por C. W. Yip, A. J. Connolly y D. E. Vanden Berk. que igualmente utiliza el dataset de SDSS, se basa en la clasificación de espectros, y lo que destacable de este documento es el tipo de reducción de dimensionalidad que utilizan para poder trabajar con dimensiones bajas para un dataset que los datos contienen varias características. Se observa que se trabaja con la transformada Karhunen-Loève, que es un método lineal bastante similar al de Principal Component Analysis (PCA), solo que para grupos de datos continuos se comportan diferentes, pero para casos discretos se tratan de igual manera. En este caso se utiliza para *Gap Correction*, que hace referencia a un ajuste que se realiza para corregir errores o discrepancias en los datos observacionales debido a la falta de información en ciertas regiones del espacio.

En este documento se pueden obtener características similares al mencionado en la sección 3.2.1, donde se hablan tópicos como el corrimiento al rojo, luminosidad, entre otras características de las estrellas (y por ende de las enanas blancas), y se hace mención en que la clasificación se hace en base a la luminosidad de las estrellas, lo que abre la idea de que en este proyecto al ser un trabajo con aprendizaje no supervisado se tome en analice que características de las estrellas se están considerando para observar el porque el algoritmo las detecta como parte del mismo grupo.

4 | Alternativas de solución del trabajo

En esta sección se presentan las alternativas para el proyecto, donde se divide en tres secciones de alternativas ya que se cuenta de tres etapas donde se pueden encontrar diversas soluciones para: reducción de dimensionalidad; clustering; y resultados gráficos.

4.1. Etapa de reducción de dimensionalidad

4.1.1. Principal Component Analysis (PCA)

La primera alternativa corresponde a PCA, técnica que permite analizar grandes cantidades de datos con altas dimensiones con la finalidad de mantener la mayor cantidad de información a través de la reducción de dimensiones de esta. Se utiliza principalmente para análisis exploratorio de datos y para modelos predictivos, además de su uso en la misma reducción de dimensiones.

La idea de PCA es que a través de la varianza de las variables iniciales y las matrices de covarianza que estas generen a tratar con las variables normalizadas, se obtengan los componentes principales que corresponden a combinaciones lineales de las variables iniciales, donde las variables más relevantes tendrán un mayor peso que las de menor importancia.

Así, PCA faculta a través de un análisis lineal, trabajando con una gran cantidad de dimensiones, obtener una solución con solo características importantes al caso a través de una velocidad moderada de procesamiento, filtro de ruido en la entrada, y es capaz de entregar una visualización de los datos que tienen grandes dimensiones (7).

4.1.2. T-distributed Stochastic Neighbor Embedding (t-SNE)

A diferencia de PCA, t-SNE es un algoritmo no lineal que de igual manera sirve para reducir altas dimensionalidades en los datos, con la idea de poder observarlos gráficamente en 2 a 3 dimensiones.

Asimismo, t-SNE a diferencia de PCA trabaja con distancias entre puntos más que la varianza máxima entre ellos, por lo que trabaja mejor con datos más unidos y por ende genera una mejor representación visual o más entendible.

Para implementar t-SNE es sencillo gracias a la librería *sklearn*, de donde el procedimiento es casi estándar para sus aplicaciones, donde podemos encontrarnos con distintos parámetros a configurar (*n_components*, *perplexity*, *learning_rate*, entre otros), pero generalmente se mantienen

en sus valores por defecto a excepción de la cantidad de componentes, que nos permite elegir la cantidad de dimensiones que queremos para el conjunto que se transforme (8).

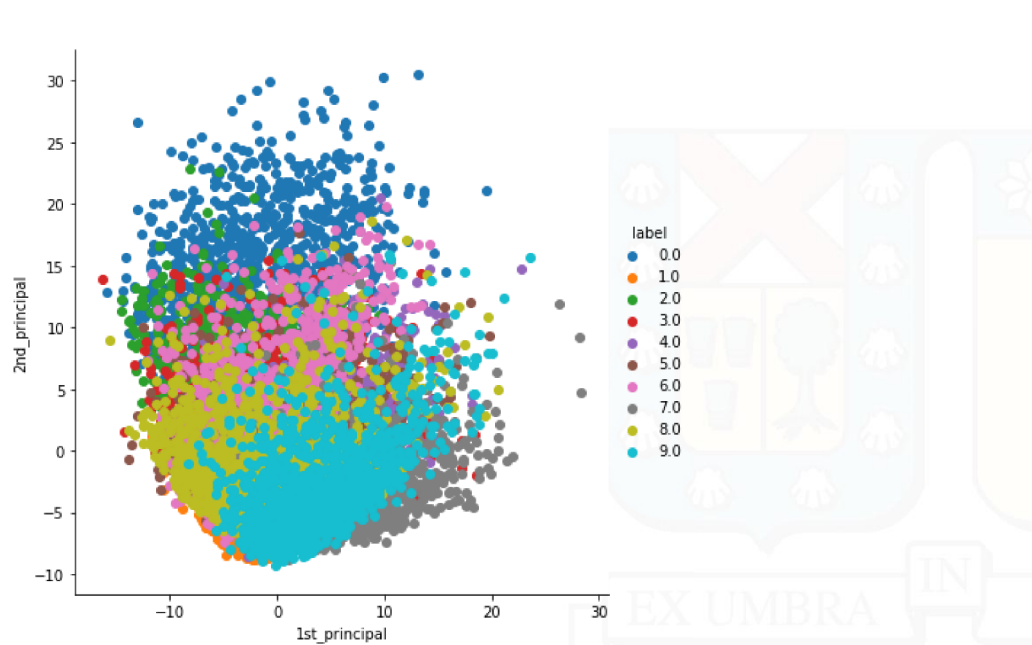


Figura 4.1: Aplicando PCA a MNIST.

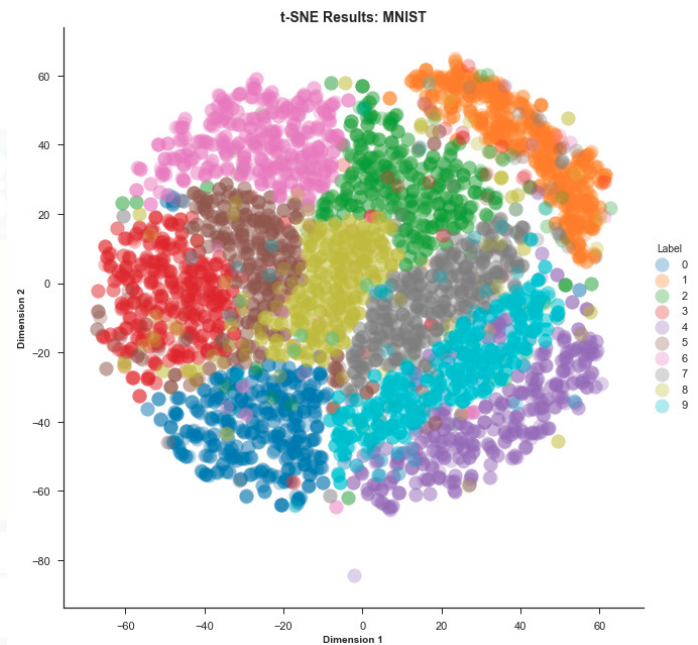


Figura 4.2: Aplicando t-SNE a MNIST.

4.2. Etapa de clustering

4.2.1. K-Means

Este algoritmo tiene como principal objetivo asignarle a los puntos un cluster a través de la mínima distancia hacia un centroide generado aleatoriamente en un inicio.

Para este algoritmo se debe elegir de manera manual la cantidad de clusters que se desean tener, por lo que para el caso es necesario saber de antemano la cantidad de grupos que queremos obtener en la salida.

Este método de clustering es uno de los más utilizados por su fácil implementación, entendimiento de la salida, y velocidad con que se ejecuta (9).

En la figura 4.4 se puede ver cómo es el procedimiento del algoritmo para determinar a que cluster corresponde cada dato a través de varias iteraciones.

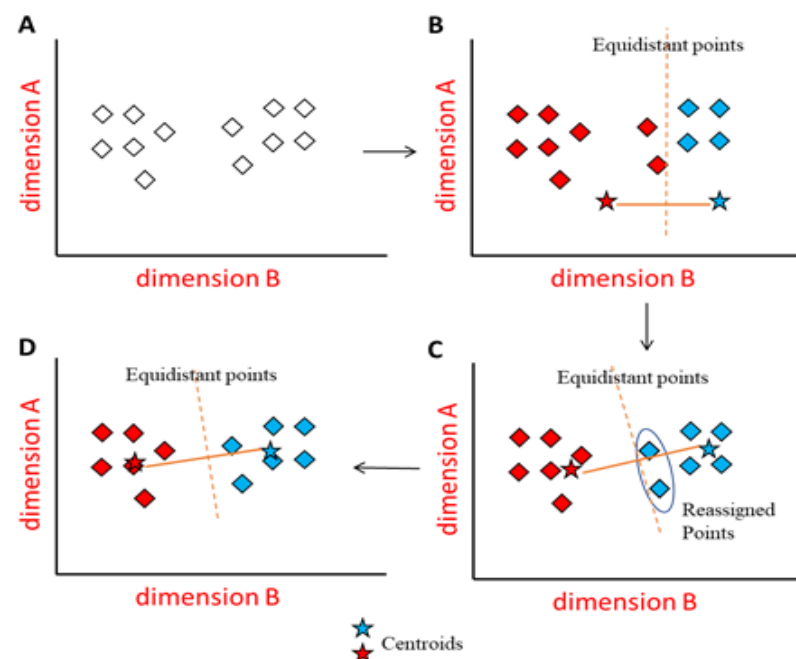


Figura 4.3: Procedimiento del algoritmo K-Means.

4.2.2. DBSCAN

DBSCAN a diferencia de K-Means, es un algoritmo que se basa en las densidades en un área, lo que permite lidiar de una mejor manera con los outliers y definir los clusters cuando hay una aglomeración demasiado grande en cierta parte del espectro.

Para iniciar su el algoritmo inicia por un punto con una densidad suficiente y crea un cluster para luego tomar los puntos cercanos a este, asignarlos a este grupo hasta no encontrar más puntos en este punto de densidad. Luego de terminar de tomar los puntos relevantes, se busca otro punto con densidad suficiente y se comienza otro cluster aparte. Así, los puntos que estén relativamente alejados o no tan cercanos a puntos altos de densidad se considerarán *outliers*.

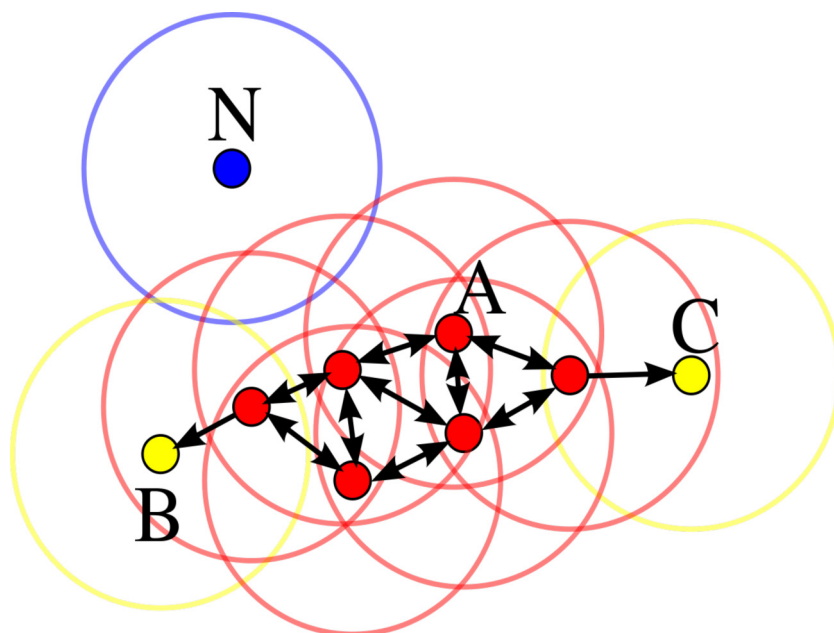


Figura 4.4: Procedimiento del algoritmo DBSCAN.

4.2.3. Mean-Shift

El siguiente método es un método basado en centroides al igual que K-Means, que al igual que dicho método, tiene una variable que controla la cantidad de clusters, en este caso a través de una variable llamada *bandwidth*(10) y con esto el algoritmo decide a través de un proceso cuantos clusters habrá finalmente.

El proceso consiste en que básicamente se mueven los puntos a los lugares con densidad más cercanos a ellos hasta el punto en que no se puedan mover más, esto a través de un proceso bastante utilizado en optimización llamado *hill climbing*, y finalmente, cuando estos puntos no sean capaces de moverse más, se considerarán parte de un mismo cluster.

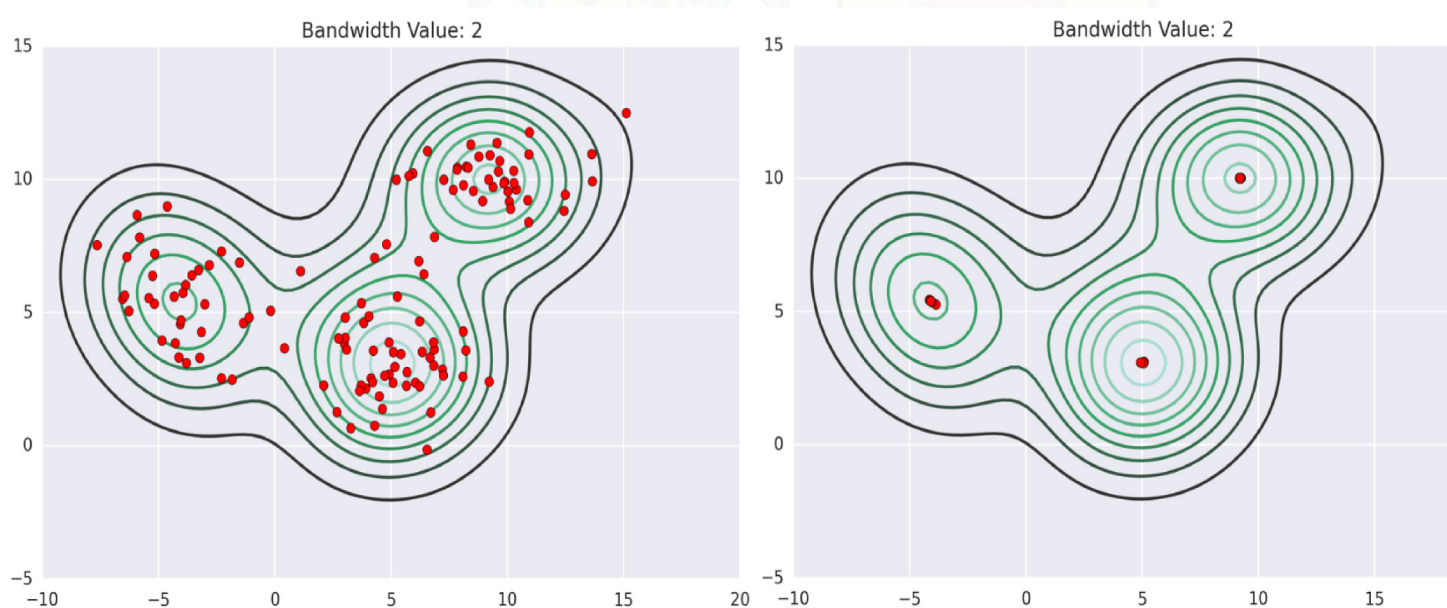


Figura 4.5: Procedimiento del algoritmo Mean Shift.

4.2.4. Agglomerative Nest (AGNES)

Finalmente, AGNES corresponde a un modelo que va creando distintos rangos crecientes de clusters, donde se inicia decidiendo que cada dato es un cluster distinto y a partir de ahí se van agrupando hasta tener a todos los datos en un solo cluster.

El proceso de este algoritmo es que por cada etapa de este proceso, se buscan los 2 clusters más cercanos entre todos los datos y se agrupan creando un cluster más grande. Así, progresivamente hasta llegar a agruparlos todos en un mismo cluster como se mencionó anteriormente.

Sin embargo, este método puede tomar tiempo si es que se trabaja con datasets muy grandes ya que al realizar el algoritmo en varios pasos/iteraciones, este puede tardar más de lo deseado (11).

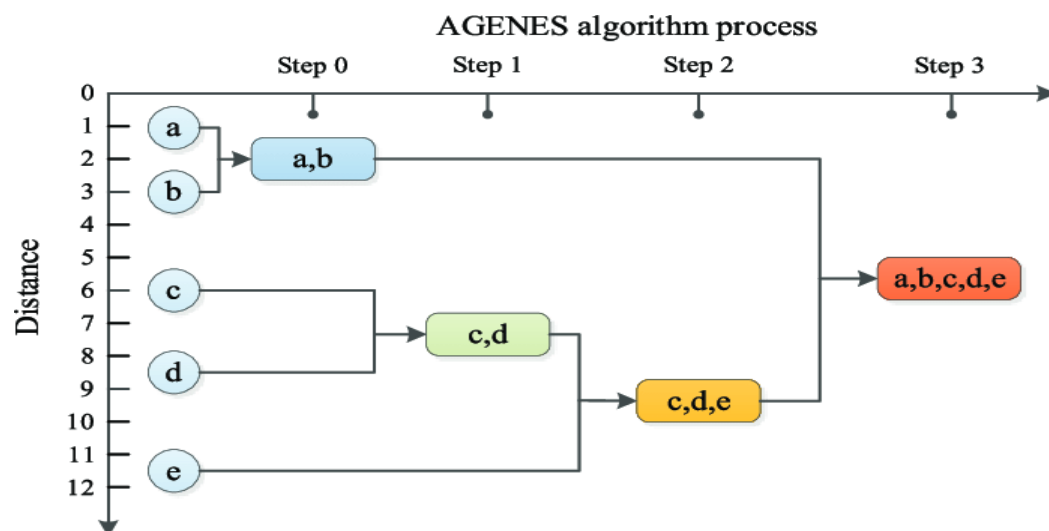


Figura 4.6: Procedimiento del algoritmo AGNES.

4.2.5. Spectral clustering

Spectral Clustering es un algoritmo de agrupamiento que se diferencia de los demás al aprovechar las propiedades espectrales de la matriz de afinidad de los datos. En lugar de depender únicamente de la distancia euclidiana entre los puntos, Spectral Clustering busca agrupaciones más complejas y no lineales al analizar la estructura de dicha matriz. Este enfoque implica construir una matriz de afinidad que refleje las relaciones entre los puntos, calcular los vectores propios y reducir la dimensionalidad del espacio de características. Posteriormente, se aplica un algoritmo de agrupamiento, como K-Means, para asignar los puntos a los diferentes clusters en función de la proximidad espacial y las similitudes estructurales capturadas en la matriz de afinidad(12).

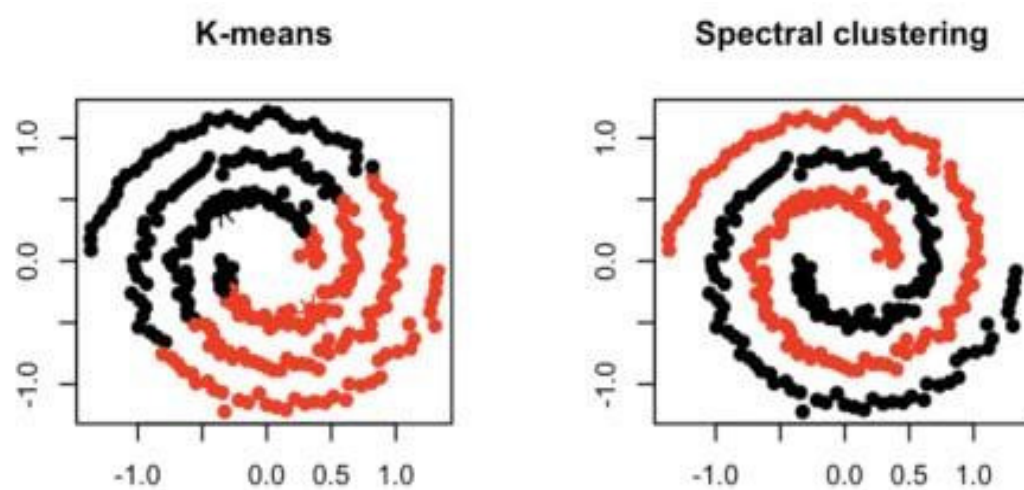


Figura 4.7: Comparación entre K-Means y Spectral clustering en espectros complejos.

4.3. Visualización de la salida

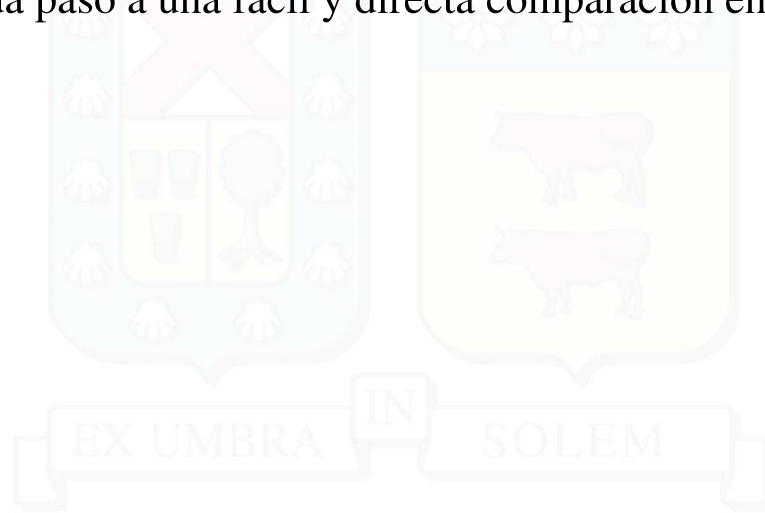
4.3.1. Gráfico de dispersión en 2 dimensiones

La primera opción corresponde al gráfico en 2 dimensiones que entrega una representación visual de los datos en un plano cartesiano el cual representa la relación entre 2 variables y como se comportan los datos respecto a estas. El gráfico de dispersión permite observar los datos repartidos

en el espacio y concluir respecto a su ubicación espacial, apreciando los límites de los clusters, los *outliers*, donde se concentran más datos, entre otros análisis relevantes para el caso.

4.3.2. Gráfico de barras

La segunda opción corresponde al gráfico de barras, el cuál permite saber con exactitud el número de datos por cluster y da paso a una fácil y directa comparación entre la cantidad de datos por clusters.



5 | Solución del proyecto

5.1. Relevancia de las soluciones en base a características

Para definir que determinará que alternativa es la más óptima, se consideran tres características claves que determinarán que tan óptima son las diversas soluciones. De esta forma, las características a considerar son la efectividad, el rendimiento computacional y la complejidad, donde:

- La efectividad es claramente la característica más relevante para el proyecto, siendo crucial que el pipeline a desarrollar entregue resultados fidedignos, y no resultados poco fiables.
- El rendimiento computacional también tomará un papel importante a la hora de determinar la mejor solución, ya que es parte directa de los objetivos del proyecto (que sea un proceso rápido).
- La complejidad pasa a un menor plano de importancia al solo determinar que tan difícil será implementar la solución, lo que no afectará directamente en la funcionalidad del sistema.

De esta forma, comparando las soluciones en base a estas características y considerando sus ventajas y desventajas se tiene lo siguiente:

5.1.1. Reducción de dimensionalidad

En cuanto a PCA, este es de sencilla implementación y entrega resultados a una alta velocidad incluso tratando con cantidades de datos grandes, por lo que es bastante efectivo para esta tarea. Por otro lado, t-SNE tiene una gran ventaja sobre PCA, que corresponde a la representación gráfica que otorga, que es mucho más interpretable que la entregada por PCA. Sin embargo, t-SNE no preserva distancias absolutas ni las proporciones de densidad del espacio de alta dimensión original, por lo que el resultado de este proceso es meramente para una fácil interpretación de los datos.

Esta característica de t-SNE impide que se pueda realizar un clustering posterior ya que este realizaría un agrupamiento de datos meramente por su relación espacial y no por su relación en cuanto a características, lo cuál no es lo que se busca.

5.1.2. Clustering

En esta etapa los diversos métodos se pueden ver afectados por la complejidad temporal que poseen, y por ende, afectando el tiempo de procesamiento de la etapa y del sistema integral. Esta es la razón del por qué se descarta el uso de AGNES y Mean Shift, algoritmos que poseen una

complejidad de $O(n^2 \log n)$ y dependiente del kernel respectivamente (alto costo computacional)(13), dejándolos en clara desventaja con las otras soluciones.

De esta manera, es necesario destacar que tanto K-Means como Spectral Clustering son métodos donde es necesario establecer el número de clusters de manera manual, mientras que por su lado DBSCAN encuentra en base a sus parámetros la cantidad de clusters encontrados de manera automática. Estas tres soluciones permiten trabajar desde distintas perspectivas el mismo problema cumpliendo con los objetivos propuestos en 2.1, además de no imponer una mayor complejidad en su implementación más allá de selección de una correcta selección de parámetros.

5.1.3. Representación gráfica.

Para esta etapa, es necesario recalcar que no se comparan con las características previas, esto debido a que no es estimable cuan efectiva es un gráfico, o la complejidad de su implementación o la carga que impone al sistema. Por lo que se evalúa la utilidad que este otorga en base a la información que entrega al usuario final.

Así, se determina que tanto el gráfico de barras como el gráfico de dispersión entregan información que se complementan entre sí, entregando amplia información al usuario de tal manera que le permita realizar un análisis más profundo a los datos con ellos.

5.2. Solución a implementar

Según lo mencionado en la sección 5.1:

1. Para la etapa de reducción de dimensionalidad PCA permite un correcto trabajo con grandes cantidades de datos, una implementación sencilla y resultados confiables. Sin embargo, la visualización de la salida de PCA no resulta fácilmente interpretable por el usuario en algunos casos, por lo que para solucionar este problema se decide implementar una etapa de t-SNE después de la etapa de clustering, solo con el fin de permitir al usuario observar de mejor manera los datos y clusters generados.
2. Para la etapa de clustering se procede a trabajar con K-Means, DBSCAN y Spectral Clustering en paralelo, donde estos algoritmos procesarán la misma salida entregada por PCA, para entregar tres salidas realizadas por distintos tipos de algoritmos válidos para este proyecto.

Es necesario mencionar que esto se realiza ya que para la finalidad del proyecto no buscamos la mejor solución, si no las soluciones válidas que ayuden al problema. Por lo que implementar estos tres algoritmos entregarán basta información al usuario que se traduce en aún más análisis posible a realizar.

3. Finalmente, en cuanto a la representación gráfica también es válido lo mencionado previamente, donde no se busca elegir una solución o alternativa, más bien las que nos entreguen la mayor cantidad de información relevante, por lo que se considera que implementar tanto el gráfico de barras como el gráfico de dispersión son necesarios.

Asimismo, se considera necesario implementar una matriz de confusión para cada uno de estos algoritmos, esto debido a que se está trabajando con diversas clases y por ende esta matriz puede permitir al usuario encontrar que tipo de enana blanca se está mezclando con otra clase,

o se esta repartiendo entre varias o se considera ruido, entre otras conclusiones útiles que se pueden obtener a partir de esto.



6 | Desarrollo de la solución

6.1. Ambiente utilizado para el desarrollo de la solución

Las diferentes soluciones que serán explicadas a continuación fueron desarrolladas utilizando el lenguaje Python v. 3.10.4 utilizando las siguientes librerías para la implementación de las alternativas:

Propósito general	Manipulación de datos	Visualización	Machine Learning
Numpy	Pandas	Pyplot	Sklearn
Time	Pathlib	Seaborn	Keras
Collections			

Tabla 6.1: Librerías utilizadas para la implementación de las soluciones.

Además, es relevante mencionar los componentes del sistema donde se ejecutaron los diversos procedimientos, ya que es una matriz clave para definir el tiempo de procesamiento que tomará cada alternativa. Por lo que, a continuación se adjuntan los componentes del computador en el cual se trabajó:

- Procesador (CPU): Intel Core i5-9400f @ 2.90GHz.
- Memoria RAM: 16 GB (2x8GB) DDR4 @ 2133GHz.
- Disco duro o SSD: SSD 512 GB NVMe.
- GPU: NVIDIA GeForce RTX 3060.

En cuanto a los datos utilizados para probar funcionamiento de las alternativas, estos fueron proporcionados por el profesor Mauricio Araya y corresponden a datos *crudos*, procesados, validados, entrenamiento y testeo. De esta forma, las pruebas que se mostrarán más adelante fueron realizadas utilizando los datos de testeo que consisten en una cantidad de 1124 datos. Además, entre estos datos también nos encontramos con el modelo entrenado con el cuál se realizarán las pruebas necesarias.

6.2. Métricas a evaluar durante el desarrollo

Para ver el desarrollo correcto y deseado de las diversas etapas, se debe corroborar que se cumplan ciertas métricas a lo largo del sistema, para que el avance de este vaya de la mano con los objetivos finales del proyecto. De esta forma, se definen las siguientes métricas a tener en consideración:

- **Tiempo de procesamiento:** Medir el tiempo de procesamiento es crítico para saber cuanto demora cada etapa y el conjunto de ellas, lo que se utilizará para determinar si la alternativa cumple con tiempos deseados relativos a los objetivos planteados en un inicio.
- **Varianza explicada:** Métrica que permite evaluar cuanta información de los datos utilizados es captada a través de los componentes principales en el proceso de reducción de dimensionalidad (PCA). Un valor alto de varianza explicada indica que el componente principal retiene una cantidad significativa de información de los datos originales, mientras que un valor bajo indica que el componente principal tiene menos influencia en la variabilidad de los datos, así se establece que en base a lo solicitado por el profesor Mauricio Araya se requiere de un mínimo de un 90 % en los componentes que se considerarán.

Estas métricas podrán verse reflejadas en los respectivos gráficos de la sección 6.3 bajo el título de cada uno de ellos.

6.3. Desarrollo de las etapas planteadas

Para el desarrollo de las etapas de cada solución, se debe dejar en claro las soluciones que se prueban para comparar funcionamiento, rendimiento y cumplimiento de métricas/objetivos. Así, los enfoques que se tomarán serán:

Etapas	Reducción de dimensionalidad	Clustering
1	PCA + t-SNE	K-Means
2	PCA + t-SNE	DBSCAN
3	PCA + t-SNE	DBSCAN + Spectral Clustering

Tabla 6.2: Alternativas desarrolladas.

Se puede observar que todas las soluciones utilizan PCA y t-SNE para la reducción de dimensionalidad, sin embargo, cabe destacar que estos se utilizan con distintas finalidades como se mencionó en la sección 5.1. PCA por su parte lo utilizamos para reducir las dimensiones de los datos latentes del inicio del pipeline, mientras que t-SNE se utilizará al final del pipeline como método para visualizar de mejor manera los clusters encontrados por los distintos algoritmos de clustering.

Añadido a esto, resulta relevante mencionar que tanto en las etapas de extracción de datos como en el desarrollo de PCA + t-SNE, las tres soluciones siguen el mismo procedimiento. Sin embargo, las diferencias entre estas alternativas se presentan únicamente en la etapa de Clustering. Por lo tanto, las secciones 6.3.1 y 6.3.2 son compartidas por todas las soluciones en cuanto a su desarrollo, y la representación gráfica final se mostrará una comparativa entre todas las soluciones.

6.3.1. Extracción de datos del espacio latente

Hasta este momento solo se había mencionado una entrada de datos como se puede observar en la figura 1.2 pero no se ha ahondado más en esa etapa ya que no corresponde a un proceso que le asignemos alternativas para obtener una *mejor* solución, más bien una acorde y que concuerde con la lógica de los códigos y algoritmos con los que trabajará.

Dicho esto, la idea principal de esta etapa es poder extraer correctamente datos del espacio latente de la red neuronal del proyecto principal, con la idea de trabajar con los datos del espacio de interés y analizar posteriormente el procesamiento de estos datos.

Así, este proceso se encarga de transformar la representación interna de la red neuronal en una forma más interpretable y adecuada para su análisis posterior donde se pueden destacar las siguientes características:

- Si bien algunas capas pueden diferir en las dimensiones de sus matrices (ver figura 6.1) esto se soluciona y explica en la sección 6.3.2
- Al inicio de esta etapa se comienza a medir el tiempo para determinar el tiempo total de procesamiento de cada alternativa desarrollada.

Dicho esto, en la etapa de extracción de datos, se carga el modelo previamente entrenado. En esta fase, se muestra un resumen del modelo, incluyendo las diferentes capas que lo componen, lo que permite al usuario del pipeline seleccionar la capa específica del espacio latente de la cual desea extraer los datos. Así, aunque se carga el modelo proporcionado por el profesor, esta etapa brinda la flexibilidad de cambiar el modelo por otro más eficiente o conveniente en caso de ser necesario durante su uso.

En esta etapa lo fundamental es la elección de la capa a la cual se le desea realizar la extracción de datos, donde se permite seleccionar cualquiera independiente de sus dimensiones. Además de seleccionar la capa para la extracción de datos, en esta etapa se define el conjunto de datos que se utilizará para el análisis posterior del sistema. Aunque actualmente se utilizan datos de prueba mencionados previamente, es posible intercambiarlos con otros conjuntos de datos, siempre y cuando cumplan con la extensión .npy (numpy array) para asegurar el correcto funcionamiento del pipeline.

Para mostrar el funcionamiento del pipeline se realizan pruebas en 3 distintas capas de ejemplo, siendo estas *dense_15*, *dense_14* y *dense_13*, esto debido a que estas capas son las que capturan las características de nivel superior de los datos y nos proporcionan una representación mucho más compacta de los datos al tener una menor cantidad de dimensiones.

Layer (type)	Output Shape	Param #
conv1d_12 (Conv1D)	(None, 2099, 128)	640
max_pooling1d_12 (MaxPoolin g1D)	(None, 1049, 128)	0
conv1d_13 (Conv1D)	(None, 523, 64)	32832
max_pooling1d_13 (MaxPoolin g1D)	(None, 261, 64)	0
conv1d_14 (Conv1D)	(None, 129, 32)	8224
max_pooling1d_14 (MaxPoolin g1D)	(None, 64, 32)	0
conv1d_15 (Conv1D)	(None, 31, 16)	2064
max_pooling1d_15 (MaxPoolin g1D)	(None, 15, 16)	0
flatten_3 (Flatten)	(None, 240)	0
dense_13 (Dense)	(None, 64)	15424
dense_14 (Dense)	(None, 32)	2080
dense_15 (Dense)	(None, 12)	396

Figura 6.1: Capas del modelo pre-entrenado con el cual se realizan las pruebas.

6.3.2. Reducción de dimensionalidad vía PCA

En esta etapa se realizan 2 procesos clave que le dan forma a las soluciones. El primero corresponde al manejo de datos, donde haremos referencia a:

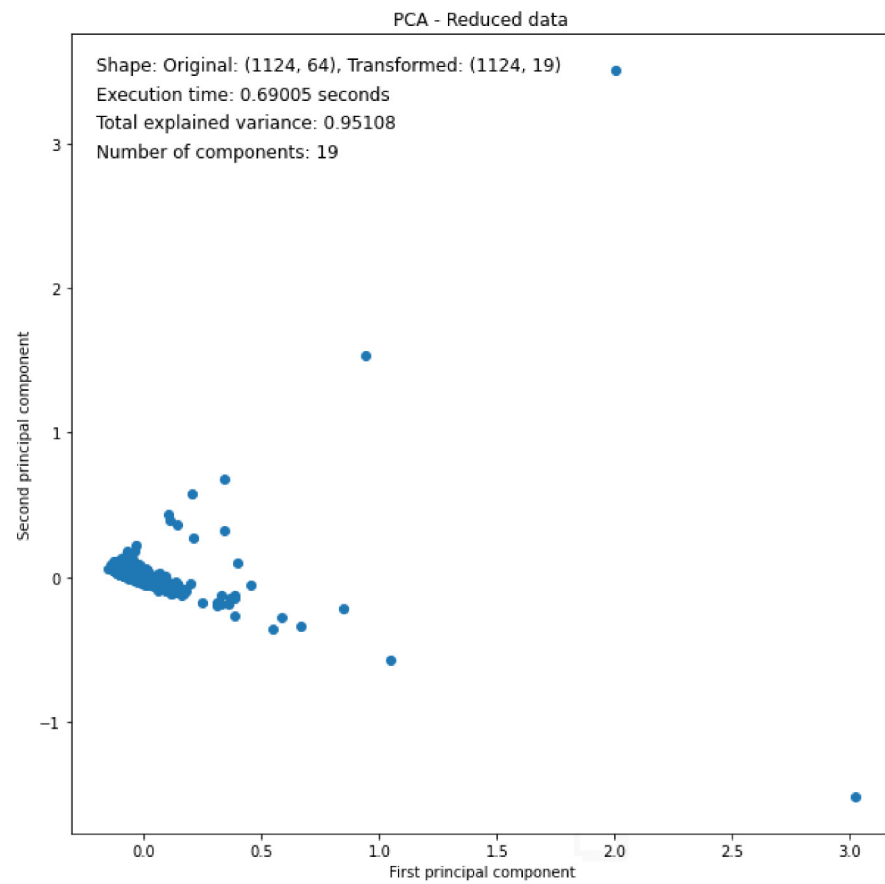
- Elección de número de componentes: En esta sección se permite la elección de número de componentes (número entero mayor a 1) o bien, también puede ser directamente la varianza explicada que se desea obtener (número decimal entre 0 y 1). De este modo, se decide escoger arbitrariamente para los casos que se mostrarán a continuación una varianza explicada mínima del 95 % para cumplir con diferencia la métrica previamente impuesta y obtener resultados óptimos.
- Escalamiento de datos: El escalamiento de datos es un paso fundamental antes de aplicar PCA debido a que se basa en la varianza y covarianza de las características para determinar los componentes principales. Así, si las características tienen diferentes escalas, algunas pueden dominar la varianza total simplemente debido a sus magnitudes absolutas lo que puede desembocar en resultados erróneos o no deseados.
- Aplicar *flatten* a los datos: Para poder aplicar PCA de manera correcta, es necesario utilizar la función *flatten* en los datos de entrada. Esto se debe a que PCA requiere que los datos se encuentren en una estructura bidimensional, y como se puede observar en la figura 6.1, existen capas que tienen salidas con 3 dimensiones, lo que no haría posible aplicar PCA, por ende es necesario aplicar esta función para poder realizar esta etapa en cualquier capa que se desee.

Y lo segundo a tratar es directamente el proceso de reducción de dimensionalidad, donde para cada iteración se medirá su respectivo tiempo de procesamiento, varianza explicada, cantidad de componentes considerados para dicha varianza explicada, y un gráfico con la salida después de realizar PCA.

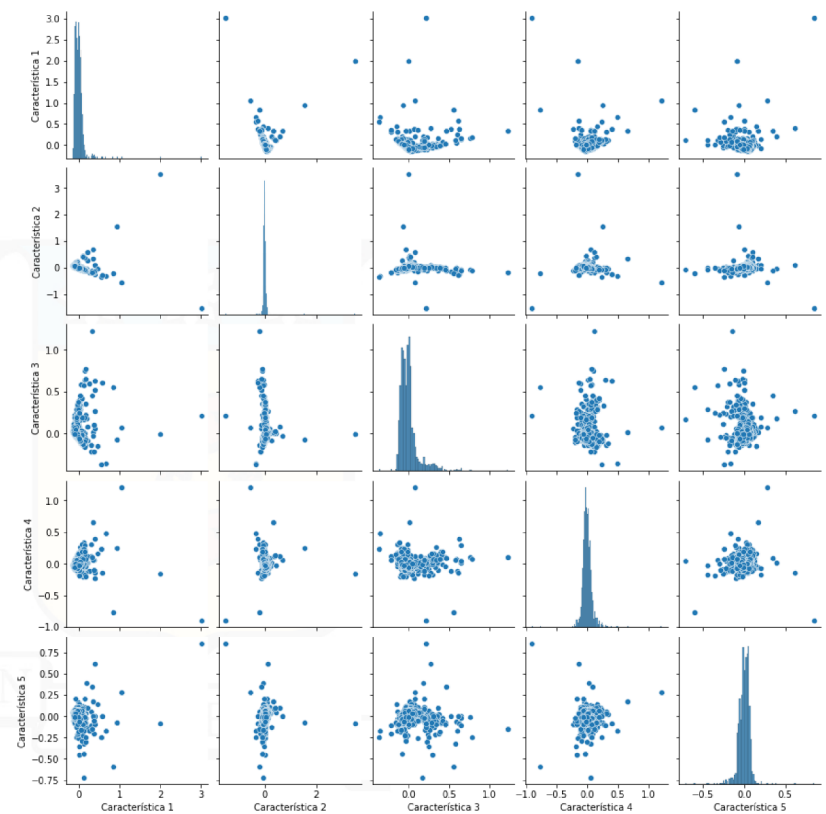
Sin embargo, previo a pasar a mostrar las pruebas se realiza la siguiente observación respecto al funcionamiento de esta etapa:

- Al momento de seleccionar la cantidad de componentes o bien, la varianza explicada mínima deseada, el tiempo de procesamiento varía según como se ingrese este valor. Si se ingresa la cantidad de componentes el tiempo se ve reducido de una gran manera comparado a poner la varianza explicada mínima deseada, esto debido a que el algoritmo debe ir calculando en cada iteración la varianza explicada acumulada lo que ralentiza el proceso. Aún así, a pesar de que el proceso se demore más a través de la elección de una varianza explicada mínima, es la mejor forma de asegurar obtener los resultados deseados y por ende las pruebas considerarán el tiempo de procesamiento con esta elección en consideración.

De esta manera, se pueden obtener los siguientes gráficos y resultados para las distintas capas:

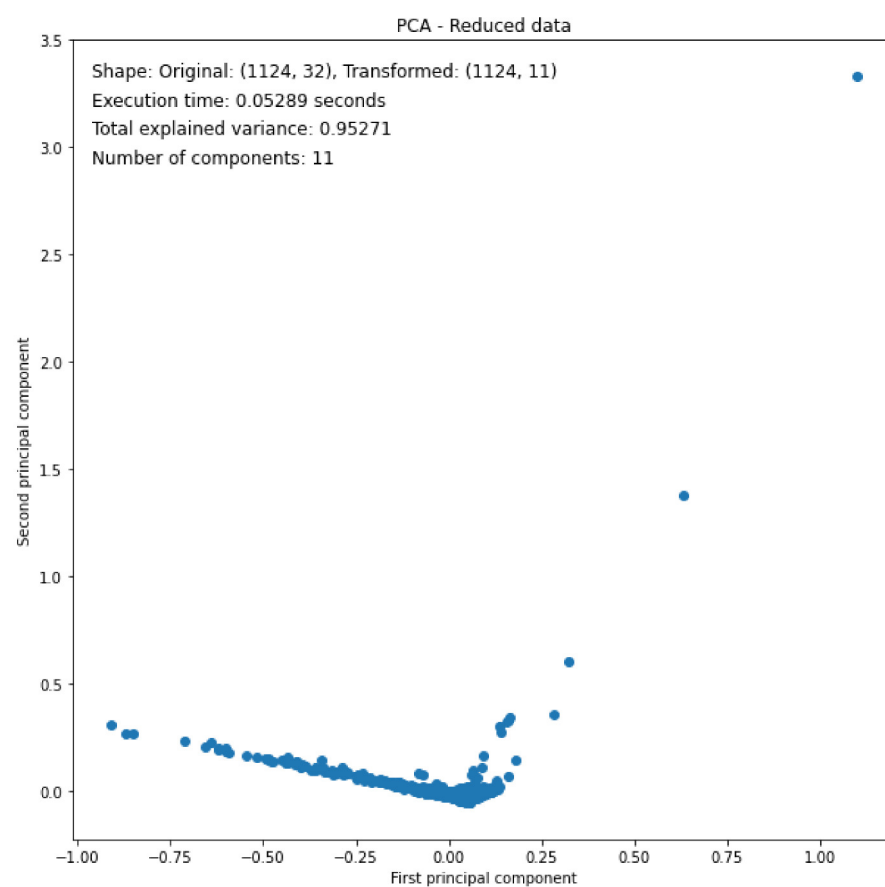


((a)) Gráfico de dispersión para datos latentes luego de utilizar PCA en la capa *dense_13*

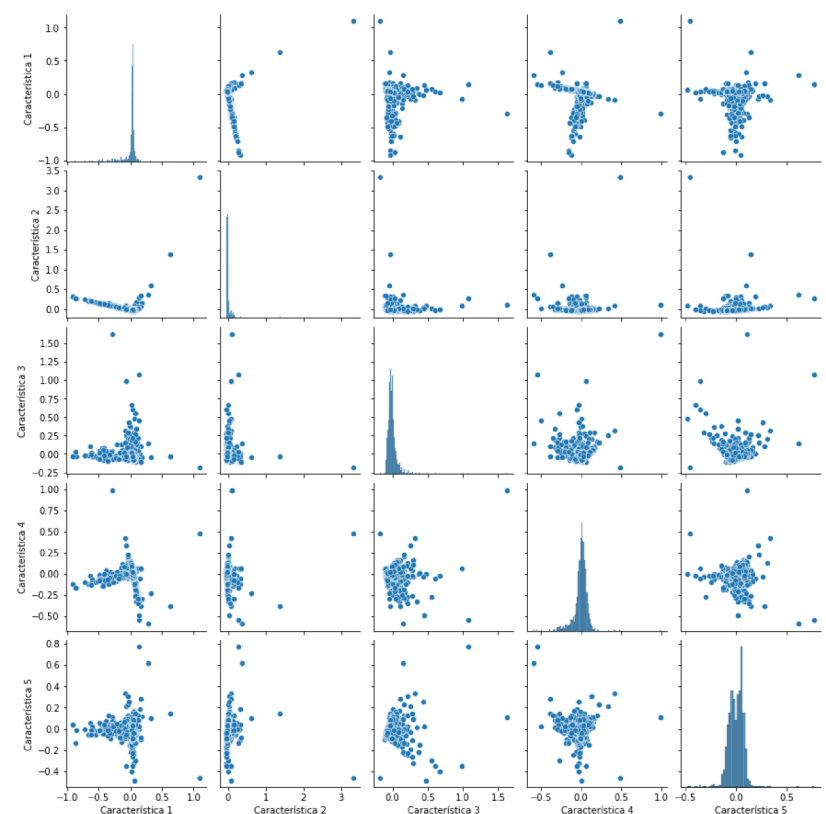


((b)) Gráficos de las 5 dimensiones con mayor varianza explicada para datos de salida de PCA en la capa *dense_13*

Figura 6.2: Gráficos y datos luego de la etapa de PCA en la capa *dense_13*.



((a)) Gráfico de dispersión para datos latentes luego de utilizar PCA en la capa *dense_14*



((b)) Gráficos de las 5 dimensiones con mayor varianza explicada para datos de salida de PCA en la capa *dense_14*

Figura 6.3: Gráficos y datos luego de la etapa de PCA en la capa *dense_14*.

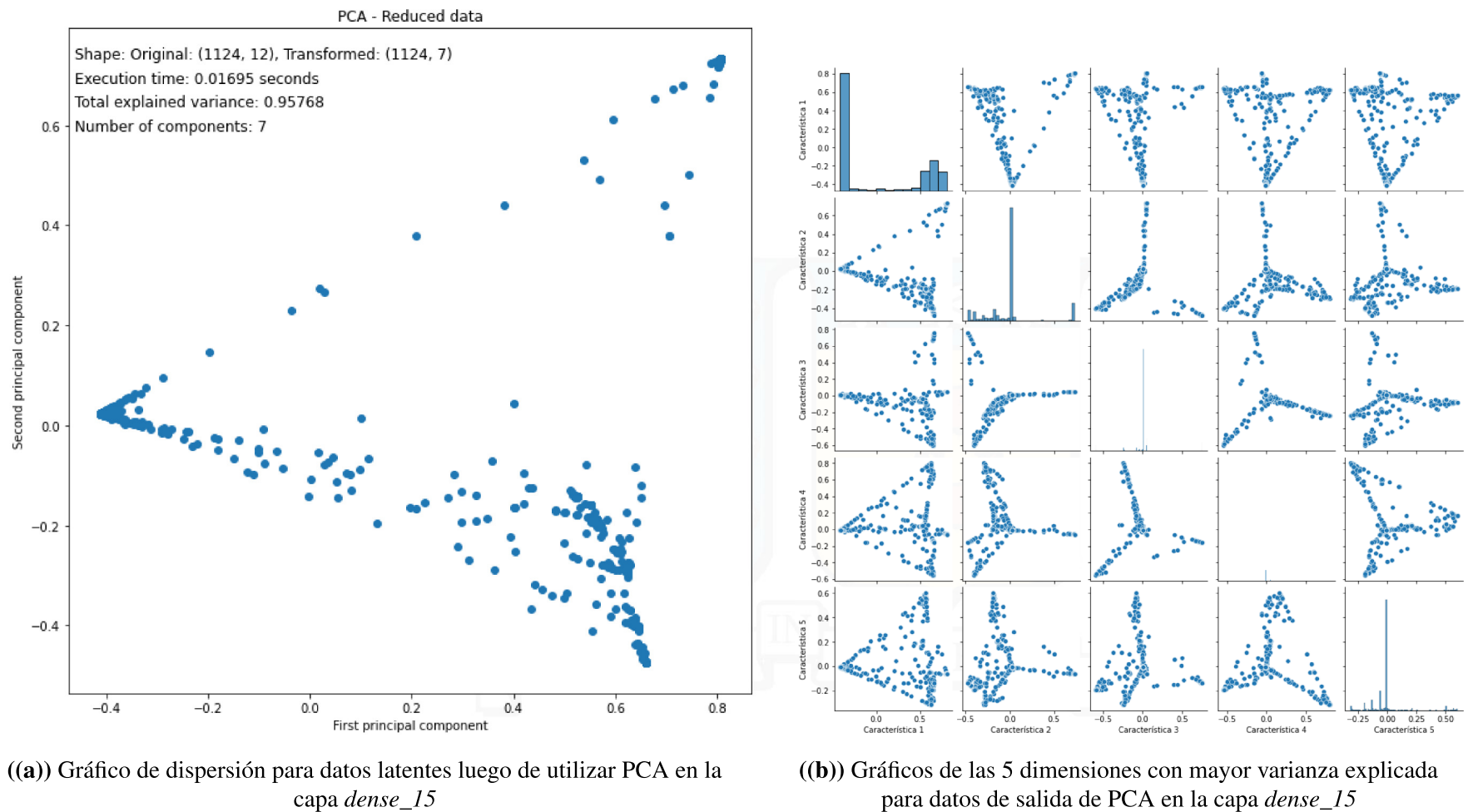


Figura 6.4: Gráficos y datos luego de la etapa de PCA en la capa *dense_15*.

Además de los gráficos de las 2 características más relevantes para cada capa, estos se acompañan de un gráfico de las relaciones que tienen las 5 dimensiones con mayor varianza explicada de cada proceso, con el fin de poder observar de que si existen algunos tipos de patrones, se puedan encontrar ciertos grupos de clusters. Sin embargo, esto ocurre solo en el caso de la capa *dense_15*, y tiene sentido debido a que en esta capa nos encontramos con una menor cantidad de características (se pasa de 12 a 7 como se puede observar en la figura 6.4) mientras que en las otros dos casos se tiene una cantidad mucha mayor de componentes. Y justamente por tener una mayor cantidad de componentes se puede observar que en las distintas dimensiones de las figuras 6.2 y 6.3 se encuentran la gran mayoría de los datos agrupados en un grupo bastante compacto.

6.3.3. Clustering

6.3.3.1. K-Means

Esta etapa en contraste a la previa, solo se preocupa de trabajar con la salida que proporciona PCA, por lo tanto, lo relevante de esta etapa es definir la cantidad de clusters deseados a través del parámetro *n_clusters*. Para esta ocasión, se decide establecer un valor de 12 clusters ya que corresponde al número de clusters que el modelo del que forma parte clasifica en total. Sin embargo, este valor es fácilmente cambiabile por lo que permite flexibilidad en su uso y no consume recursos extras.

De esta forma, los gráficos obtenidos para las distintas capas luego de realizar K-Means son los siguientes:

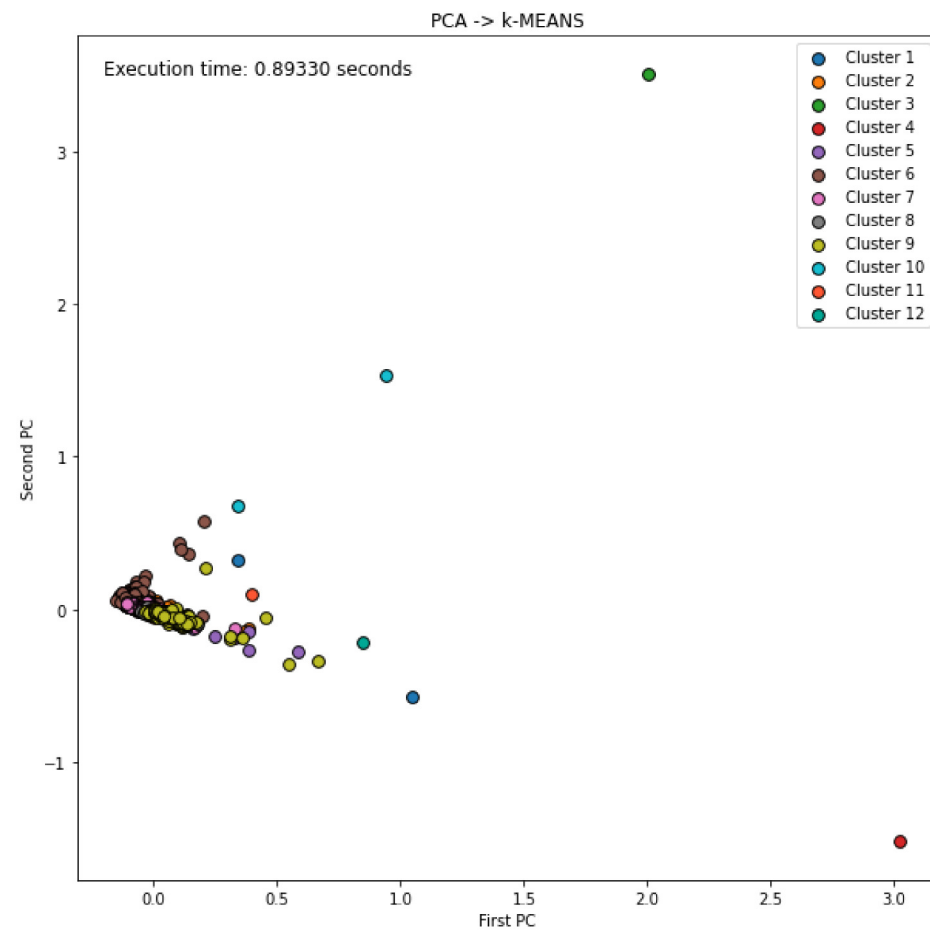


Figura 6.5: K-means para datos de salida de PCA en la capa *dense_13*.

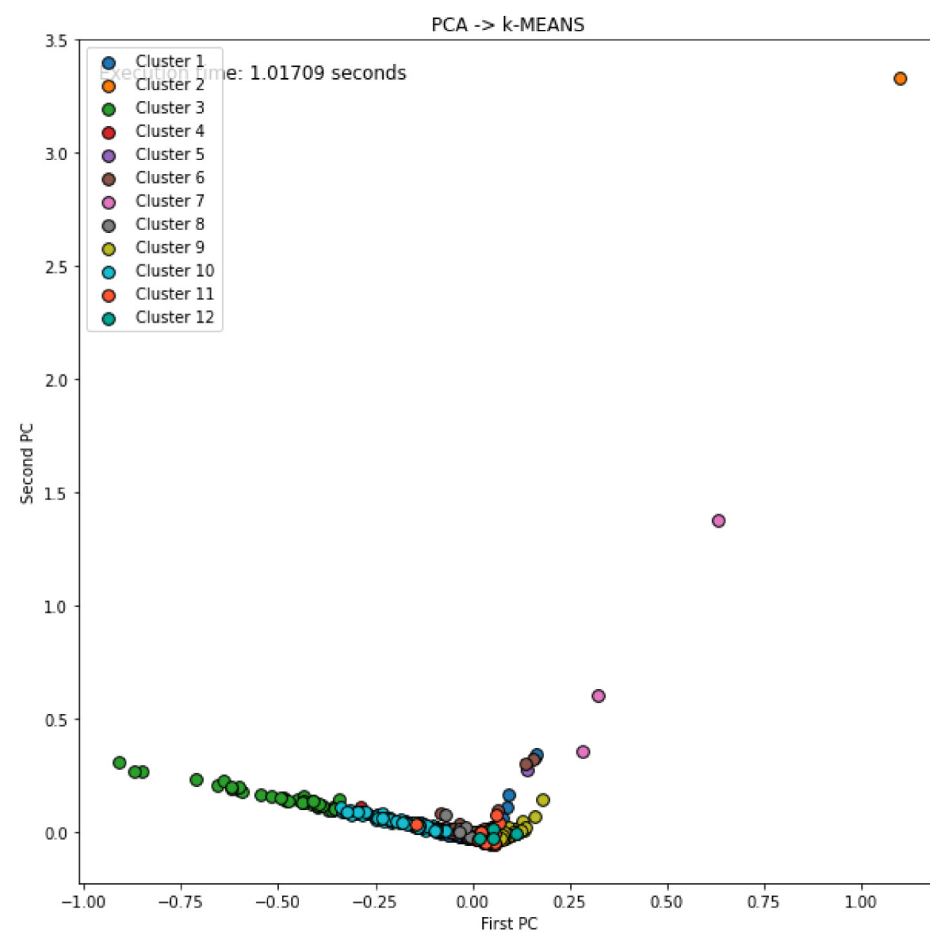


Figura 6.6: K-means para datos de salida de PCA en la capa *dense_14*.

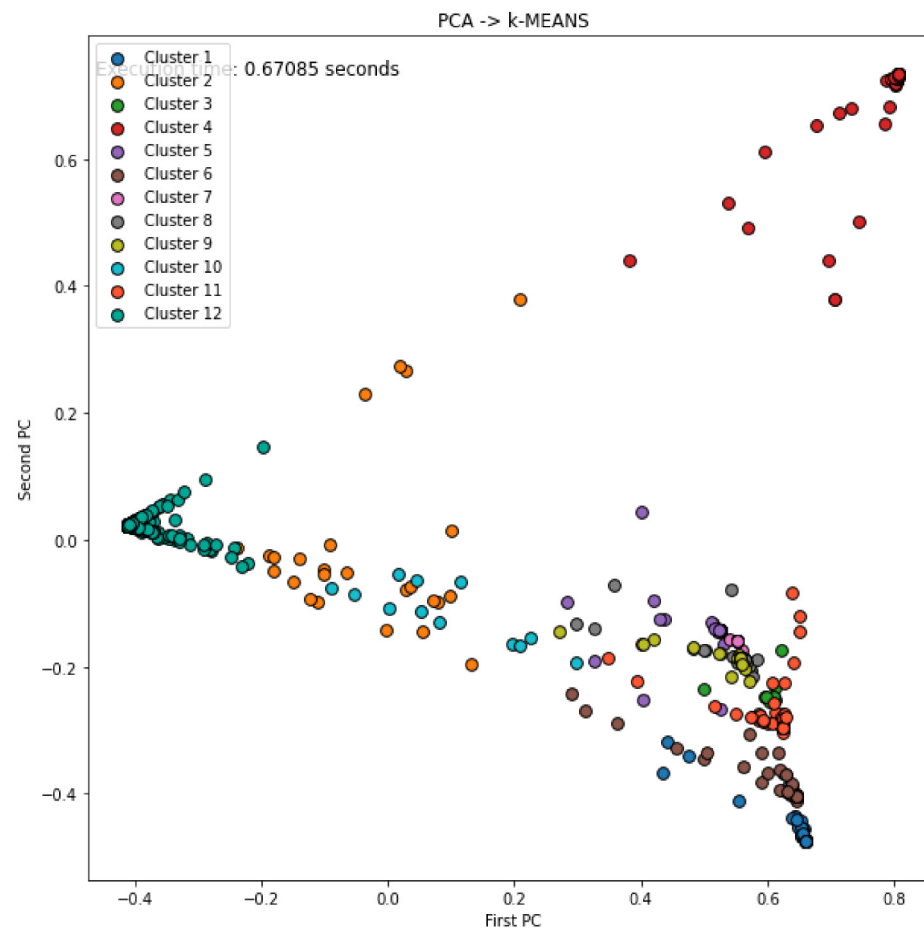


Figura 6.7: K-means para datos de salida de PCA en la capa *dense_15*.

A partir de estos gráficos podemos observar que:

1. En cuanto a tiempos de procesamiento, los métodos entre el más lento y el más rápido solo existe una diferencia de 0.35 [s], lo cuál es una magnitud mínima para términos de relevancia a la hora de diferenciar en que capa se demora más esta etapa. También podemos agregar que esta diferencia de tiempo no está ligada a la cantidad de características que se deben procesar, ya que podemos observar que en la capa que existen más de ellas (*dense_13*) no es la que tarda más ni la que tiene menor cantidad de características (*dense_15*) es la que demora menos.
2. Se puede apreciar que el único grupo donde existe algún tipo de patrón es en la capa *dense_15*, mientras que los otros se ven claramente mucho más concentrados en un punto lo cuál hace casi imposible divisar los distintos clusters. Aquí es donde nace la necesidad de buscar una manera de visualizar los resultados de mejor manera justificando la implementación de t-SNE como método de visualización.

6.3.3.2. DBSCAN

Para DBSCAN se realiza solo este proceso. De esta manera, para DBSCAN se debe determinar los 2 parámetros claves que son *epsilon* y *minPts*. El parámetro *epsilon* se utiliza para definir el tamaño de la vecindad alrededor de un punto en el espacio de características y por ende determina la distancia máxima entre dos puntos para considerarlos vecinos. Por su parte, *minPts* corresponde a la cantidad mínima de puntos requeridos para considerar a un grupo como cluster.

Para definir el valor de estos parámetros se realiza lo siguiente:

- *Epsilon*: Para determinar su valor se utiliza el método de los k vecinos más cercanos, donde primero se calcula la distancia al k-ésimo vecino más cercano para cada punto. Luego, se

grafica estas distancias ordenadas y buscamos un cambio significativo en la pendiente, y el valor del “eje y” corresponderá al valor de *epsilon* estimado adecuado. De igual manera este valor puede ir variándose para adecuarlo al número de clusters que uno desearía, pero siempre y cuando se entienda el contexto y las consecuencias que trae el manejo de este parámetro.

- *minPts*: Este parámetro es más contextual, por lo que en nuestro caso se le asigna el valor de 4, ya que se establece en un valor bajo para observar si es posible encontrar grupos de clusters pequeños que puedan ser relevantes a la hora de análisis.

Dicho esto, los gráficos obtenidos para este proceso son los siguientes:

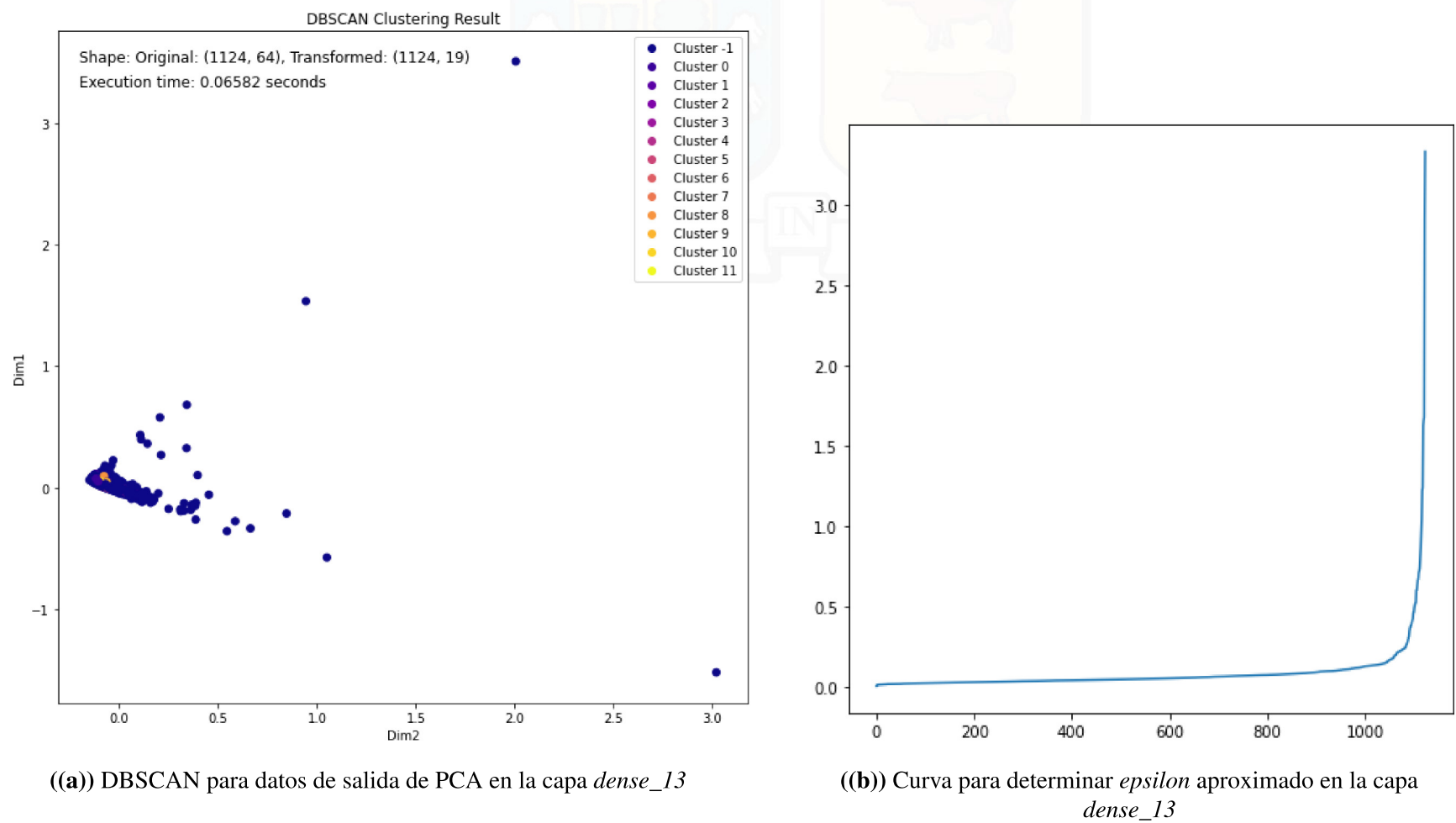


Figura 6.8: DBSCAN luego de la etapa de PCA en la capa *dense_13*.

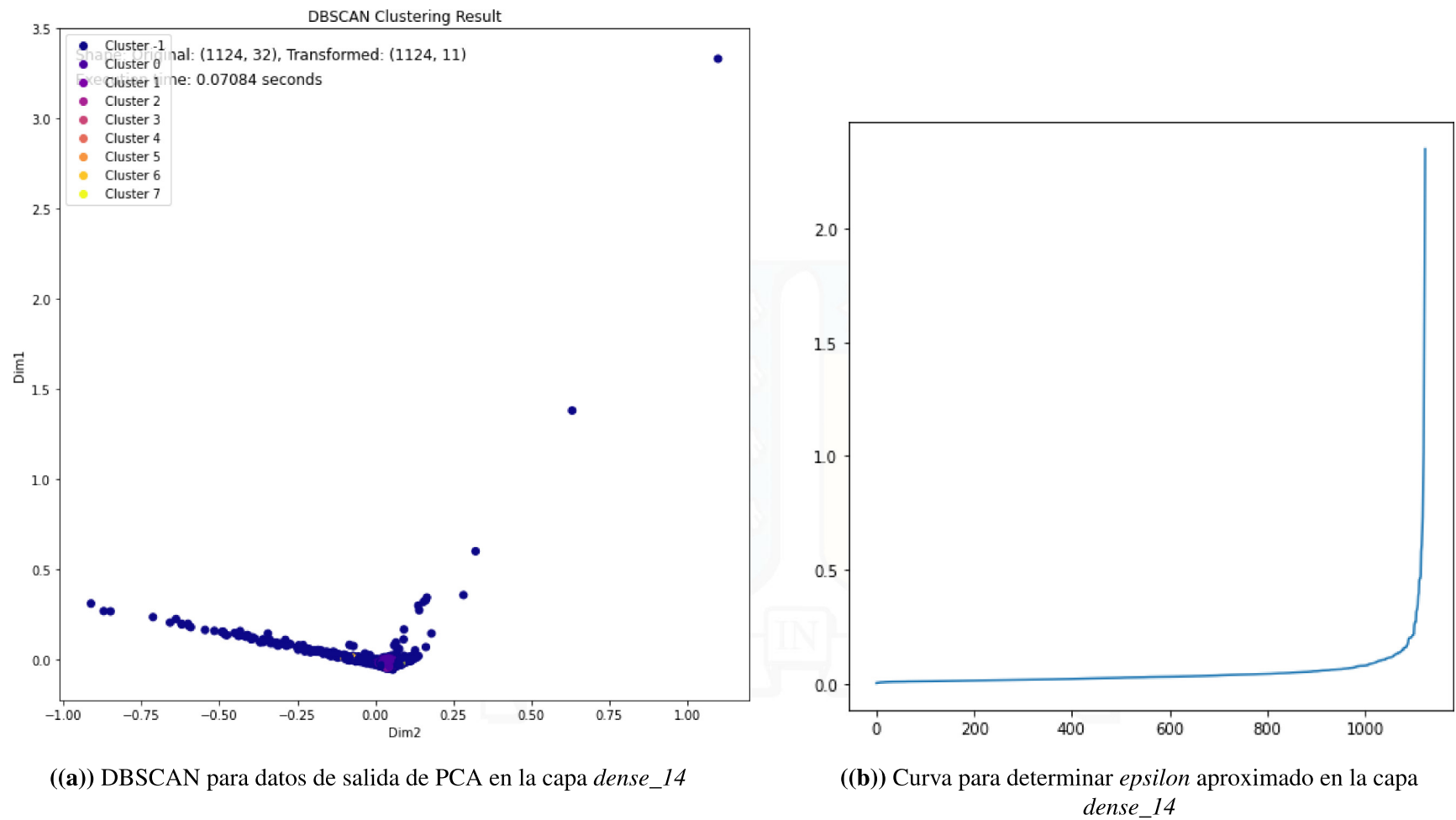


Figura 6.9: DBSCAN luego de la etapa de PCA en la capa *dense_14*.

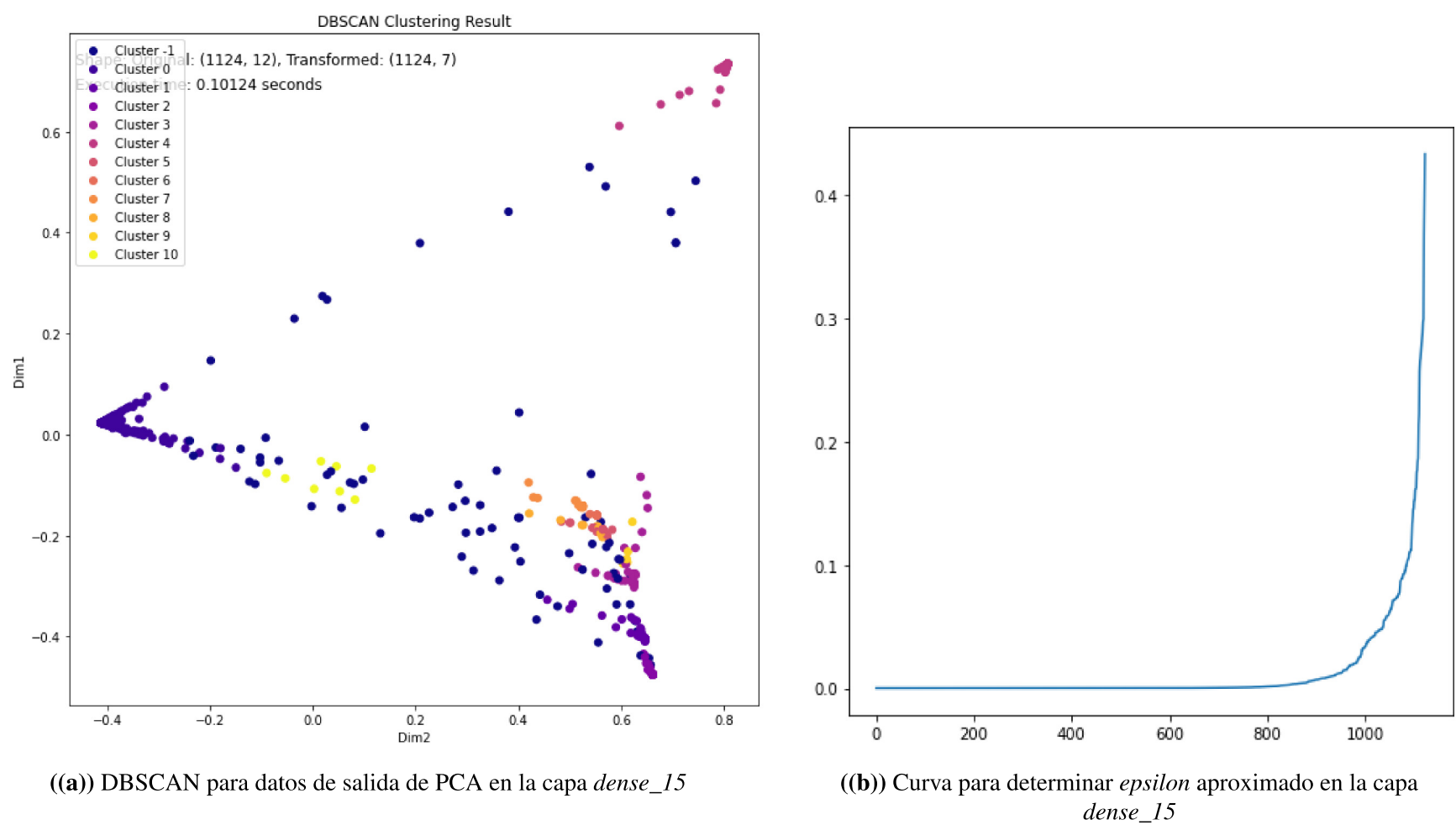


Figura 6.10: DBSCAN luego de la etapa de PCA en la capa *dense_15*.

Se puede observar que ocurre lo mismo que en K-Means, donde vemos la misma figura ya que esto no cambia por medio de los métodos de clustering, por lo que es difícil diferenciar los clusters

que se clasifican y por ende la necesidad de implementar t-SNE luego de esta etapa. Además, se puede observar por las distintas curvas para encontrar el *epsilon* adecuado que los que se utilizaron para cada caso fueron 0.03, 0.03 y 0.1 para *dense_13*, *dense_14* y *dense_15* respectivamente. Si bien para el último caso fue útil esta curva, para los otros dos no tanto ya que utilizando el valor que otorgaba el gráfico se podía observar que se generaba un solo cluster y ruido, por lo que no era óptimo para entender gráficamente los resultados, y por ello la adecuación de este parámetro a otro valor.

6.3.3.3. Spectral Clustering luego de DBSCAN

En esta opción se toman los resultados obtenidos en DBSCAN y se le aplican luego Spectral Clustering, lo que nos permite tomar formas más complejas de esta solución y agruparlas de distintas maneras, combinando así distintos métodos de agrupación e intentando obtener mejores resultados. Así, se obtienen resultados diversos a lo largo de las distintas capas, consiguiéndose los gráficos 6.11, 6.12 y 6.13:

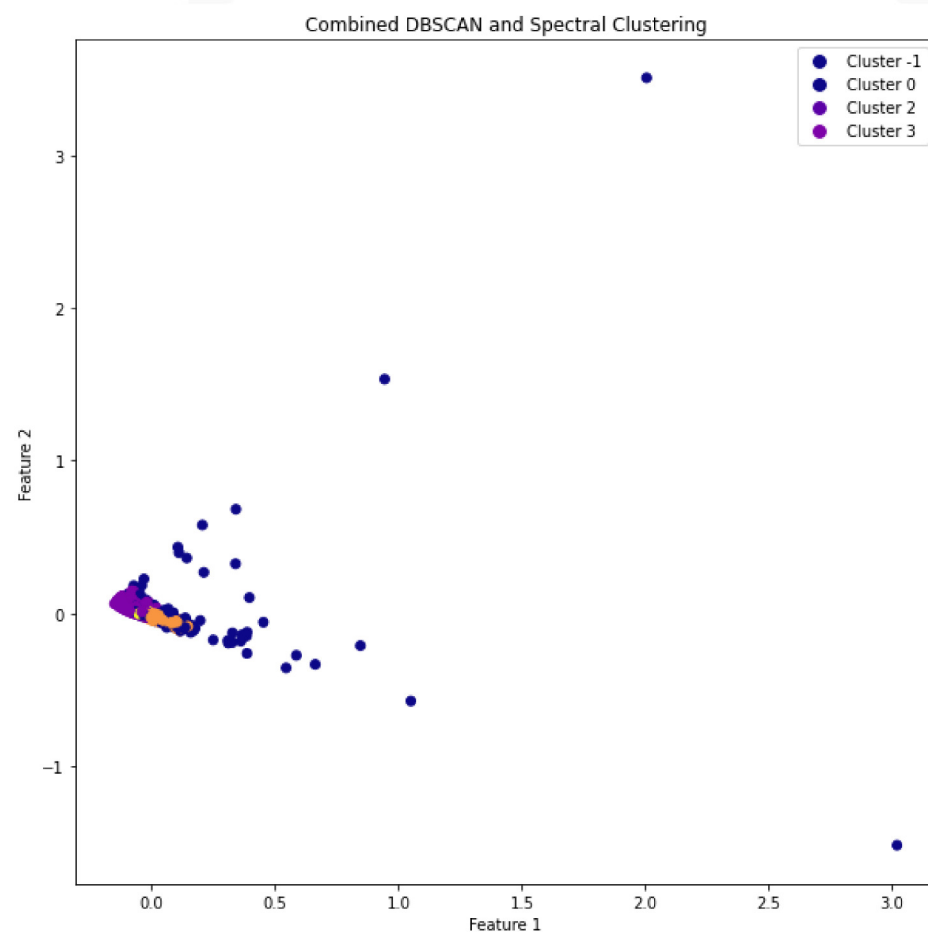


Figura 6.11: Spectral clustering para datos de salida de DBSCAN en la capa *dense_13*.

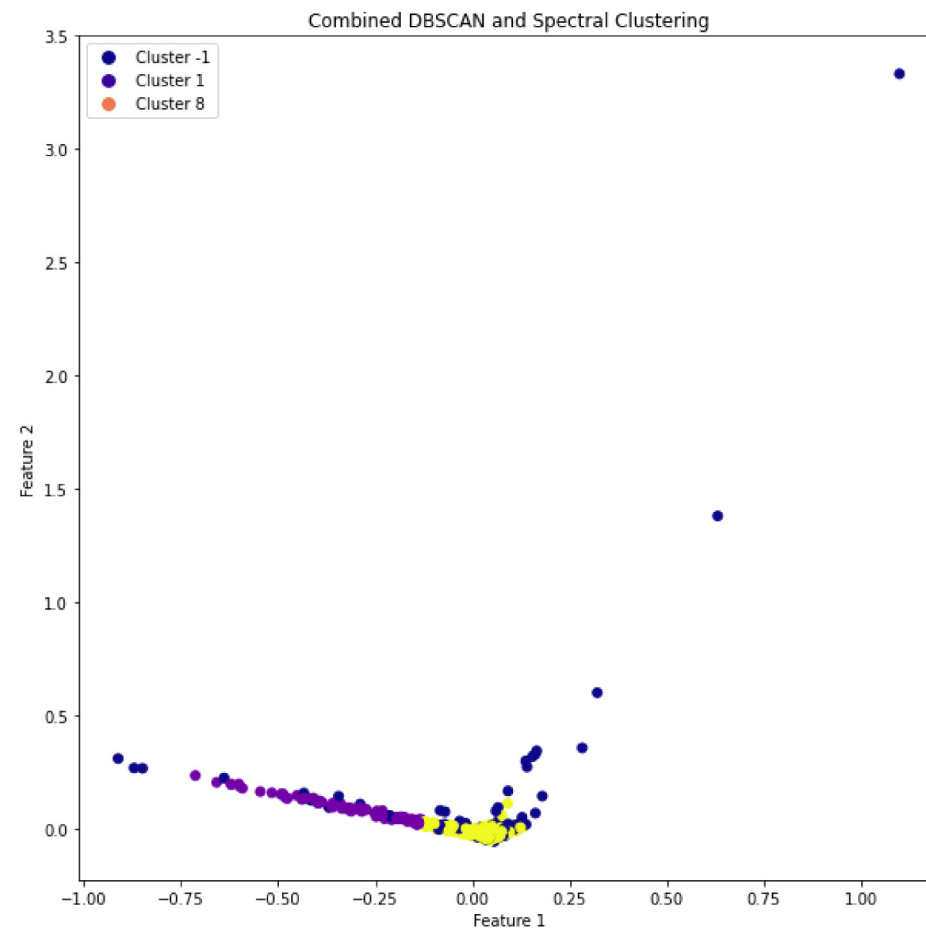


Figura 6.12: Spectral clustering para datos de salida de DBSCAN en la capa *dense_14*.

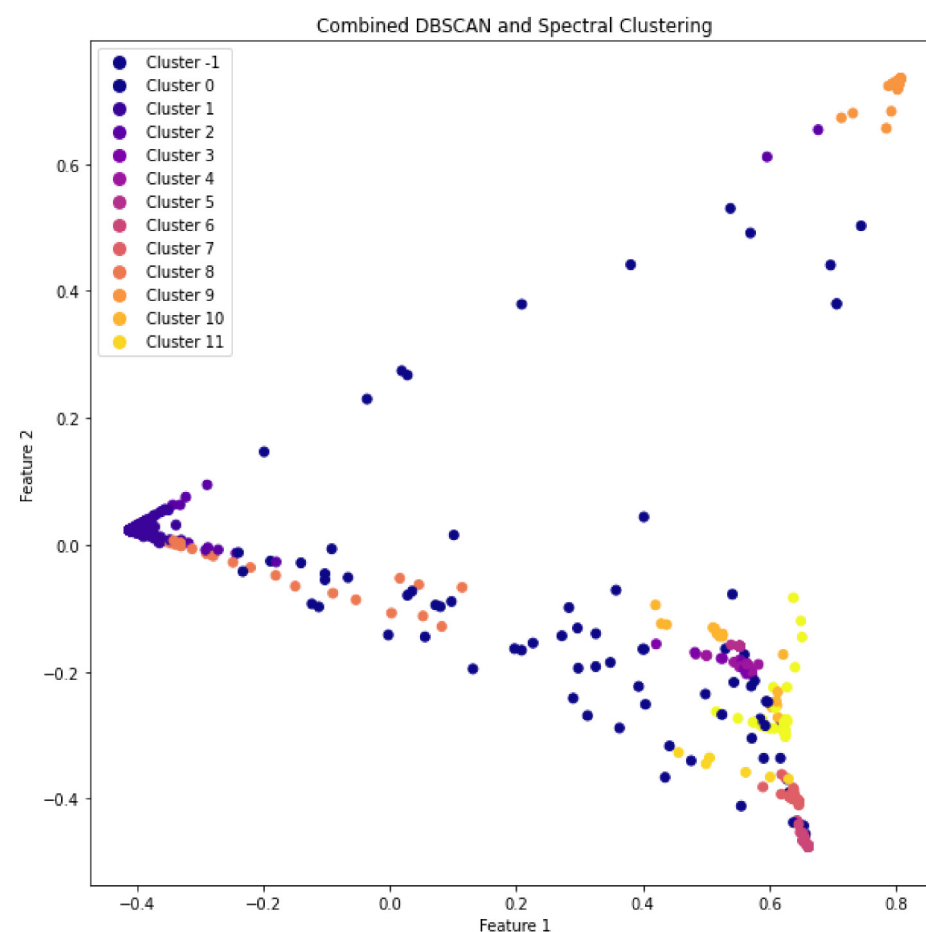


Figura 6.13: Spectral clustering para datos de salida de DBSCAN en la capa *dense_15*.

Es importante nombrar que los *epsilons* utilizados para el proceso previo de DBSCAN en esta etapa fueron los mismos que se utilizaron en la sección 6.3.3.2. De esto podemos observar que no se aprecian grandes cambios respecto a los obtenidos por el proceso de DBSCAN por si solo, incluso se puede ver que empeora el rendimiento para la capa *dense_14* al encontrar solo 2 clusters además

del ruido.

6.3.4. t-SNE

Como se mencionó en las secciones previas, esta etapa nos centraremos en el desarrollo de t-SNE como método de visualización, ya que con los gráficos obtenidos en las secciones 6.3.3.1, 6.3.3.2 y 6.3.3.3 los gráficos no permiten diferenciar los clusters más allá de lo poco que se podría observar en los gráficos de la capa *dense_15*. Por ello, se aplica t-SNE a todas las soluciones anteriores de clustering y distintas capas. También, se añade un gráfico que corresponde a los datos con las etiquetas reales, con la finalidad de realizar una comparativa de los resultados obtenidos y los resultados etiquetados que se encuentran en el dataset provisto. De esta manera, los resultados obtenidos son los siguientes:

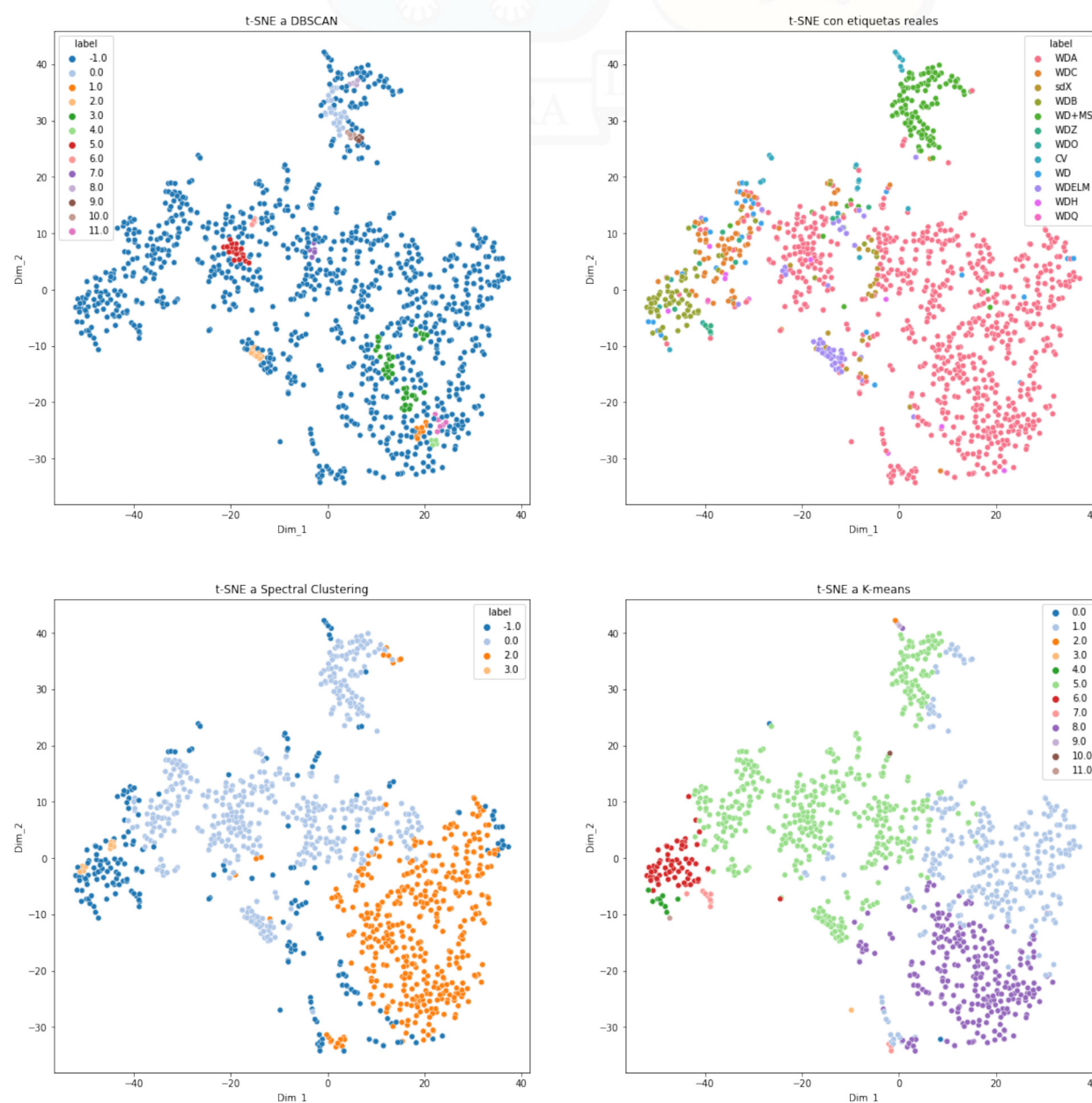


Figura 6.14: t-SNE a la salida de cada algoritmo de Clustering en la capa *dense_13*.

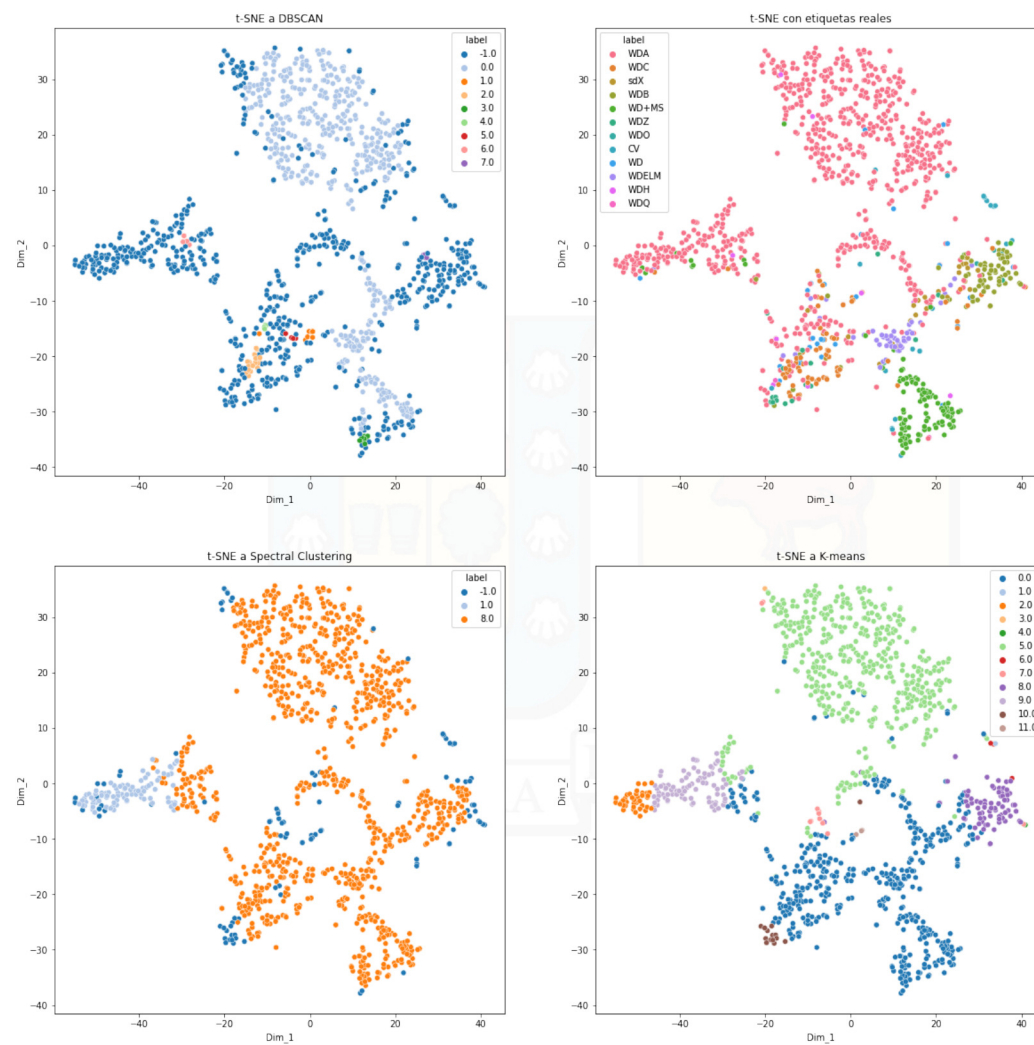


Figura 6.15: t-SNE a la salida de cada algoritmo de Clustering en la capa *dense_14*.

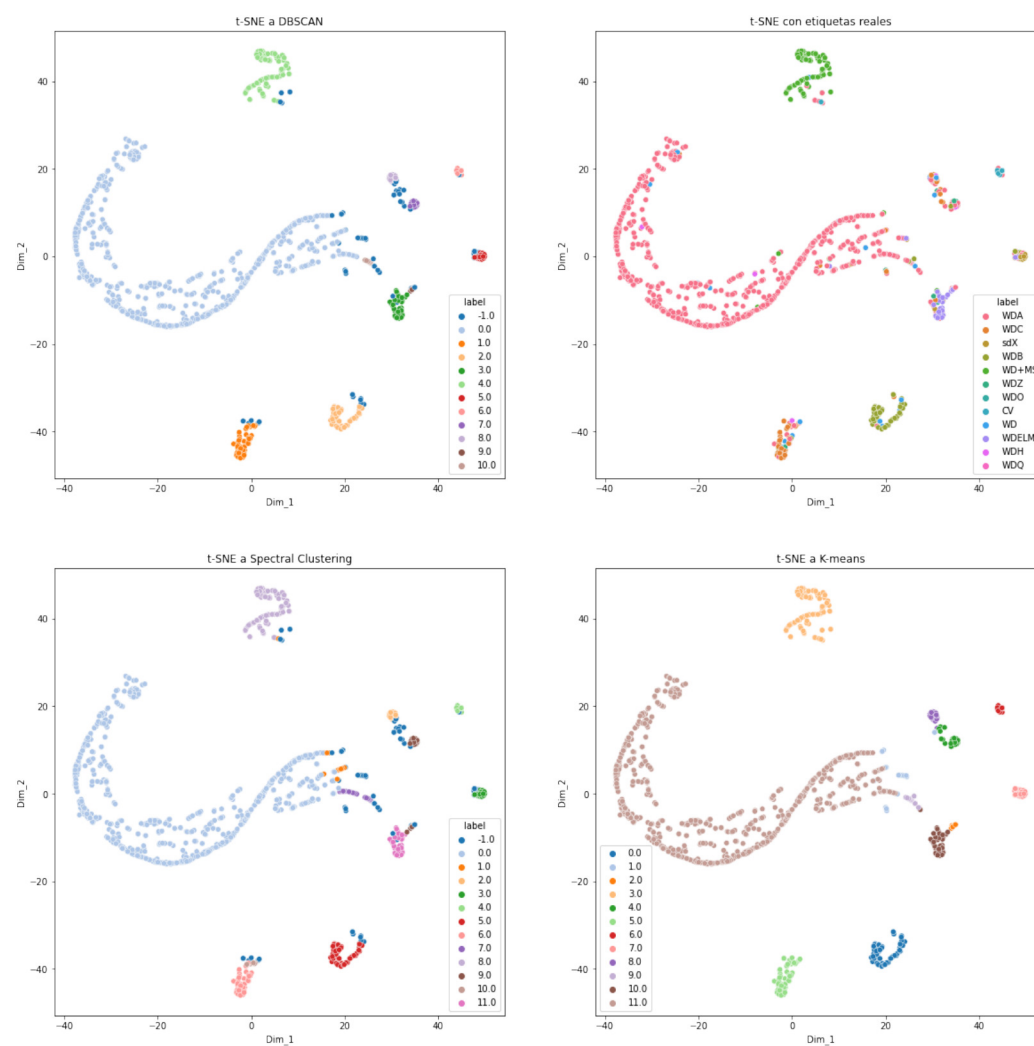


Figura 6.16: t-SNE a la salida de cada algoritmo de Clustering en la capa *dense_15*.

Gracias a t-SNE se puede observar claras mejoras en las distintas gráficas obtenidas, dejando mucho más claro los distintos clusters formados por los algoritmos previos. Se puede ver claramente que los resultados no son los mejores para las capas *dense_13* y *dense_14*, donde una de las razones pueden ser la cantidad de componentes, densidad de los datos en dicha capa, los datos de por sí, entre otros factores. Sin embargo, los resultados en la capa *dense_15* son muy positivos, ya que es posible identificar clusters de manera muy clara y precisa, donde todas las soluciones entregan resultados bastante confiables al tener resultados similares a los que se obtienen con los datos etiquetados. Es posible apreciar como el algoritmo de DBSCAN detecta justamente los 12 clusters correspondientes sin contar el ruido, lo que permite sacar conclusiones al respecto de ciertas clasificaciones para determinadas enanas blancas. Además, Spectral clustering y K-Means nos entregan 12 clusters (definidos manualmente) en cada una de sus soluciones, entregando un cluster menos que DBSCAN, lo que permite observar distintos clusters formados y por ende la posibilidad de un mayor análisis al respecto.

6.3.5. Representación gráfica

Si bien cada sección de las previas entrega una visualización de su salida y más importante la sección 6.3.4 ya que esta corresponde a la salida visual final por lo menos de la representación espacial de los datos, no corresponde a toda la información que se podría entregar como relevante para el usuario de este pipeline, donde se encuentra relevante por ende agregar:

- Gráficos de barra: Como se había mencionado en la sección 5.2, la salida sería flexible intentando abarcar la mayor cantidad de información útil para el usuario final, y la cantidad de datos por clusters es una de ellas, permitiendo analizar la comparativa de cuantos datos fueron asignados a cada uno de ellos y como lo es en los datos etiquetados, pudiendo de esta manera obtener una noción de que cluster podría corresponder a cuál.
- Matrices de confusión: Este elemento gráfico es realmente útil y se puede usar gracias a que contamos con los datos etiquetados, lo que nos permite ver la correspondencia entre las etiquetas reales y las otorgadas por los distintos algoritmos utilizados.

Sin embargo, los gráficos de barra y las matrices de confusión se realizan solo para los resultados de la capa *dense_15* debido a su relevancia y resultados obtenidos comparado a las otras capas. De esta forma, en las figuras a continuación se mostrarán los gráficos de barras y matrices de confusión correspondientes para cada caso:

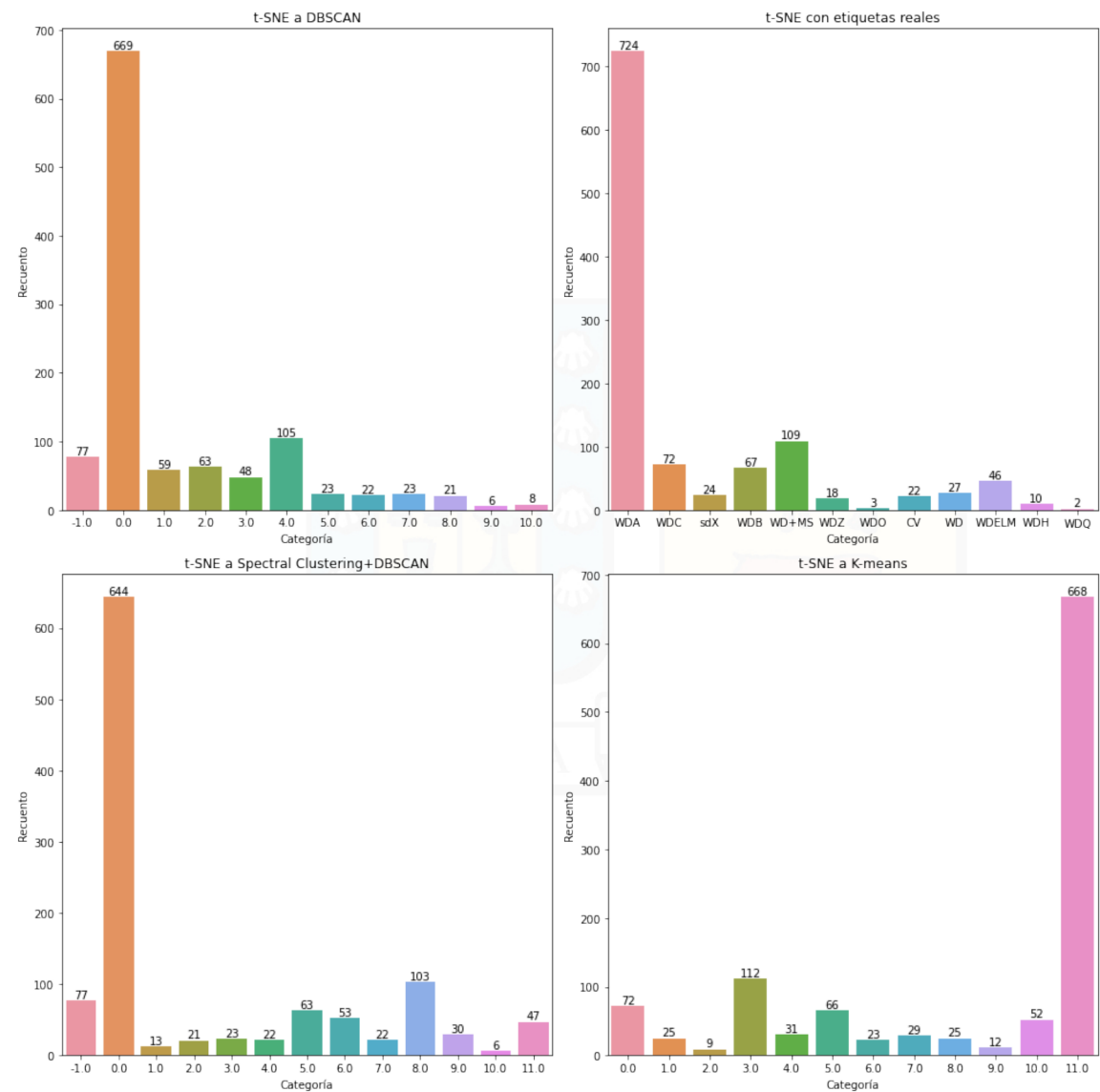
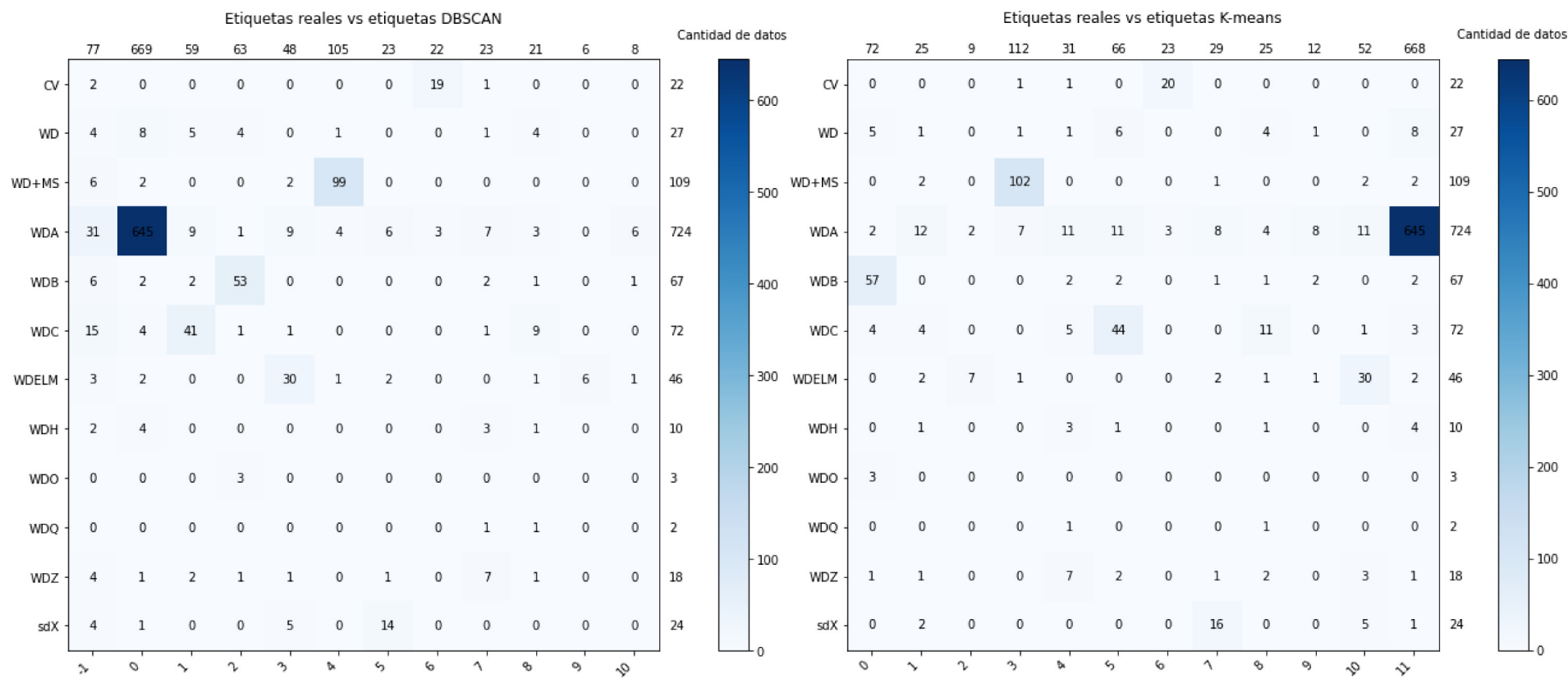


Figura 6.17: Gráficos de barra con cantidad de datos para cada cluster generado en la capa *dense_15*.



((a)) Matriz de confusión para solución con DBSCAN en la capa *dense_15*

((b)) Matriz de confusión para solución con K-means en la capa *dense_15*

Figura 6.18: Matrices de confusión para algoritmos con DBSCAN y K-Means en la capa *dense_15*.

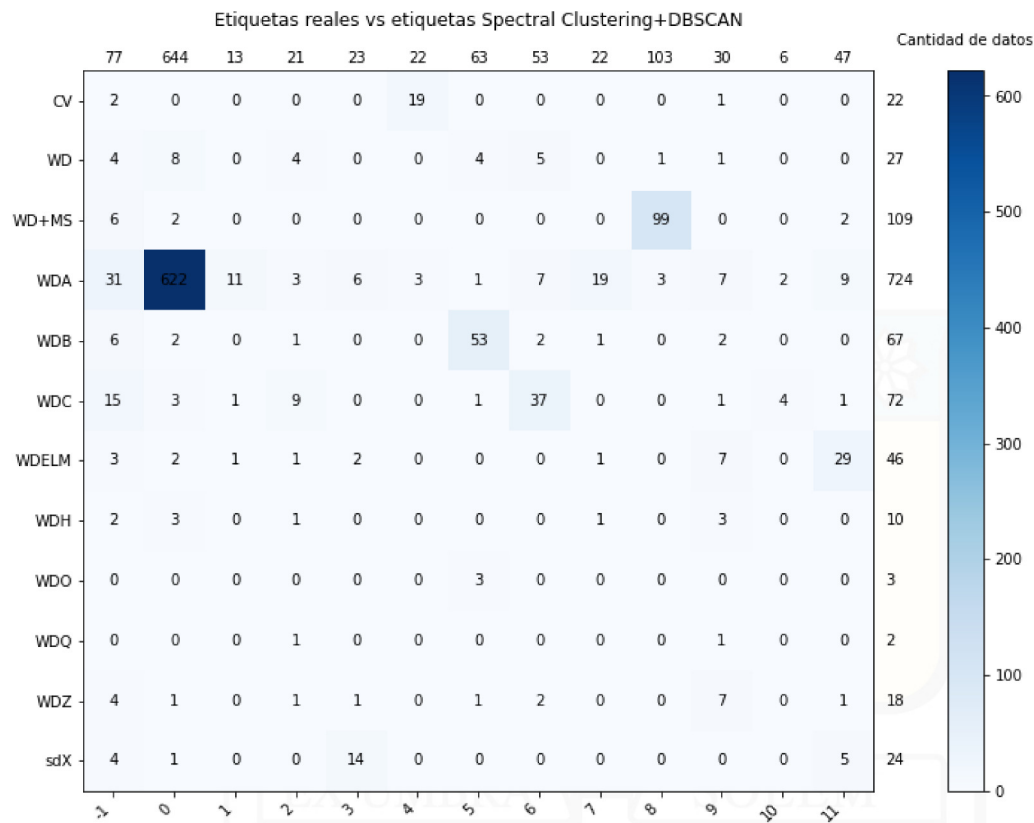


Figura 6.19: Matriz de confusión para solución con Spectral clustering en la capa *dense_15*.

6.3.6. Tiempos finales de procesamiento

Se toman los tiempos de procesamiento para cada caso, desde que se carga el modelo hasta la visualización de la matriz de confusión correspondiente, donde se obtiene:

Capa	Tiempo de procesamiento
dense_13	21.42464 [s]
dense_14	22.98102 [s]
dense_15	21.26763 [s]

Tabla 6.3: Tiempos finales de procesamiento para cada solución desarrollada.

La tabla muestra los tiempos de procesamiento que toma el algoritmo en realizar todas las etapas, considerando todos los algoritmos de clustering, es decir, DBSCAN, K-means y Spectral clustering + DBSCAN por cada capa. La idea de tomar los tiempos de procesamiento por capa considerando todas las alternativas y no individuales es por como está formado el sistema, y porque realizarlo de esta manera indicará el tiempo que le tomará al usuario ya que el pipeline que se le otorgará considerará todas las alternativas, lo que al fin y al cabo se traduce en más utilidad para el usuario.

7 | Conclusiones y posibles trabajos a futuro

7.1. Conclusiones

Luego de presentar todas las alternativas desarrolladas y sus respectivos resultados, se inicia viendo el cumplimiento de los objetivos planteados en 2.1 y las métricas impuestas en 6.2:

- **Tiempos de procesamiento cortos:** Como se mencionó en el mismo objetivo en un inicio, los tiempos son difíciles de asignarles un valor preciso ya que esto va totalmente de la mano con el tamaño del dataset que se utilice, además de otros factores tales como la calidad del dispositivo donde se ejecute el código como se mencionó en 6.1.

De igual manera, con el sistema mencionado previamente se cumplió esta imposición de que el proceso en su totalidad tomará en la escala de los segundos, como se puede observar en la tabla 6.3. Ninguna alternativa toma más de 23 segundos, lo cuál es bastante veloz para la cantidad de gráficas que este genera y la cantidad de alternativas que este abarca.

- **Pipeline correctamente particionado y general:** Si bien este objetivo no se puede visualizar al no mostrar el código del pipeline en este documento, se puede observar a través del desarrollo de las etapas en 6.3 que es posible apreciar como se distinguen las etapas y sus salidas de manera sólida y concisa, mostrando gráficos con resultados para cada una de ellas y siendo etapas separadas de las demás, por lo que resulta fácil de entender en que lugar esta cada proceso y en que lugar buscar las respuestas deseadas.
- **Cantidad de clusters detectados:** Este objetivo se mencionó que dependía netamente de la alternativa seleccionada, ya que en los desarrollos realizados se trabajó con algoritmos que requerían escribir de manera manual el número de clusters (K-Means) y otros que no (DBSCAN). Sin embargo, en DBSCAN se puede observar a través de las figuras 6.8, 6.9 y 6.10 que se encontraron 12, 8 y 11 clusters respectivamente sin considerar el cluster generado para clasificar el ruido. Esto indica que tenemos una cantidad de clusters bastante cercana a la que el modelo del que formará parte este pipeline clasifica (12 clases en total), por lo que se puede contar que este objetivo fue alcanzado. Sin embargo, cabe mencionar que si bien la cantidad de clusters puede ser parecida o igual a las buscadas, es muy importante que los clusters generados tengan sentido. Esto es apreciable en el caso de la capa *dense_15* donde se observan clusters generados muy parecidos a los que poseen etiquetas reales, y no tan parecidos en las otras dos capas.
- **Varianza explicada:** Esta métrica se puede ver cumplida dentro de las figuras 6.2, 6.3 y 6.4,

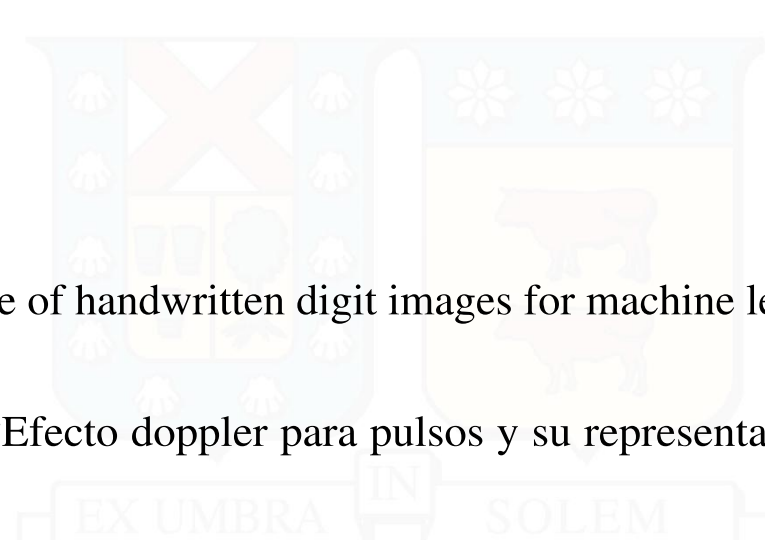
ya que como se mencionó en la sección 6.3.2 el valor de la varianza explicada puede ser introducida manualmente en la función que realiza el proceso de PCA, por lo que se obtendrá un valor mayor al parámetro introducido que en nuestro caso fue de 0.95 o 95 % para cumplir con lo solicitado y obtener resultados óptimos.

Con esto dicho, se puede concluir a través de los desarrollos planteados que lo mejor para este caso no es necesariamente buscar la solución más efectiva, más bien buscar la mayor cantidad de soluciones que cumplan con los objetivos planteados. Esto se puede ver cumplido a través de los diversos desarrollos, gráficas y resultados finales, todo esto con el propósito de generar un mejor Pipeline para el usuario final, del cual este pudiese extraer la mayor cantidad de información útil para un próspero análisis de resultados.

7.2. Posibles trabajos a futuro

En el desarrollo de este trabajo siempre se tuvo en mente más allá de completar con los objetivos y resultados esperados, al usuario final y el cómo este usaría el pipeline, por lo que uno de los posibles trabajos a futuro que este sistema podría tener más allá de sus funcionalidades es el desarrollo de una interfaz gráfica o programa que pueda facilitar su uso. La idea sería que el usuario solo interactuara con esta interfaz, subiendo los modelos o datasets a través de botones de búsqueda de archivos; cambiar parámetros o valores como el *epsilon*, minPts, varianza explicada mínima, entre otros a través de *sliders* o *scrolls*; o tener la posibilidad de interactuar con los gráficos para un uso más completo, cómodo, sencillo y rápido de lo que ya es el pipeline actual. Esto complementaría los requisitos funcionales del sistema y estando centrado más en los requisitos no-funcionales de este generando finalmente un servicio más integral.

Bibliografía

- 
- [1] D. L., “The mnist database of handwritten digit images for machine learning research,” 2012. 2.3
- [2] A. R. y Reinaldo Welte, “Efecto doppler para pulsos y su representación en el plano (x, t),” 2009. 3.1.1
- [3] M. A. Taverna, “Función de luminosidad de galaxias a partir de corrimientos al rojo fotométricos,” 2014. 3.1.1
- [4] M. A. V. B. S. Shruti Sehgal, Harpreet Singh, “Data analysis using principal component analysis,” 2014. 3.1.2.1
- [5] S. N. S. R. T. Chandrakala, “An analysis of outlier detection through clustering method,” 2020. 3.1.3
- [6] S. B. Rupanka Bhuyan, “A survey of some density based clustering techniques,” 2023. 3.1.3
- [7] W. R. A. Maćkiewicz, “Principal components analysis (pca),” 1992. 4.1.1
- [8] 4.1.2
- [9] R. S. Mohiuddin Ahmed and S. M. S. Islam, “The k-means algorithm: A comprehensive survey and performance evaluation,” 2020. 4.2.1
- [10] P. M. D. Comaniciu, “Mean shift: A robust approach toward feature space analysis,” pp. 608–609, 2002. 4.2.3
- [11] M. C. Q. W. Q. H. J. Xin Wang, Yue Yang and H. Wang, “Agnes-smote: An oversampling algorithm based on hierarchical clustering and improved smote,” 2020. 4.2.4
- [12] Y. W. A. Y. Ng, M. I. Jordan, “On spectral clustering: Analysis and an algorithm,” 2001. 4.2.5
- [13] Y. T. Dongkuan Xu, “A comprehensive survey of clustering algorithms,” 2015. 5.1.2