

Análisis de componentes principales y sus aplicaciones en el campo del análisis de malware

Carmen Johana Calderón Chona

June 03, 2025

Contents

1. Introducción	5
2. Conceptos de álgebra lineal y probabilidad	6
2.1. Álgebra lineal:	6
2.1.0.1. Matrices	6
2.1.0.2. Determinante de una matriz cuadrada	7
2.1.0.3. Inversa de una matriz cuadrada	7
2.1.0.4. Algunas matrices importantes	7
2.1.0.5. Valores y vectores propios	8
2.1.0.6. Teorema espectral	9
2.2. Variables aleatorias, esperanza y medidas muestrales	9
2.3. Vectores Aleatorios y Matriz de Covarianza	11
3. Sobre PCA: Procedimiento y detalles importantes	12
3.1. Procedimiento para realizar el análisis de componentes principales	12
3.2. Matriz de covarianza de un conjunto de datos	15
3.3. Detalles importantes de PCA	18
3.3.0.1. Eliminar columnas con varianza cercana a 0	18
3.3.0.2. Filtrar variables numéricas que tengan sentido conjunto	18
3.3.0.3. Asegurar que estén escaladas	19
3.3.0.4. Manejo de valores faltantes antes de aplicar PCA	19
3.3.0.5. Consideraciones acerca de lo que es un valor propio “pequeño”	19
3.3.0.6. Escalado antes de PCA	19
3.3.1. Distancia de Mahalanobis y PCA robusto	20
3.3.1.1. Detección de outliers y la necesidad de una medida multivariada . . .	20
3.3.1.2. La distancia de Mahalanobis	21
3.3.1.3. Ejemplo ilustrativo (basado en Wikipedia)	22
3.3.1.4. Consideraciones sobre la invertibilidad de la matriz de covarianza y su estimación robusta	22
3.3.1.5. Estimación robusta con Minimum Covariance Determinant (MCD) .	23
3.3.1.5.0.1. Funcionamiento del MCD	24
3.3.1.5.0.2. Comparación con otros estimadores robustos .	24
3.3.1.5.0.3. Implementación eficiente: FAST-MCD	25
3.3.1.6. PCA Robusto	26
4. PCA y análisis de malware	28
4.1. Modelos de predicción para analizar malware	28
4.2. Aplicación de PCA en la detección de malware	29
4.3. Spectral clustering	30
4.3.1. Algunos conceptos previos	30
4.3.1.1. Grafo de similitud	30
4.3.1.1.1. Construcción del grafo de similitud	31
4.3.1.2. Matriz Laplaciana	31
4.3.1.2.1. Definición	32
4.3.1.2.2. Otras variantes	32
4.3.2. Algoritmo	32

4.3.2.1. Pasos generales del algoritmo	33
4.3.2.2. Problemas de implementación	34
4.3.2.3. Descripción del algoritmo U-SPEC (Ultra-Scalable Spectral Clustering)	36
4.3.2.4. Consideraciones sobre los parámetros de USPEC	37
4.3.2.5. Descripción del algoritmo Ultra-Scalable Ensemble Clustering (U- SENC)	38
4.3.2.5.1. Parámetros del algoritmo USENC(fea, k, m, distance, p=1000, Knn=5, lowK=2, highK=5)	39
4.3.3. Estudio de la estructura de los datos usando USPEC y USENC	40
4.4. Implementación: Microsoft Malware Prediction	40
4.4.0.1. Primer entregable	40
4.4.0.2. Segundo entregable	41
4.4.0.2.0.1. Entrenamiento de modelos: Fase 1 – Baseline .	42
4.4.0.2.0.2. Evaluación de modelos avanzados	42
4.5. Conclusiones	43
4.6. Glosario	44
Bibliography	46
Index of Figures	48

1. Introducción

En la era digital actual, los ciberataques representan una de las amenazas más significativas para la seguridad de los sistemas informáticos. Es así como la detección temprana de malware —software malicioso diseñado para dañar o comprometer dispositivos— se ha convertido en una tarea crítica, tanto para empresas como para usuarios individuales. En este contexto, las técnicas de aprendizaje automático han demostrado ser herramientas prometedoras para anticipar ataques basándose en grandes volúmenes de datos sobre el comportamiento y las características del sistema.

Sin embargo, estos conjuntos de datos suelen tener una alta dimensionalidad, lo que puede introducir ruido, dificultar el entrenamiento de modelos eficientes y aumentar el riesgo de sobreajuste. Una solución a este problema es el uso del Análisis de Componentes Principales (PCA), una técnica estadística que permite transformar el espacio de variables originales en un conjunto reducido de componentes no correlacionados, preservando la mayor parte de la varianza de los datos.

Este trabajo se enfoca en responder la siguiente pregunta de investigación: ¿Es posible construir un modelo de aprendizaje automático que, utilizando PCA como paso de preprocesamiento, logre predecir correctamente si un dispositivo será infectado por malware en la mayoría de los casos? Para ello, se utiliza el conjunto de datos de la competencia “Microsoft Malware Prediction” disponible en Kaggle, con el objetivo de evaluar si PCA mejora el rendimiento del modelo y facilita el análisis de las variables más relevantes.

2. Conceptos de álgebra lineal y probabilidad

Antes de adentrarnos en el tema de estudio central en este documento, es fundamental repasar algunos conceptos básicos tanto de álgebra lineal como de probabilidad. Esta sección tiene como objetivo proporcionar las herramientas teóricas necesarias para comprender con claridad los resultados que se presentan más adelante.

2.1. Álgebra lineal:

2.1.0.1. Matrices

Una matriz es una disposición rectangular de números (llamados elementos) organizados en filas y columnas. Por ejemplo, una matriz de m filas y n columnas se llama matriz de tamaño $m \times n$, y se representa comúnmente como:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \quad (1)$$

Si tienes una matriz A de tamaño $m \times n$, su traspuesta, denotada como A^T , es una matriz de tamaño $n \times m$ tal que:

$$(A^T)_{ij} = A_{ji} \quad (2)$$

Por ejemplo, dada la matriz

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad (3)$$

su traspuesta es

$$A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \quad (4)$$

Algunas propiedades de la traspuesta, cuya demostración se puede consultar en [1] son:

- $(A^T)^T = A$
- $(A + B)^T = A^T + B^T$
- $(kA)^T = kA^T$, para cualquier escalar k
- $(AB)^T = B^T A^T$

Es decir, el elemento en la fila i y columna j de la traspuesta es igual al elemento en la fila j y columna i de la matriz original.

Una matriz cuadrada es una matriz con el mismo número de filas y columnas, es decir, de dimensión $n \times n$. Estas matrices son especialmente importantes porque permiten definir operaciones como el determinante, la traza, o conceptos como autovalores y autovectores.

2.1.0.2. Determinante de una matriz cuadrada

El determinante de una matriz cuadrada es un número que resume ciertas propiedades de la matriz, como si es invertible o no. El determinante de una matriz A de orden n se denota como $\det(A)$ o $|A|$.

Para matrices grandes, una forma de calcular el determinante es usando la expansión por cofactores (o fórmula de Laplace). Esta se hace usualmente a lo largo de una fila o una columna. Por ejemplo, para una matriz 3×3 :

$$\det(A) = a_{11} \cdot C_{11} - a_{12} \cdot C_{12} + a_{13} \cdot C_{13} \quad (5)$$

donde C_{ij} es el **cofactor**, que se calcula como el determinante del menor (la submatriz que resulta al eliminar la fila i y la columna j), multiplicado por $(-1)^{i+j}$.

Algunas propiedades importantes del determinante son las siguientes:

- Si $\det(A) \neq 0$, entonces A es invertible.
- El determinante de una matriz triangular (superior o inferior) es el producto de sus elementos diagonales.
- $\det(AB) = \det(A) \det(B)$
- $\det(A^T) = \det(A) \det(A^T) = \det(A)$

Para ver una demostración de estas propiedades y otras, consulte [1].

2.1.0.3. Inversa de una matriz cuadrada

La inversa de una matriz cuadrada A (de tamaño $n \times n$) es otra matriz A^{-1} tal que:

$$A \cdot A^{-1} = A^{-1} \cdot A = I \quad (6)$$

donde $I = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}$ es la matriz identidad de tamaño $n \times n$, es decir, una matriz con unos en la diagonal principal y ceros en el resto.

Una matriz cuadrada A tiene inversa (es invertible o no singular) si y solo si su determinante es distinto de cero.

Si $\det(A) = 0$, la matriz no tiene inversa y se llama singular o no invertible.

2.1.0.4. Algunas matrices importantes

Una matriz triangular es una matriz cuadrada en la que todos los elementos por encima o por debajo de la diagonal principal son cero. En el primer caso, a la matriz triangular se le denomina inferior y en el segundo, se dice que la matriz triangular es superior.

Por ejemplo, las matrices $\begin{bmatrix} 2 & -1 & 3 \\ 0 & 5 & 4 \\ 0 & 0 & 7 \end{bmatrix}$ y $\begin{bmatrix} 4 & 0 & 0 \\ -2 & 1 & 0 \\ 5 & 3 & 6 \end{bmatrix}$ son triangulares superior e inferior, respectivamente.

Una matriz diagonal, por otro lado, es una matriz cuadrada en la que todos los elementos fuera de la diagonal principal son cero:

$$D = \begin{pmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d_n \end{pmatrix} \quad (7)$$

Una matriz simétrica es una matriz cuadrada $A \in \mathbb{R}^{n \times n}$ tal que $A = A^T$

Es decir, sus elementos son simétricos respecto a la diagonal principal $a_{ij} = a_{ji}$.

Las matrices simétricas tienen propiedades muy importantes, especialmente en el análisis de sistemas físicos, optimización y más.

Adicionalmente, tenemos el concepto de matrices de permutación. Una matriz de permutación es una matriz cuadrada que se obtiene al permutar las filas (o columnas) de la matriz identidad. Ejemplo de matriz identidad I_3 :

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (8)$$

Si permutamos la primera y segunda fila, obtenemos:

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (9)$$

Las matrices de permutación cumplen con las siguientes propiedades:

- Son ortogonales, esto es, $P^{-1} = P^T$
- Al multiplicar una matriz A por una matriz de permutación P , se reordenan las filas o columnas de A . Por ejemplo, si M es una matriz de tamaño apropiado, el producto, PM es sólo una permutación de las filas de M y MP es sólo una permutación de las columnas de M .
- Se usan en algoritmos de factorización (como factorización LU) y en métodos numéricos para mejorar la estabilidad de los cálculos.

2.1.0.5. Valores y vectores propios

Dado una matriz cuadrada A , un vector propio $v \neq 0$ y un valor propio $\lambda \in \mathbb{R}$ (o \mathbb{C}) cumplen:

$$Av = \lambda v \quad (10)$$

Esto significa que aplicar la matriz A al vector v no cambia su dirección, solo su escala.

Para encontrar los valores propios, se resuelve la ecuación característica:

$$\det(A - \lambda I) = 0 \quad (11)$$

Los λ que satisfacen esta ecuación son los valores propios. Para cada valor propio, se pueden encontrar los vectores propios resolviendo el sistema:

$$(A - \lambda I)v = 0 \quad (12)$$

2.1.0.6. Teorema espectral

El Teorema Espectral establece que toda matriz simétrica real $A \in \mathbb{R}^{n \times n}$ puede ser **diagonalizada** por una matriz ortogonal. Es decir, existe una matriz ortogonal Q y una matriz diagonal D tal que:

$$A = QDQ^T \quad (13)$$

donde Q tiene como columnas a los vectores propios ortonormales de A y D contiene en su diagonal los valores propios reales de A .¹

Para una ver una prueba del mismo, consulte [2]

2.2. Variables aleatorias, esperanza y medidas muestrales

Una variable aleatoria es una función que asigna un número real a cada resultado posible de un experimento aleatorio. Por ejemplo, al lanzar un dado, podemos definir una variable aleatoria X que tome el valor del número que sale. Esta función permite modelar situaciones inciertas de manera cuantitativa.

Una de las nociones fundamentales asociadas a una variable aleatoria es su esperanza matemática o valor esperado, denotado $E[X]$. Esta representa el promedio ponderado de los posibles valores de la variable, teniendo en cuenta su probabilidad de ocurrencia. Es decir, no es simplemente un promedio aritmético, sino un promedio ajustado por cuán probable es cada valor. La esperanza ofrece una primera aproximación del “centro” de la distribución de la variable aleatoria.

En la práctica, sin embargo, rara vez conocemos la distribución completa de una variable. En su lugar, disponemos de un conjunto finito de observaciones o datos, llamado muestra. A partir de esta muestra, estimamos la esperanza mediante la media muestral, que es el promedio aritmético de los datos observados. Si contamos con n observaciones X_1, X_2, \dots, X_n la esperanza muestral se calcula como:

¹Note que los valores propios de una matriz simétrica real son reales. En efecto, suponga que $v \in \mathbb{C}^n$ es un vector propio de A , con valor propio $\lambda \in \mathbb{C}$. Entonces:

$$Av = \lambda v \quad (14)$$

Ahora, tome el producto interno complejo v^*Av , donde v^* es el conjugado transpuesto de v (a saber, el transpuesto de v con sus elementos conjugados). Se tendría entonces que $v^*Av = \lambda v^*v$. Pero como A es simétrica real, también se cumple que $v^*Av = (Av)^*v = (\lambda v)^*v = \lambda^*v^*v$. Entonces

$$\lambda v^*v = \lambda^* v^*v \quad (15)$$

Y como $v^*v > 0$ (producto interno positivo si $v \neq 0$):

$$\lambda = \lambda^* \quad (16)$$

lo cual implica que $\lambda \in \mathbb{R}$.

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i \quad (17)$$

Otra medida esencial es la varianza, que cuantifica cuánto se dispersan los valores de la variable respecto a su media. Formalmente, la varianza de una variable aleatoria se define como el valor esperado del cuadrado de las desviaciones respecto a la media:

$$\text{Var}(X) = E[(X - E[X])^2] \quad (18)$$

Nuevamente, como en la mayoría de los casos no conocemos $E[X]$, utilizamos la **varianza muestral**, que estima la dispersión a partir de los datos. Esta se calcula como el promedio de los cuadrados de las diferencias entre cada valor observado y la media muestral, pero dividiendo entre $n - 1$ en lugar de n . Este ajuste, conocido como corrección de Bessel, compensa el sesgo introducido al estimar la media a partir de los datos:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2 \quad (19)$$

La división entre $n - 1$ asegura que el estimador sea insesgado, es decir, que en promedio produzca el valor correcto de la varianza poblacional.

Cuando se analizan dos variables aleatorias a la vez, interesa saber si existe alguna relación entre ellas. Para eso se utiliza la covarianza, una medida que describe cómo varían conjuntamente. Si al aumentar una variable la otra también tiende a aumentar, la covarianza es positiva; si al aumentar una variable la otra tiende a disminuir, es negativa. La covarianza entre dos variables aleatorias X y Y se define como:

$$\text{Cov}(X, Y) = E[(X - E[X])(Y - E[Y])] \quad (20)$$

En la práctica, la covarianza muestral se estima a partir de datos con la fórmula:

$$s_{XY} = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y}) \quad (21)$$

La covarianza muestral permite detectar patrones de dependencia entre dos variables dentro de una muestra: si ambas tienden a aumentar o disminuir juntas, su covarianza será positiva; si una crece mientras la otra decrece, será negativa. Sin embargo, el valor de la covarianza depende de las unidades de medida de las variables, lo que dificulta su comparación entre distintos contextos o escalas.

Para solucionar este problema se utiliza la correlación, que es una versión normalizada de la covarianza. La correlación muestral entre dos variables X y Y se define como:

$$r_{XY} = \frac{s_{XY}}{s_X s_Y} \quad (22)$$

donde s_{XY} es la covarianza muestral entre X y Y , y s_X , s_Y son las desviaciones estándar muestrales de X y Y , respectivamente.

El resultado es un número entre -1 y 1 que se interpreta de la siguiente forma:

- Si $r_{XY} = 1$, hay una correlación perfectamente positiva (ambas variables aumentan juntas de manera proporcional).
- Si $r_{XY} = -1$, hay una correlación perfectamente negativa (una sube mientras la otra baja en proporción).
- Si $r_{XY} = 0$, no hay una relación lineal aparente entre las dos variables.

2.3. Vectores Aleatorios y Matriz de Covarianza

Un vector aleatorio de dimensión n es un vector cuyas entradas son variables aleatorias:

$$X = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_n \end{bmatrix} \quad (23)$$

Si cada para cada componente X_i , $E[X_i]$ existe, se define **la esperanza del vector aleatorio** como:

$$E[X] = \begin{bmatrix} E[X_1] \\ E[X_2] \\ E[X_3] \\ \vdots \\ E[X_n] \end{bmatrix} \quad (24)$$

La **varianza del vector aleatorio** se generaliza a través de la siguiente

$$V(X) = E[(X - E[X])(X - E[X])^T] \quad (25)$$

Esta matriz se conoce como la matriz de covarianza del vector X y se denota por Σ . Sus entradas representan las covarianzas entre los componentes del vector:

$$\Sigma = \begin{pmatrix} \text{Var}(X_1) & \text{Cov}(X_1, X_2) & \dots & \text{Cov}(X_1, X_n) \\ \text{Cov}(X_2, X_1) & \text{Var}(X_2) & \dots & \text{Cov}(X_2, X_n) \\ \vdots & \vdots & \dots & \vdots \\ \text{Cov}(X_n, X_1) & \text{Cov}(X_n, X_2) & \dots & \text{Var}(X_n) \end{pmatrix} \quad (26)$$

Entre algunas de las propiedades de la matriz de covarianza se encuentran las de ser simétrica y definida positiva. Para ver más propiedades y su demostración son consulte [3].

3. Sobre PCA: Procedimiento y detalles importantes

3.1. Procedimiento para realizar el análisis de componentes principales

Sean X, Y variables aleatorias. Recordemos que si X, Y son variables aleatorias, su covarianza se define como $\text{Cov}(X, Y) = E[(X - E[X])(Y - E[Y])]$. Note que $\text{Cov}(X, X) = \text{Var}(X, X)$.

Con el fin de generalizar los conceptos anteriores a dimensiones más grandes se introduce el concepto de *vector aleatorio*. Como se mencionó en la sección 2, un vector aleatorio X de dimensión n se define como un vector cuyas componentes individuales $X_1, X_2, X_3, \dots, X_n$ corresponden a variables aleatorias. De esta manera, los conceptos estadísticos anteriores se pueden generalizar a dichos vectores.

Sea $X = \begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix}$ un vector aleatorio. Si $E[X_i] < \infty$, para $i = 1, \dots, n$ recordemos que la esperanza de X como el vector $E[X] := \begin{pmatrix} E[X_1] \\ \vdots \\ E[X_n] \end{pmatrix}$.

Entonces de manera análoga al caso de una variable aleatoria, se define el operador de varianza sobre un vector aleatorio X como:

$$V(X) = E[(X - E[X])(X - E[X])^T] \quad (27)$$

Adicionalmente, para PCA un concepto muy importante es el de matriz de covarianza. La matriz de covarianza es una matriz cuadrada que contiene la covarianza entre los elementos de un vector, más específicamente, la matriz de covarianza de un vector aleatorio X se define como:

$$\Sigma = \begin{pmatrix} \text{Var}(X_1) & \text{Cov}(X_1, X_2) & \dots & \text{Cov}(X_1, X_n) \\ \text{Cov}(X_2, X_1) & \text{Var}(X_2) & \dots & \text{Cov}(X_2, X_n) \\ \vdots & \vdots & \dots & \vdots \\ \text{Cov}(X_n, X_1) & \text{Cov}(X_n, X_2) & \dots & \text{Var}(X_n) \end{pmatrix} \quad (28)$$

Proposición 3.1.

Sea $X = \begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix}$ un vector aleatorio tal que $E[X_i] < \infty$ para $i = 1, \dots, n$ y Σ su matriz de covarianza, entonces

$$\Sigma = V(X) \quad (29)$$

Prueba

Note que $X - E[X] = \begin{pmatrix} X_1 - E[X_1] \\ X_2 - E[X_2] \\ X_3 - E[X_3] \\ \vdots \\ X_n - E[X_n] \end{pmatrix}$, entonces

$$\begin{aligned}
(X - E[X])(X - E[X])^T &= \begin{pmatrix} X_1 - E[X_1] \\ X_2 - E[X_2] \\ X_3 - E[X_3] \\ \vdots \\ X_n - E[X_n] \end{pmatrix} \begin{pmatrix} X_1 - E[X_1] & X_2 - E[X_2] & X_3 - E[X_3] & \dots & X_n - E[X_n] \end{pmatrix} \\
&= \begin{pmatrix} (X_1 - E[X_1])^2 & (X_1 - E[X_1])(X_2 - E[X_2]) & \dots & (X_1 - E[X_1])(X_n - E[X_n]) \\ (X_2 - E[X_2])(X_1 - E[X_1]) & (X_2 - E[X_2])^2 & \dots & (X_2 - E[X_2])(X_n - E[X_n]) \\ \vdots & \vdots & \dots & \vdots \\ (X_n - E[X_n])(X_1 - E[X_1]) & (X_n - E[X_n])(X_2 - E[X_2]) & \dots & (X_n - E[X_n])^2 \end{pmatrix}
\end{aligned}$$

De esta manera,

$$E((X - E[X])(X - E[X])^T) \quad (32)$$

$$\begin{aligned}
&= \begin{pmatrix} E[(X_1 - E[X_1])^2] & E[(X_1 - E[X_1])(X_2 - E[X_2])] & \dots & E[(X_1 - E[X_1])(X_n - E[X_n])] \\ E[(X_2 - E[X_2])(X_1 - E[X_1])] & E[(X_2 - E[X_2])^2] & \dots & E[(X_2 - E[X_2])(X_n - E[X_n])] \\ \vdots & \vdots & \dots & \vdots \\ E[(X_n - E[X_n])(X_1 - E[X_1])] & E[(X_n - E[X_n])(X_2 - E[X_2])] & \dots & E[(X_n - E[X_n])^2] \end{pmatrix} \\
&= \begin{pmatrix} \text{Var}(X_1) & \text{Cov}(X_1, X_2) & \dots & \text{Cov}(X_1, X_n) \\ \text{Cov}(X_2, X_1) & \text{Var}(X_2) & \dots & \text{Cov}(X_2, X_n) \\ \vdots & \vdots & \dots & \vdots \\ \text{Cov}(X_n, X_1) & \text{Cov}(X_n, X_2) & \dots & \text{Var}(X_n) \end{pmatrix} [*] \quad (34)
\end{aligned}$$

Pero la expresión $[*]$ corresponde precisamente a la definición de matriz de covarianza, por lo que $V(X) := E[(X - E[X])(X - E[X])^T] = \Sigma$.

Entre algunas de las propiedades de la matriz de covarianza se encuentra que es definida positiva y simétrica (recuerde que $\text{Cov}(X_i, X_j) = \text{Cov}(X_j, X_i)$). Para una prueba detallada de estas propiedades y otras diríjase a [3]

Recordemos que el teorema espectral nos dice que una matriz cuadrada con entradas en los reales es simétrica si y sólo si es diagonalizable ortogonalmente, esto es, una matriz cuadrada $n \times n$, A es simétrica si y sólo si podemos encontrar una **matriz ortogonal** Q y una **matriz diagonal** D tales que

$$A = QDQ^T = QDQ^{-1} \quad (35)$$

Por lo tanto, como la matriz de covarianza es simétrica, también es diagonalizable ortogonalmente. Sea QDQ^{-1} su descomposición.

Las siguientes dos proposiciones serán relevantes en nuestro estudio de PCA:

Proposición 3.2.

Para la descomposición QDQ^T de la matriz de covarianza Σ anterior, si se reordenan los valores propios en D , por ejemplo, intercambiando λ_i con λ_j , y se hace exactamente el mismo

intercambio en las columnas correspondientes q_i y q_j de Q obteniendo nuevas matrices D' y Q' , se tiene que Q' será ortogonal y $\Sigma = Q'D(Q')^T$.

Prueba

Una matriz es ortogonal si y sólo si sus columnas forman un conjunto de vectores ortonormales. Intercambiar dos de sus columnas no altera esta ortonormalidad (el conjunto seguirá siendo el mismo), así que Q' será también ortogonal.

Por otro lado, hacer ese intercambio descrito que afecta tanto a Q como a D básicamente corresponde a hacer una permutación coordinada. En efecto, sea P la **matriz de permutación** que intercambia las columnas i y j de la matriz Q produciendo Q' . Note que por ser matriz de permutación es ortogonal, entonces $P^T = P^{-1}$. El intercambio de λ_i con λ_j corresponde a un intercambio de las filas i y j y luego de las columnas i y j ; en este orden, podemos expresar el intercambio de λ_i y λ_j como $P^T D P$. Luego, $D' = P^T D P$ y se tiene que

$$Q'D'(Q')^{-1} = (QP)(P^T D P)(P^T Q^T) = Q(P P^T) D (P P^T) Q^T = Q D Q^T = \Sigma \quad (36)$$

Proposición 3.3.

Definimos $M \in \mathbb{R}^{n \times n}$ como una matriz diagonal con entradas en $\{+1, -1\}$, es decir:

$$M = \text{diag}(d_1, d_2, \dots, d_n), d_i \in \{+1, -1\} \quad (37)$$

Sea

$$\hat{Q} = Q M \quad (38)$$

, donde Q es la matriz ortogonal de la descomposición de Σ dada, entonces:

- a) \hat{Q} sigue siendo ortogonal
- b) Puede elegirse M para que $\det(\hat{Q}) = 1$
- c) $Q D Q^T = \hat{Q} D \hat{Q}^T$

Prueba

a)

$$\hat{Q}^T \hat{Q} = M^T Q^T Q M = M^T M = I \quad (39)$$

b) Recuerde que $\det(\hat{Q}) = \det(Q)\det(M)$ por propiedades de determinante. Como Q es ortogonal su determinante es 1 o -1. Note que independientemente del valor del determinante de Q , como $\det(M) = \prod_{i=1}^n d_i = \pm 1$, siempre se puede ajustar un signo en M para forzar que el producto total sea 1.

c)

$$(QM)D(QM)^T = QMDM^T Q^T \quad (40)$$

Como M es diagonal con ± 1 en la diagonal, entonces

$$MDM^T = MDM = M \quad (41)$$

pues $(MDM)_{ij} = m_i d_i m_j = d_i \delta_{ij}$ (sigue siendo D). Entonces $(QM)D(QM)^T = QDQ^T$

Por las dos proposiciones anteriores podemos seleccionar una descomposición ortogonal para Σ , QDQ^T tal que las entradas de la matriz diagonal D serán los valores propios ordenados de la forma $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \geq 0$ y adicionalmente, podemos escoger una matriz Q que cumpla que $\det(Q) = 1$. Más adelante veremos que el ordenar los valores propios facilita el identificar las direcciones de mayor varianza en PCA, mientras tanto nos centraremos en la suposición de que $\det(Q) = 1$. Bajo esta hipótesis, el cambio de variables $\mathbf{x}' = Q^T \mathbf{x}$ corresponde a una **rotación de ejes**. Básicamente, al hacer $\mathbf{x}' = Q^T \mathbf{x}$ estamos expresando el vector \mathbf{x} en la base ortonormal (respecto al producto escalar euclídeo) formada por los vectores columna de Q , denotados como q_1, \dots, q_n . Es decir, proyectamos \mathbf{x} sobre cada *dirección principal* (este es el nombre que le daremos a los vectores q_i) haciendo $x_i' = \langle q_i, \mathbf{x} \rangle$. Así habremos cambiado el sistema de coordenadas.

3.2. Matriz de covarianza de un conjunto de datos

Para ir concretando las ideas de cómo funciona PCA sobre un conjunto de datos, suponga que tenemos m características o atributos que se desean medir para n trabajadores. Estas características asumen valores en los reales, entonces cada muestra \mathbf{x}_i (dada por una medición

de las m características para un trabajador i) será un vector en \mathbb{R}^m , esto es, $\mathbf{x} = \begin{pmatrix} x_{1i} \\ x_{2i} \\ \vdots \\ x_{mi} \end{pmatrix}$

donde cada x_{ji} corresponde al valor que se obtuvo para el trabajador i en la característica j . Denotaremos por X a la matriz $m \times n$ cuyas columnas serán los vectores \mathbf{x}_i para $i = 1, \dots, n$, es decir, $X = [\mathbf{x}_1 \mid \mathbf{x}_2 \mid \dots \mid \mathbf{x}_n]$ donde cada fila j estará formada por las mediciones de los trabajadores $1, 2, 3, \dots, n$ para la característica j . Ahora consideraremos la matriz

$$Y := [\mathbf{x}_1 - \bar{\mathbf{x}} \mid \mathbf{x}_2 - \bar{\mathbf{x}} \mid \dots \mid \mathbf{x}_n - \bar{\mathbf{x}}] \quad (42)$$

donde $\bar{\mathbf{x}} := \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$. Básicamente, aquí habremos centrado los datos (note que la suma de cada fila de Y será cero). Más adelante se explicará detalladamente el porqué es importante centrar los datos, pero por el momento, puede decirse a grandes rasgos que en un conjunto de datos no centrados, la matriz de covarianza también incluye las contribuciones de las medias, lo que podría confundir la **varianza** y el **sesgo**. En ese sentido, el centrado elimina el **sesgo sistemático** y garantiza que las componentes principales halladas con PCA (que se definirá a continuación) representen fielmente la variabilidad subyacente.

Definiremos ahora la matriz $S := \frac{1}{n} Y Y^T$. Note que esta matriz está dada por

$$\frac{1}{n} Y Y^T = \frac{1}{n} \begin{pmatrix} (x_{11} - \bar{x}) & (x_{12} - \bar{x}) & \dots & (x_{1n} - \bar{x}) \\ (x_{21} - \bar{x}) & (x_{22} - \bar{x}) & \dots & (x_{2n} - \bar{x}) \\ \vdots & \vdots & \dots & \vdots \\ (x_{m1} - \bar{x}) & (x_{m2} - \bar{x}) & \dots & (x_{mn} - \bar{x}) \end{pmatrix} \begin{pmatrix} (x_{11} - \bar{x}) & (x_{21} - \bar{x}) & \dots & (x_{m1} - \bar{x}) \\ (x_{12} - \bar{x}) & (x_{22} - \bar{x}) & \dots & (x_{m2} - \bar{x}) \\ \vdots & \vdots & \dots & \vdots \\ (x_{1n} - \bar{x}) & (x_{2n} - \bar{x}) & \dots & (x_{mn} - \bar{x}) \end{pmatrix} \quad (43)$$

$$= \frac{1}{n} \begin{pmatrix} \sum_{i=1}^n (x_{1i} - \bar{x})^2 & \sum_{i=1}^n (x_{1i} - \bar{x})(x_{2i} - \bar{x}) & \dots & \sum_{i=1}^n (x_{1i} - \bar{x})(x_{mi} - \bar{x}) \\ \sum_{i=1}^n (x_{2i} - \bar{x})(x_{1i} - \bar{x}) & \sum_{i=1}^n (x_{2i} - \bar{x})^2 & \dots & \sum_{i=1}^n (x_{2i} - \bar{x})(x_{mi} - \bar{x}) \\ \vdots & \vdots & \dots & \vdots \\ \sum_{i=1}^n (x_{mi} - \bar{x})(x_{1i} - \bar{x}) & \sum_{i=1}^n (x_{mi} - \bar{x})(x_{2i} - \bar{x}) & \dots & \sum_{i=1}^n (x_{mi} - \bar{x})^2 \end{pmatrix} \quad (44)$$

Pero recuerde que para dos variables aleatorias W, Z , su covarianza muestral está dada por $\text{Cov}(W, Z) = \frac{1}{n} \sum_{i=1}^n (w_i - \bar{w})(z_i - \bar{z})$, entonces cada entrada i, j de la matriz S corresponde a la covarianza muestral de los vectores que aparecen en las filas i, j de la matriz Y o más precisamente, de las variables aleatorias que se muestrearon para las características i y j . Esta matriz es la matriz covarianza que usaremos en PCA.

Por un momento dejaremos la matriz de covarianza a un lado y nos centraremos en Y . Para cada columna y_i de Y , se tiene que si $a := \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}$ es un vector en $\mathbb{R}^{n \times 1}$,

$$\sum_{j=1}^n a_j y_j = a^T Y \quad (45)$$

Nos interesa la varianza de cada $z_j = a^T y_j$, puesto que mide qué tan dispersos están los valores proyectados de nuestros datos sobre una dirección a en el espacio. a lo largo de esa dirección. Se puede decir, entonces que si la nube de puntos es muy larga en la dirección a , entonces las proyecciones z_i estarán muy separadas o en otras palabras, tendrán alta varianza y si la nube de puntos es muy estrecha en esa dirección las proyecciones estarán muy juntas, lo que significa baja varianza.

Consideramos entonces la varianza de $a^T Y$:

$$\text{Var}(a^T X) = \text{Var}\left((a^T Y - E[a^T Y])(a^T Y - E[a^T Y])^T\right) \quad (46)$$

Pero recordemos que la media de cada fila de Y es cero, luego, lo anterior es igual a

$$\text{Var}(z_i) = \frac{1}{n} \sum_{i=1}^n (z_1)^2 = \frac{1}{n} \|z\|^2 = \frac{1}{n} \|a^T Y\|^2 = \frac{1}{n} (a^T Y)(a^T Y)^T \quad (47)$$

$$= \frac{1}{n} a^T Y Y^T a = a^T \Sigma a \quad (48)$$

lo cual nos dice que si queremos maximizar $\text{Var}(a^T Y)$ necesitamos encontrar un vector m — dimensional que maximice $a^T \Sigma a$.

Se impone una restricción adicional de requerir que $a^T a = 1$ (es decir, el vector a tiene norma unitaria) para garantizar que estamos buscando la dirección óptima en el espacio de vectores sin estar influenciados por la magnitud del vector. Sin esta restricción, podría lograr una forma cuadrática más grande $a^T \Sigma a$ simplemente escalando a para que tenga una norma más grande, lo que no proporcionaría una visión significativa de la varianza de los datos a lo largo de las direcciones en el espacio de características.

De este modo, esto se convierte en un problema de optimización donde usaremos un **multiplicador de Lagrange** para maximizar $a^T Sa - \lambda(a^T a - 1)$.

El lagrangiano para el problema está dado por

$$L(a, \lambda) = a^T Sa - \lambda(a^T a - 1) \quad (49)$$

con las condiciones $\max_a a^T Sa, \quad a^T a = 1$.

Tenemos que calcular $\nabla_a L(a, \lambda)$ lo cual se hace usando las derivadas matriciales:

$$\nabla_a L(a, \lambda) = 2Sa - 2\lambda a \quad (50)$$

porque $\frac{\partial a^T Sa}{\partial a} = 2Sa$ dado que S es simétrica, $\frac{\partial a^T a}{\partial a} = \frac{\partial \|a\|^2}{\partial a} = 2a$ y $\frac{\partial(1)}{\partial(a)} = 0$.

Y si igualamos a cero

$$2Sa - 2\lambda a = 0 \rightarrow Sa = \lambda a \quad (51)$$

Lo que nos permite concluir que a debe ser un valor propio de S y λ su valor propio.

Los m vectores propios (a_1, a_2, \dots, a_m) que obtendremos serán el conjunto completo de vectores propios de S . La combinación lineal $a_k^T Y$ será lo que denominaremos *componente principal* y para la obtención de cada una se tiene que el ordenamiento de los valores propios de la matriz de covarianza juega un papel importante, pues consideramos los valores propios con el valor más alto, ya que un valor propio más alto significará que se captura más varianza. Esto se debe a que un valor λ más alto resultará en un vector de magnitud mayor a lo largo del vector propio, es decir, estamos recuperando la máxima información de la matriz de covarianza. Se utilizan los vectores propios correspondientes para esos valores propios.

Más específicamente, haremos la rotación $Z = Q^T Y$ donde Q es la matriz ortogonal con restricción $\det(Q) = 1$ de la diagonalización ortogonal de Σ . Podría usarse una matriz Q ortogonal con determinante -1 , puesto que esa inversión no afecta el subespacio generado y solo cambia la orientación como si de un espejo se tratase; sin embargo, aunque es equivalente estadísticamente, no se suele desear si uno quiere mantener consistencia geométrica. Por ejemplo, en aplicaciones como compresión, interpretación de componentes, o visualización, mantener la orientación es útil y evita confusión.

Note que si $Q = [q_1 \mid q_2 \mid q_3 \mid \dots \mid q_m]$, entonces cada fila de $Z = \begin{bmatrix} q_1^T \\ q_2^T \\ q_3^T \\ \vdots \\ q_m^T \end{bmatrix} [y_1 \mid y_2 \mid \dots \mid y_n] = \begin{bmatrix} q_1^T y_1 & q_1^T y_2 & \dots & q_1^T y_n \\ q_2^T y_1 & q_2^T y_2 & \dots & q_2^T y_n \\ \vdots & \vdots & \vdots & \vdots \\ q_m^T y_1 & q_m^T y_2 & \dots & q_m^T y_n \end{bmatrix}$. Cada fila z_k^T de Z es la combinación lineal:

$$z_k = q_k^T y_i = [q_k^T y_1, q_k^T y_2, \dots, q_k^T y_n] \quad (52)$$

donde cada $q_k^T y_i$ es la proyección del dato y_i sobre el eje q_k . Recordemos que los vectores q_k conforman una base ortonormal para \mathbb{R}^m , donde m es la cantidad de *features* o características

y que corresponden a vectores propios de Σ , por lo que las filas de Z serán las componentes principales.

Observe que los datos originales se pueden recuperar por medio de la matriz Z . En efecto, si z_i denota la i —ésima columna de la matriz Z , entonces para $i = 1, \dots, n$ tenemos que

$$x_i = y_i + \bar{x} = Qz_i + \bar{x} \quad (53)$$

Para la reducción de dimensionalidad, recordemo que λ_i mide la varianza de los datos en la dirección del vector propio q_i , es decir, si proyectamos todos los datos sobre la dirección q_i , la varianza de esas proyecciones será exactamente λ_i . En este sentido, si λ_i es pequeño (cercano a cero), habrá muy poca varianza en la dirección q_i (los datos casi no se dispersan hacia esa dirección) y más aún, si $\lambda_i = 0$, todos los datos estarán contenidos en un **hiperplano ortogonal** a q_i , o en otras palabras, no habrá información útil (ni variación) en esa dirección. En conclusión, un λ_i pequeño quiere decir que esa dirección no aporta significativamente a describir la forma de la nube de datos.

Suponga que los valores propios $\lambda^{k+1}, \lambda_{k+2}, \dots, \lambda_m$, son muy cercanos a cero, entonces la reducción de dimensionalidad se haría eliminando los vectorios propios asociados a esos valores propios de Q , de tal forma que $\bar{Z} = [q_1 | q_2 | \dots | q_k]^T Y$, es decir, solamente tomamos las primeras k componentes principales. Geométricamente, esto puede pensarse como la proyección de los datos rotados al subespacio generado por los vectores q_1, \dots, q_k . Básicamente, la información de los m atributos originales la estamos resumiendo en los k atributos dados por las filas de \bar{Z} . Finalmente, la eliminación de características redundantes nos permitirá analizar de una mejor manera los datos iniciales.

3.3. Detalles importantes de PCA

PCA es una técnica basada en álgebra lineal que transforma las variables originales en combinaciones lineales (componentes principales). Recordemos que, para poder hacer esto, necesita:
Calcular varianzas y covarianzas
Construir una matriz de correlación o covarianza
Aplicar descomposición en valores propios (eigen decomposition)

Pero estas operaciones sólo tienen sentido con datos numéricos. Sin embargo, en general, no es necesario que todas las variables numéricas se incluyan sin más. En efecto, existen algunas buenas prácticas al momento de seleccionar variables para PCA, entre las cuales se encuentran:

3.3.0.1. Eliminar columnas con varianza cercana a 0

Una variable con varianza cercana a cero significa que casi todos los datos tienen el mismo valor (por ejemplo, 99% de las filas tienen “5”). Por lo que, puede decirse que, estas variables no aportan variabilidad al conjunto de datos; además, como el PCA se basa en encontrar las direcciones de mayor varianza, una variable constante no influye en los componentes principales y finalmente, incluirlas puede introducir ruido innecesario o afectar la interpretación del resultado.

3.3.0.2. Filtrar variables numéricas que tengan sentido conjunto

Si se mezclan variables de naturaleza diferente (por ejemplo: datos financieros con características técnicas), los componentes pueden terminar siendo combinaciones arbitrarias de cosas

que no tienen relación lógica. Esto genera componentes difíciles de interpretar, lo cual limita su utilidad práctica. Así que es mejor aplicar PCA a grupos de variables que analicen aspectos similares del problema.

3.3.0.3. Asegurar que estén escaladas

El PCA se basa en varianza, y la varianza depende del orden de magnitud de la variable. Las variables con valores más grandes dominan la varianza total y, por tanto, influyen más en los componentes principales. Usar una técnica como StandardScaler (que pone media = 0 y desviación estándar = 1) es algo que se recomienda hacer para evitar este sesgo, y asegurar que todas las variables aporten de manera equilibrada al análisis.

3.3.0.4. Manejo de valores faltantes antes de aplicar PCA

Las operaciones que se realizan en PCA no pueden llevarse a cabo si hay valores nulos (NaN). Es por tal razón que los valores faltantes deben imputarse, usualmente con la media o mediana, o con algún método más sofisticado. Si no se hace, el PCA fallará directamente o se eliminarán automáticamente filas, lo cual puede distorsionar los resultados o reducir la muestra útil.

3.3.0.5. Consideraciones acerca de lo que es un valor propio “pequeño”

En análisis de componentes principales (PCA) y análisis factorial, un valor propio o eigenvalue representa la cantidad de varianza explicada por cada componente o factor. Tradicionalmente, según el criterio de Kaiser, se consideran relevantes solo aquellos factores con eigenvalues mayores a 1, ya que aportan más varianza que una variable original individual. Sin embargo, este criterio no es absoluto ni debe utilizarse como única regla para decidir la retención de factores.

Un eigenvalue se considera “pequeño” cuando es menor que 1, indicando que el factor o componente explica menos varianza que una variable original. Sin embargo, factores con eigenvalues ligeramente inferiores a 1 (por ejemplo, entre 0.95 y 0.99) pueden ser valiosos si cumplen ciertos criterios complementarios, como tener una carga factorial significativa, aportar una proporción relevante de la varianza total, ser consistentes con la teoría subyacente del estudio y ser apoyados por métodos como el análisis paralelo o la inspección del gráfico de sedimentación.

Por lo tanto, la decisión de retener factores con eigenvalues pequeños debe basarse en un análisis integral y no únicamente en un umbral fijo. Además, la retención de tales factores debe ser transparente y justificada, pues la inclusión de factores con eigenvalues bajos puede llevar a sobreextraer componentes o dificultar la interpretación.

Véase [4] para más información al respecto.

3.3.0.6. Escalado antes de PCA

Uno de los detalles cruciales a considerar antes de aplicar análisis de componentes principales es si se deben escalar o estandarizar los datos. La decisión de escalar afecta directamente el resultado del análisis, ya que modifica la matriz sobre la cual se realiza la descomposición espectral.

Cuando calculamos la matriz de covarianza Σ , estamos evaluando la variabilidad absoluta de los datos. Si una variable tiene una varianza mucho mayor que las demás, entonces su influencia numérica en la matriz será mucho mayor. Por ejemplo suponga que X_1 varía entre 1 y 10 y X_2 varía entre 1,000 y 10,000. Una regla empírica bastante usada cuando no tenemos todos los datos dice que $\sigma \approx \frac{\text{valor máximo} - \text{valor mínimo}}{4}$ (véase [5]). Así que $\sigma_1 \approx 3$ y $\sigma_2 \approx 3000$, entonces la matriz de covarianza Σ estará “dominada” por los valores grandes en la dirección de X_2 . Como consecuencia el primer componente principal (el de mayor varianza) apuntará casi en la dirección de X_2 . Vemos que esto ocurre no porque X_2 sea más relevante, sino porque su escala es mayor. Entonces al aplicar PCA sobre Σ , estaremos maximizando la varianza en unidades absolutas.

Por lo anterior, se recomienda estandarizar los datos, transformando cada variable para que tenga media cero y desviación estándar uno:

$$Y_{ij} = \frac{X_{ij} - \mu_j}{\sigma_j} \quad (54)$$

$$R = \text{cor}(X) = \left(\frac{1}{n}\right) \cdot Y^T Y \quad (55)$$

Aquí, cada columna de Y tiene media cero y varianza uno.

Esta matriz tiene 1's en la diagonal (porque cada variable estandarizada tiene varianza 1), y los coeficientes de correlación en las entradas fuera de la diagonal.

De este modo, al aplicar PCA sobre R , estaremos extrayendo componentes principales que maximizan la varianza relativa entre variables, sin que ninguna domine debido a su escala. Es decir, todas las variables contribuyen en igualdad de condiciones al cálculo.

Los autovalores λ_i de R nos dirán qué fracción de la varianza total estandarizada explican los componentes principales. Como la suma total de las varianzas estandarizadas es k (una por variable), se cumple que $\lambda_1 + \lambda_2 + \dots + \lambda_k = k$ y así, cada λ_i indicará la proporción de información estructural explicada por el i —ésimo componente principal, sin sesgo por escalas.

3.3.1. Distancia de Mahalanobis y PCA robusto

El Análisis de Componentes Principales (PCA) es una técnica de reducción de dimensionalidad que permite transformar un conjunto de variables posiblemente correlacionadas en un nuevo sistema ortogonal de ejes, llamados componentes principales, que capturan la mayor parte de la variabilidad presente en los datos. Sin embargo, su aplicación estándar que se basa en la matriz de covarianza empírica es altamente sensible a la presencia de outliers. En efecto, como se ve en [6], incluso una sola observación atípica puede distorsionar notablemente la orientación de los componentes, afectando tanto la interpretación como el desempeño de cualquier modelo que se entrene sobre estos nuevos ejes. Por ello, surge la necesidad de un enfoque más robusto frente a estas anomalías.

3.3.1.1. Detección de outliers y la necesidad de una medida multivariada

Antes de aplicar un PCA robusto, es necesario identificar y caracterizar las observaciones atípicas dentro del conjunto de datos. Para ello, se requiere una métrica de distancia que sea

apropiada en un contexto multivariado, es decir, que considere la distribución global de los datos, tome en cuenta la correlación entre las variables y sea invariante a la escala de las variables. Como primera opción y de manera intuitiva, es posible pensar en usar la distancia euclídea; sin embargo, no satisface ninguna de estas condiciones de forma adecuada. En efecto, observe que:

1. No considera la forma de la distribución: La distancia euclídea simplemente mide la “recta” entre dos puntos en el espacio, y esto no tiene en cuenta si los datos están más dispersos en una dirección que en otra. Lo anterior puede llevar a subestimar o sobreestimar distancias en presencia de variables con alta dispersión o estructuras elongadas.
2. Ignora la correlación entre variables: Suponga que dos variables están altamente correlacionadas. En este caso, una combinación de ambas no aporta mucha información nueva, pero la distancia euclídea las trata como si fueran independientes, duplicando artificialmente su peso. Esto puede llevar a errores en la detección de outliers, ya que los puntos que siguen esa correlación natural podrían parecer más “lejanos” de lo que realmente son.
3. Es sensible a la escala de las variables: Si una variable tiene valores que van de 0 a 1 y otra de 0 a 10000, la segunda dominará completamente el cálculo de distancia. A menos que se realice un escalado previo, la distancia euclídea no permite una comparación justa entre dimensiones. Incluso con escalado, sigue sin capturar correlaciones entre atributos.

Por esta razón, se introduce la distancia de Mahalanobis, una alternativa más adecuada, ya que ajusta el espacio de representación según la estructura estadística de los datos. Grosso modo, usa la matriz de covarianza para “transformar” el espacio, rotando y escalando los ejes de forma que todas las variables sean comparables y sus correlaciones sean tenidas en cuenta.

3.3.1.2. La distancia de Mahalanobis

La distancia de Mahalanobis, propuesta por el estadístico indio P. C. Mahalanobis en 1936, cuantifica qué tan lejos se encuentra un punto respecto al “centro” de una distribución multivariada, considerando su forma y dispersión. Formalmente, para un vector de características X , el vector μ de medias (una media por cada variable) que representa el centroide de la distribución multivariada y la matriz de covarianza se define como

$$D_{M(X)} = \sqrt{(X - \mu)^T \Sigma^{-1} (X - \mu)} \quad (56)$$

donde

- $X - \mu$ es un vector de desplazamiento respecto al centro
- Σ^{-1} permite escalar y rotar el espacio para que la dispersión en todas las direcciones quede “igualada”

El producto escalar resultante da una medida de “lejanía” normalizada según la geometría de la nube de datos. Finalmente, esta distancia es clave para detectar outliers: aquellos puntos con una distancia de Mahalanobis alta pueden ser considerados como anómalos en el contexto de todas las variables simultáneamente.

3.3.1.3. Ejemplo ilustrativo (basado en Wikipedia)

Para comprender intuitivamente la utilidad de la distancia de Mahalanobis frente a la distancia euclídea, se recomienda consultar el ejemplo clásico descrito en la entrada de Wikipedia sobre la distancia de Mahalanobis. En dicho ejemplo, se plantea el caso de un pescador que desea clasificar salmones según sus características físicas: longitud y anchura. Como estas variables tienen diferentes escalas y varianzas, aplicar directamente la distancia euclídea conduce a una clasificación sesgada, ya que otorga más peso a la variable con mayor variabilidad (la longitud). La distancia de Mahalanobis, al tener en cuenta la varianza y la correlación entre variables, permite realizar una comparación más justa y representativa entre observaciones, ponderando adecuadamente las diferencias en cada dimensión.

3.3.1.4. Consideraciones sobre la invertibilidad de la matriz de covarianza y su estimación robusta

El análisis de componentes principales (PCA) y la distancia de Mahalanobis requieren, en su formulación matemática, la inversión de la matriz de covarianza del conjunto de datos. No obstante, la inversa de la matriz de covarianza no siempre existe, y este detalle técnico tiene profundas implicaciones prácticas en el tratamiento de datos reales. Comencemos recordando que la matriz de covarianza es invertible si y sólo si es no singular, lo que significa que su determinante es distinto de cero o, equivalentemente, que tiene rango completo, es decir, todas sus columnas (o filas) son linealmente independientes. Esto no se cumple en los casos donde hay variables correlacionadas o redundantes, pues si una o más variables son combinaciones lineales exactas de otras, la matriz pierde rango (por ejemplo, si una columna es el doble de otra) o donde hay pocas muestras respecto al número de variables, ya que si el número de observaciones n es menor (o apenas comparable) al número de variables p , la matriz de covarianza será de rango deficiente. Este problema es común en contextos de alta dimensionalidad como genómica o visión por computador. A continuación, se describen algunas estrategias desarrolladas en la literatura para abordar este problema, agrupadas según los supuestos asumidos y las herramientas matemáticas utilizadas:

1. Regularización por Ridge o Shrinkage Estimators

Una estrategia directa consiste en regularizar la matriz de covarianza muestral sumando una constante positiva λ a su diagonal:

$$\Sigma_{\text{reg}} = \Sigma + \lambda I \quad (57)$$

Esto garantiza que la matriz resultante sea positiva definida y, por tanto, invertible, aún si la matriz original no lo era. Esta idea subyace en métodos ampliamente utilizados como ridge regression o los estimadores de shrinkage, y ha sido fundamental en aplicaciones como la optimización de portafolios financieros (Deng y Tsui, 2013).

2. Estimación Parcial mediante Decomposición de Cholesky Modificada (MCD)

En contextos como datos longitudinales, series temporales o señales espectrales, donde las variables tienen un orden natural, se puede explotar esta estructura mediante una decomposición de Cholesky modificada, como la propuesta en [7]. Este enfoque garantiza positividad definida y proporciona una interpretación en términos de regresión secuencial. En particular,

en [8] proponen un modelo de Block Cholesky Decomposition (BCD) que extiende este enfoque cuando se conoce parcialmente el orden de las variables. El BCD permite mantener la interpretabilidad y garantizar la validez estadística del estimador, incorporando además técnicas de regularización para fomentar la sparsity (es decir, matrices con muchos ceros), lo cual es fundamental para aplicaciones en modelos gráficos y selección de variables.

3. Selección de Variables y Reducción Dimensional

Otra estrategia clásica es eliminar variables redundantes o altamente correlacionadas antes de estimar la covarianza. Esto mejora el condicionamiento de la matriz y evita problemas de singularidad. Sin embargo, este enfoque puede resultar riesgoso si se eliminan variables informativas. En el contexto del presente proyecto, esta estrategia no fue utilizada, ya que se busca mantener la estructura completa de los datos y abordar directamente el problema de los outliers mediante estimación robusta.

4. Estimación Robusta: Minimum Covariance Determinant (MCD)

Cuando el problema no es la colinealidad per se, sino la presencia de outliers que distorsionan las estimaciones, la solución más adecuada es utilizar un estimador robusto como el Minimum Covariance Determinant (MCD), propuesto por [9]. El MCD busca el subconjunto de tamaño h que tenga la menor determinante de la covarianza, excluyendo los valores extremos. El algoritmo FAST-MCD implementado en scikit-learn permite su aplicación eficiente incluso en alta dimensionalidad. La matriz estimada mediante MCD es típicamente invertible, ya que se calcula sobre un subconjunto más “estable” de observaciones, excluyendo aquellas que causan problemas de colinealidad o singularidad. MCD no sólo permite calcular la inversa de la matriz de covarianza en presencia de outliers, sino que también proporciona la media robusta y las distancias de Mahalanobis robustas, las cuales son fundamentales para la detección de atípicos y el desarrollo de PCA robusto.

5. Modelos Basados en Optimización Convexa (Lasso y Programación Lineal)

En contextos de alta dimensión, algunos métodos recientes formulan la estimación de la inversa de la matriz de covarianza como un problema de optimización convexa. Por ejemplo, [10] explora la conexión entre la regresión multivariada y la matriz de precisión, proponiendo un método basado en programación lineal que explota la sparsity estructural en la matriz inversa. Este enfoque es especialmente útil en modelos gráficos gaussianos, donde la estructura de ceros en la inversa de la covarianza corresponde a relaciones de independencia condicional entre variables.

3.3.1.5. Estimación robusta con Minimum Covariance Determinant (MCD)

Estimación Robusta con Minimum Covariance Determinant (MCD) Entre las estrategias anteriormente mencionadas se hizo uso de la del estimador Minimum Covariance Determinant (MCD) y es que además de reducir significativamente la influencia de los outliers, proporcionar una matriz de covarianza invertible y bien condicionada, al evitar puntos que generan colinealidades artificiales y preservar la estructura real del conjunto de datos, facilitando técnicas de análisis posteriores como el PCA robusto y la distancia de Mahalanobis robusta, el MCD ofrece una serie de ventajas con respecto a otros métodos robustos que se detallarán más adelante.

3.3.1.5.0.1. Funcionamiento del MCD

El MCD se puede describir conceptualmente en los siguientes pasos:

1. Generación de subconjuntos candidatos: A partir del conjunto total de datos X , el algoritmo genera múltiples subconjuntos de tamaño h , donde h es típicamente el 75% del número total de muestras. Este tamaño es lo suficientemente grande para captar la estructura subyacente de los datos, pero lo bastante pequeño para excluir puntos extremos.
2. Evaluación de subconjuntos: Para cada subconjunto, se calcula la media muestral, la matriz de covarianza y el determinante de dicha matriz. El determinante de la matriz de covarianza actúa como una medida del volumen de dispersión de ese subconjunto: cuanto menor sea, más compacto y coherente se considera el grupo de observaciones.
3. Selección del subconjunto óptimo: Se selecciona el subconjunto cuyo determinante de la covarianza es el mínimo entre todos los candidatos. Este subconjunto es interpretado como el núcleo más representativo de los datos.
4. Estimación robusta: Finalmente, a partir de ese subconjunto óptimo, se calcula una media robusta que representa el centro de la distribución sin influencia de outliers y una matriz de covarianza robusta que refleja la dispersión real de los datos centrales.

3.3.1.5.0.2. Comparación con otros estimadores robustos

El MCD pertenece a una familia de métodos robustos de estimación de covarianza. A continuación, se comparan algunas de las principales alternativas:

- Minimum Volume Ellipsoid (MVE): Este método busca el elipsoide de menor volumen que contenga una fracción del conjunto de datos. Aunque conceptualmente similar al MCD, MVE es más sensible numéricamente y más costoso computacionalmente [11].
- S-estimadores: Basados en minimizar funciones de pérdida robustas, como la función bisquare de Tukey, estos métodos permiten una gran flexibilidad. Sin embargo, su rendimiento depende críticamente de la elección del parámetro de tuning que controla la sensibilidad al outlier. Esto introduce incertidumbre en el análisis si no se ajustan adecuadamente (véase [11]).
- Shrinkage estimators: Estos métodos combinan la matriz empírica con una matriz objetivo (por ejemplo, la identidad) mediante una interpolación. Aunque útiles para evitar problemas de singularidad, no son inherentemente robustos a outliers, ya que aún utilizan toda la muestra sin excluir valores atípicos [12].

Pese a la variedad de opciones, el MCD representa una opción sólida para este proyecto por las siguientes razones:

- Tiene un breakdown point, o proporción máxima de valores atípicos que el estimador puede tolerar antes de dar resultados arbitrarios, alto (hasta el 50%), lo que le confiere alta robustez ante outliers [13].
- Proporciona una estimación directa de la media y la covarianza robustas.
- Es compatible de manera natural con PCA robusto, ya que permite aplicar el análisis solo sobre el subconjunto central de datos [14].

- Su implementación en scikit-learn es estable, reproducible y computacionalmente eficiente gracias al algoritmo FAST-MCD.

- No requiere una elección delicada de parámetros como los S-estimadores, lo cual facilita su integración sin sobreajuste. Si bien existen estimadores alternativos con ventajas específicas en ciertos contextos (como menor varianza asintótica o mayor flexibilidad), el MCD ofrece un balance ideal entre robustez, simplicidad y aplicabilidad práctica, siendo especialmente adecuado para análisis multivariados en presencia de ruido y outliers como el de este proyecto.

En comparación, MCD ofrece un excelente balance entre robustez, interpretabilidad y aplicabilidad multivariada, por lo cual ha sido ampliamente adoptado como técnica estándar en detección de outliers y análisis exploratorio robusto. Sin embargo, pese a sus múltiples ventajas, el MCD presenta ciertas limitaciones:

- Costo computacional: El cálculo del MCD puede resultar exigente en términos de tiempo y memoria cuando se aplica a conjuntos de datos con gran número de observaciones o variables. Esto se debe a que implica buscar múltiples subconjuntos posibles y calcular determinantes de matrices de covarianza. Sin embargo, en este proyecto, el tamaño del dataset es moderado y se encuentra dentro de los rangos para los que el algoritmo FAST-MCD (véase la siguiente sección) de scikit-learn se desempeña eficientemente (Rousseeuw & Van Driessen, 1999).

- Elección del parámetro hhh: Este parámetro define cuántas observaciones se consideran para calcular la matriz de covarianza robusta (típicamente entre el 70% y 90% del total). Si se elige un h muy pequeño, se gana robustez pero se pierde eficiencia; si se elige muy grande, se incluyen más outliers. En este caso, se ha utilizado el valor por defecto recomendado por scikit-learn, que optimiza automáticamente el valor de h para balancear robustez y eficiencia según el tamaño de la muestra y la dimensión de los datos.

- Sensibilidad a múltiples clusters o densidades: En datasets con estructuras internas complejas, como varios grupos con distintas distribuciones o densidades, el MCD podría seleccionar un subconjunto representativo de solo uno de ellos, ignorando los demás. Sin embargo, esto no es un problema en este proyecto, ya que el conjunto de datos no presenta alta heterogeneidad en cuanto a estructuras internas o múltiples distribuciones separadas, el problema de clasificación abordado tiene una variable objetivo (target) balanceada, lo que garantiza que cada clase está representada adecuadamente en el conjunto de entrenamiento y prueba y además, la partición de los datos se realiza con `train_test_split(..., stratify=y)`, lo que preserva la proporción de clases en ambos conjuntos, reduciendo el riesgo de que una clase quede subrepresentada al aplicar la detección de outliers o el PCA robusto sobre el subconjunto “limpio”.

3.3.1.5.0.3. Implementación eficiente: FAST-MCD

Aunque el enfoque descrito en la sección sobre el funcionamiento del MCD es conceptualmente directo, la búsqueda exhaustiva de todos los subconjuntos posibles de tamaño h sería computacionalmente inviable, especialmente para grandes conjuntos de datos. Para abordar este problema, [9] propusieron el algoritmo FAST-MCD, una variante eficiente que reduce enormemente la carga computacional manteniendo una alta calidad de estimación. FAST-MCD no evalúa todos los subconjuntos posibles, sino que utiliza un procedimiento iterativo y aleatorizado basado en inicializaciones inteligentes, reestimaciones y pasos de afinamiento

(refinement steps) que convergen rápidamente hacia un subconjunto cercano al óptimo global. Gracias a esta estrategia, FAST-MCD puede escalar a conjuntos de datos de dimensión media (cientos de muestras y decenas de variables), y es el enfoque utilizado en scikit-learn mediante la clase `sklearn.covariance.MinCovDet`.

3.3.1.6. PCA Robusto

El análisis clásico de componentes principales (PCA) es una técnica estadística fundamental para la reducción de dimensionalidad, ampliamente utilizada por su capacidad para extraer las direcciones de mayor varianza en los datos. Sin embargo, como se ha venido hablando, su aplicación directa presenta una debilidad crítica: la sensibilidad extrema a valores atípicos (outliers). Esto se debe a que el PCA se basa en la estimación de la matriz de covarianza empírica, la cual puede ser distorsionada significativamente por un pequeño número de observaciones anómalas, lo que afecta tanto la orientación de las componentes principales como la calidad de la compresión y proyección obtenida. Para subsanar esta limitación y preservar la estructura inherente del conjunto de datos, se adoptó un enfoque robusto que integra la detección de outliers basada en la distancia de Mahalanobis con un procedimiento de escalado robusto y una aplicación posterior de PCA. Este flujo de trabajo combina técnicas estadísticas robustas con métodos de reducción de dimensionalidad, mejorando la estabilidad de las proyecciones en presencia de ruido extremo. En primer lugar, se aplica el estimador de covarianza robusta Minimum Covariance Determinant (MCD), implementado mediante la clase `MinCovDet` de scikit-learn, que calcula una media y una matriz de covarianza resistentes a la influencia de outliers. Este estimador opera restringiéndose a un subconjunto de las observaciones (típicamente el 75% menos atípico), buscando minimizar el determinante de la covarianza dentro de ese subconjunto, lo cual maximiza la concentración de datos y excluye puntos aberrantes.

Con esta covarianza robusta Σ_{robusta} y media robusta μ_{robusta} , se calcula la distancia de Mahalanobis para cada observación x_i

$$D_{M(x_i)} = \sqrt{(x_i - \mu_{\text{robusta}})^T \Sigma_{\text{robusta}}^{-1} (x_i - \mu_{\text{robusta}})} \quad (58)$$

Esta medida multivariante permite cuantificar qué tan alejada está una observación de la distribución central robusta. Se define un umbral de corte mediante el percentil 97.5% de la distribución χ^2 con k grados de libertad (donde k es la cantidad de variables numéricas). Aquellas observaciones con distancias por encima de este umbral son marcadas como outliers multivariantes. Una vez identificados los outliers, se procede a su eliminación para obtener un subconjunto “limpio” del conjunto de entrenamiento, con el objetivo de evitar que valores atípicos influyan en los pasos posteriores. Luego, se realiza un escalado robusto de las variables numéricas, centrando cada dimensión en su media robusta y normalizando por la desviación estándar derivada de la matriz de covarianza robusta (i.e., usando la raíz cuadrada de la diagonal de Σ_{robusta}):

$$X_{\text{esc}} = \frac{X - \mu_{\text{robusta}}}{\sqrt{\text{diag}(\Sigma_{\text{robusta}})}} \quad (59)$$

Este paso garantiza que las variables numéricas estén centradas y comparables en magnitud, sin depender de estadísticas sensibles a outliers como la media y desviación estándar empíricas.

Finalmente, se aplica el PCA clásico sobre los datos robustamente escalados, utilizando el subconjunto libre de outliers como conjunto de entrenamiento. A pesar de emplear el algoritmo estándar de PCA (basado en descomposición en valores singulares o SVD), el hecho de que los datos hayan sido previamente filtrados y transformados mediante estimaciones robustas permite que las componentes principales capturen las direcciones de mayor variación estructural del conjunto, sin estar sesgadas por ruido extremo. Este enfoque puede considerarse una forma híbrida de PCA robusto, donde la robustez no proviene de modificar internamente el algoritmo de PCA, sino de una fase de preprocesamiento estadísticamente robusta, que asegura que los supuestos del PCA clásico sean razonablemente satisfechos en la práctica.

4. PCA y análisis de malware

4.1. Modelos de predicción para analizar malware

Un malware (del inglés malicious software) es cualquier programa informático diseñado con la intención de dañar, interrumpir, infiltrarse o controlar sistemas informáticos, servidores o redes sin el consentimiento del usuario. Existen diversos tipos de malware que se clasifican según su comportamiento, objetivo o método de propagación (para más detalles, véase [15]).

El análisis de malware es el proceso de estudiar el comportamiento, las características y el código de software malicioso con el propósito de identificar infecciones, evaluar el alcance del daño, descubrir cómo se llevó a cabo la intrusión, determinar su origen, identificar las vulnerabilidades explotadas y prevenir futuras amenazas. En esencia, puede entenderse como el arte de disecar malware.

A lo largo del tiempo, se han desarrollado múltiples técnicas y metodologías que, combinadas, permiten realizar análisis de muestras de malware de manera sistemática y efectiva. Sin embargo, el panorama de amenazas evoluciona rápidamente. Los avances tecnológicos, el crecimiento exponencial del software y la creciente complejidad de los entornos computacionales han impulsado la aparición de nuevas variantes de malware. Algunas son versiones modificadas de amenazas antiguas que han sido adaptadas para evadir mecanismos de detección modernos; otras son completamente nuevas, diseñadas para explotar entornos actuales como servicios en la nube, dispositivos móviles o infraestructuras de inteligencia artificial.

Por ejemplo, hace algunos años bastaba con examinar las cadenas de texto embebidas en un ejecutable para detectar comportamientos maliciosos. Hoy en día, sin embargo, existen **malware polimórficos y metamórficos** que alteran su estructura internamente en cada ejecución, e incluso algunos que integran técnicas de aprendizaje automático para modificar dinámicamente su comportamiento en función del entorno en el que se ejecutan. Estas nuevas capacidades hacen que los enfoques tradicionales resulten insuficientes, y exigen la incorporación de técnicas avanzadas, como el análisis de comportamiento o el aprendizaje automático.

Básicamente, con el crecimiento exponencial del número de muestras de malware, especialmente a partir de la década de 2010, cuando se comenzaron a publicar artículos donde se hablaba sobre el uso de machine learning para hacer detección de malware, las técnicas tradicionales de detección —como las basadas en **firmas estáticas**— comenzaron a volverse ineficaces ante variantes cada vez más sofisticadas, como el malware polimórfico y metamórfico. Algunos artículos fundacionales pueden verse en las referencias [16], [17]. Y en ese sentido, fue a partir de 2010, cuando la industria (por ejemplo, AV-Test, Kaspersky y Symantec) reportó un crecimiento exponencial en el número de variantes de malware creadas automáticamente por herramientas de **obfuscation, packers y malware metamórfico** (vea [18]). Todo esto motivó el surgimiento de métodos basados en aprendizaje automático para automatizar y mejorar la detección.

A grandes rasgos, el proceso general para aplicar modelos predictivos en la detección de malware suele seguir las siguientes etapas:

1. Extracción de características (feature extraction): a partir de una muestra de malware o software benigno, se extraen características relevantes. Estas pueden ser estáticas (como

opcodes, llamadas al sistema, hashes, cadenas de texto, etc.) o dinámicas (como comportamientos observados en entornos **sandbox**).

2. Construcción de un dataset: cada muestra se representa como un vector de características. Se etiqueta con su clase correspondiente (por ejemplo, “malicioso” o “benigno”).
3. Preprocesamiento: se limpian y normalizan los datos. Aquí pueden aplicarse técnicas de selección o reducción de dimensionalidad.
4. Entrenamiento de modelos: En esta fase se entrenan **clasificadores supervisados** (como **Random Forests, SVM, Redes Neuronales**) que aprenden a distinguir entre software malicioso y benigno en función de sus características.
5. Evaluación y despliegue: los modelos se validan con métricas como **precisión, recall y AUC**, y luego se despliegan para analizar nuevas muestras automáticamente.

4.2. Aplicación de PCA en la detección de malware

En general, los vectores de características extraídos de las muestras de malware suelen tener cientos o miles de dimensiones. En este contexto, el análisis de componentes principales (PCA) ha sido un enfoque que ha ganado relevancia como técnica previa al entrenamiento del modelo, por varias razones entre las cuales se encuentra la reducción dimensionalidad que permite reducir las características a usar conservando la información más significativa (es decir, la que explica mayor varianza), lo que mejora la eficiencia computacional y reduce el riesgo de sobreajuste, además de potenciar el rendimiento de los clasificadores y facilitar la representación gráfica y el análisis exploratorio permitiendo explorar la separación entre clases y detectar patrones ocultos.

El uso de PCA en modelos de predicción de infección por malware ha sido respaldado por estudios como *Malware traffic classification using principal component analysis and artificial neural network for extreme surveillance* (2019) donde se propone un sistema robusto de clasificación de tráfico que utiliza PCA y Redes Neuronales Artificiales (RNA) para proporcionar una vigilancia extrema. Además, este método propuesto busca exponer diversos ciberataques basados en IA, su impacto actual y su futuro. La simulación se realiza mediante un agente autónomo de desarrollo propio que aprende por sí mismo. Los resultados experimentales confirmaron que los esquemas propuestos fueron eficientes para clasificar el tráfico de ataque con un 99% de precisión en comparación con los métodos de estado del arte.

En otro artículo titulado *Features Reduction with PCA Technique for Malware Detection Using Machine Learning* (2023) se usó PCA para reducción de características y luego se utilizaron cuatro clasificadores de aprendizaje automático (KNN, árbol de decisión Naïve Bayes y un bosque aleatorio) para la detección de malware. Los resultados demostraron que el mejor rendimiento provenía de la detección utilizando bosque aleatorio con una precisión promedio de 0,991688.

Existen más estudios que han evaluado la aplicabilidad de PCA en la detección de malware (vea [19], [20], [21]) y cuyos resultados han sido variados (algunos determinaron una mejora en el rendimiento de ciertos modelos, mientras que otros no vieron que se afectara la precisión de modelos de predicción de malware que usaban PCA).

4.3. Spectral clustering

El clustering espectral es una técnica de aprendizaje no supervisado que permite identificar estructuras latentes en los datos a partir de la información contenida en un grafo de similitud. A diferencia de métodos como k-means, que dependen fuertemente de la forma de los clústeres (por ejemplo, que sean esféricos o linealmente separables), el spectral clustering se basa en conceptos de álgebra lineal y teoría espectral de grafos para estudiar de manera holística la estructura y cohesión de los clústeres, lo que le permite detectar agrupamientos incluso con formas arbitrarias o no convexas (vea [22]).

En este trabajo se opta por estudiar esta técnica debido a que, a pesar de haber aplicado enfoques supervisados como PCA y robust PCA con estrategias de detección de outliers, no se observaron mejoras sustanciales en el rendimiento del modelo. Esto sugiere que los patrones relevantes podrían no estar bien capturados en la representación lineal inicial de los datos o podrían estar en una estructura subyacente más compleja. Ante esto, se propone un enfoque no supervisado con el objetivo de explorar la posibilidad de una estructura agrupada en el espacio de características, sin imponer restricciones fuertes sobre la forma de los posibles clústeres.

El clustering espectral es particularmente adecuado para este tipo de exploración, ya que no opera directamente sobre las coordenadas originales de los datos, sino sobre una representación espectral obtenida a partir del grafo de afinidad construido entre las muestras. Esta transformación espectral puede revelar relaciones no evidentes a través de los primeros vectores propios del Laplaciano del grafo, lo cual, según investigaciones previas como [23], facilita una separación más clara entre regiones densas (del espacio de datos).

Cabe resaltar que aunque existen múltiples variantes del algoritmo de clustering espectral, aquí se abordará una versión básica que, sin dejar de ser simple, contiene muchos de los principios fundamentales que explican por qué funciona este enfoque. Estudiar esta versión permitirá construir una comprensión sólida antes de avanzar hacia modelos más sofisticados.

4.3.1. Algunos conceptos previos

El paper [23], aunque introduce muy bien el clustering espectral, da por sentados varios conceptos fundamentales, especialmente de álgebra lineal y teoría de grafos. A continuación se explican los conceptos considerados clave antes de entrar al algoritmo.

4.3.1.1. Grafo de similitud

En el contexto del clustering espectral, un grafo de similitud es una representación matemática que permite modelar relaciones de cercanía o semejanza entre datos. Cada punto de datos se representa como un nodo (o vértice), y las relaciones de similitud entre ellos se codifican como aristas ponderadas entre los nodos. El peso de una arista indica cuán similares son los dos puntos conectados.

Formalmente, se define un grafo $G = (V, E)$ donde $V = \{x_1, \dots, x_n\}$ es el conjunto de nodos (cada nodo representa un dato), $E \subset V \times V$ es el conjunto de aristas y se asocia una matriz de pesos o matriz de similitud $W \in R^{n \times n}$, donde cada entrada W_{ij} representa la similitud entre los nodos x_i y x_j .

4.3.1.1.1. Construcción del grafo de similitud

Hay varias formas de construir esta matriz W , dependiendo del criterio usado para determinar qué nodos deben conectarse y con qué peso. Las tres más comunes son:

- Grafo completamente conectado (fully connected)

Todos los pares de puntos están conectados. El peso se define usando una función de similitud como la función RBF (Radial Basis Function) o Gaussiana:

$$W_{ij} = \exp\left(\frac{-\|x_i - x_j\|}{2\sigma^2}\right) \quad (60)$$

donde $\|x_i - x_j\|$ es la distancia euclidiana entre los puntos x_i y x_j y $\sigma > 0$ es un hiperparámetro que controla la escala de la similitud.

Este tipo de grafo tiende a ser denso y costoso computacionalmente, pero útil cuando se quiere capturar toda la estructura global de los datos.

- k –Nearest Neighbors (k-NN) graph

Cada nodo se conecta con sus k vecinos más cercanos (en términos de distancia, típicamente euclidiana). Existen dos variantes comunes:

- **No simétrico:** x_i se conecta con x_j si x_k está entre los k vecinos más cercanos de x_i
- **Simétrico (mutual k-NN):** Se conectan si x_i está entre los k vecinos más cercanos de x_j y viceversa. El peso W_{ij} puede ser binario (1 si están conectados, 0 si no), o puede utilizarse también una función de similitud como la Gaussiana.
- ε -neighborhood graph

Aquí se conecta un par de nodos x_i y x_j si la distancia entre ellos es menor que un umbral ε . Este grafo es más sensible a la densidad de los datos y puede producir grafos desconectados si ε es muy pequeño (algo cuyas repercusiones se tratarán más adelante).

Como propiedades destacables de la matriz de similitud se tiene que usualmente es simétrica y no negativa. Además, $W_{ii} = 0$, en la mayoría de los casos, es decir, no se conecta un nodo consigo mismo.

En general, la estructura del grafo influye fuertemente en los resultados del clustering espectral, ya que determina la forma de la matriz Laplaciana usada en el algoritmo.

4.3.1.2. Matriz Laplaciana

La matriz laplaciana es una representación algebraica de un grafo que captura información sobre su estructura y conectividad. Es una herramienta central en clustering espectral porque permite estudiar las propiedades del grafo a través del análisis de sus autovalores y autovectores.

4.3.1.2.1. Definición

Hay distintas variantes de la matriz laplaciana, pero la más básica es la matriz laplaciana no normalizada, definida como:

$$L = D - W \quad (61)$$

donde W es la matriz de adyacencia o de similitud y D es la matriz de grados, una matriz diagonal donde cada elemento diagonal D_{ii} es la suma de las similitudes de i con todos los demás nodos:

$$D_{ii} = \sum_j W_{ij}$$

entonces, el elemento L_{ij} de la matriz laplaciana representa:

- $L_{ii} = D_{ii}$: El grado del nodo i
- $L_{ij} = -W_{ij}$ si $i \neq j$: La similitud negativa entre i y j
- $L_{ij} = 0$ si no hay conexión entre i y j

4.3.1.2.2. Otras variantes

Para aplicaciones como clustering espectral, es común trabajar con versiones normalizadas de la matriz laplaciana, que corrigen la influencia del grado de los nodos:

- Laplaciana simétrica normalizada:

$$L_{\text{sym}} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \quad (62)$$

- Laplaciana aleatoria:

$$L_{\text{rw}} = D^{-1} L = I - D^{-1} W \quad (63)$$

Ambas versiones tienen propiedades útiles para el análisis espectral, y su elección depende del algoritmo

Cabe resaltar que la matriz laplaciana es semidefinida positiva (todos sus autovalores son reales y no negativos) y su menor autovalor es 0, y el número de ceros (multiplicidad del 0 como autovalor) corresponde al número de componentes conexas del grafo.

4.3.2. Algoritmo

El fundamento del clustering espectral está en la representación de los datos como un grafo de similitud, donde cada nodo representa un dato y las aristas reflejan qué tan similares son entre sí. A partir de esta estructura, el algoritmo utiliza la matriz Laplaciana o alguna de sus variantes normalizadas— para obtener una representación embebida de los datos en un espacio de menor dimensión donde es más sencillo agruparlos.

4.3.2.1. Pasos generales del algoritmo

Given a set of points $S = \{s_1, \dots, s_n\}$ in \mathbb{R}^d that we want to cluster into k subsets:

1. Form the affinity matrix $A \in \mathbb{R}^{n \times n}$ defined by $A_{ij} = \exp(-\|s_i - s_j\|^2 / 2\sigma^2)$ if $i \neq j$, and $A_{ii} = 0$.
2. Define D to be the diagonal matrix whose (i, i) -element is the sum of A 's i -th row, and construct the matrix $L = D^{-1/2} A D^{-1/2}$.
3. Find x_1, x_2, \dots, x_k , the k largest eigenvectors of L (chosen to be orthogonal to each other in the case of repeated eigenvalues), and form the matrix $X = [x_1 x_2 \dots x_k] \in \mathbb{R}^{n \times k}$ by stacking the eigenvectors in columns.
4. Form the matrix Y from X by renormalizing each of X 's rows to have unit length (i.e. $Y_{ij} = X_{ij} / (\sum_j X_{ij}^2)^{1/2}$).
5. Treating each row of Y as a point in \mathbb{R}^k , cluster them into k clusters via K-means or any other algorithm (that attempts to minimize distortion).
6. Finally, assign the original point s_i to cluster j if and only if row i of the matrix Y was assigned to cluster j .

Para entender el porqué este algoritmo funciona, se nos propone imaginar el “escenario ideal”, que en el contexto del clustering espectral es uno en el que los datos están organizados en componentes completamente desconectadas dentro del grafo de similitud. Es decir, los datos de diferentes grupos no tienen ninguna conexión entre sí: no existe ninguna arista que una un punto de un grupo con un punto de otro. Matemáticamente, esto significa que el grafo está compuesto por k componentes conexas independientes.

En este caso, la matriz Laplaciana L (o su versión normalizada) tiene una propiedad útil: su multiplicidad para el valor propio cero es igual al número de componentes conexas. Entonces los vectores propios asociados a estos ceros pueden usarse para reconstruir una codificación exacta de los clusters, ya que cada uno de ellos está asociado exclusivamente a una componente del grafo.

Así, en este escenario ideal, el clustering espectral actúa casi como un clasificador perfecto, es decir, puede identificar sin ambigüedad los grupos de datos.

Sin embargo, en la práctica raramente se cuenta con datos tan perfectamente agrupados. Más bien, el grafo de similitud suele estar completamente conectado, aunque los pesos de las aristas que conectan puntos de diferentes grupos tienden a ser bajos. Es decir, aunque hay conexión entre todos los nodos, los grupos están más densamente conectados internamente que con el resto del grafo.

Esto hace que la matriz Laplaciana ya no tenga valores propios exactamente iguales a cero, sino que los valores propios más pequeños estén cercanos a cero. Aun así, la intuición espectral sigue siendo útil: los primeros k vectores propios tienden a contener información suficiente para distinguir entre grupos de datos densamente conectados.

Aquí, el clustering espectral funciona como una relajación del **problema de corte discreto** (como el RatioCut o el Normalized Cut), permitiendo identificar agrupaciones que, aunque no completamente desconectadas, minimizan las conexiones entre grupos y maximizan la

cohesión interna. El uso de los vectores propios proporciona una forma de “proyectar” los datos en un espacio donde esas divisiones se vuelven más evidentes.

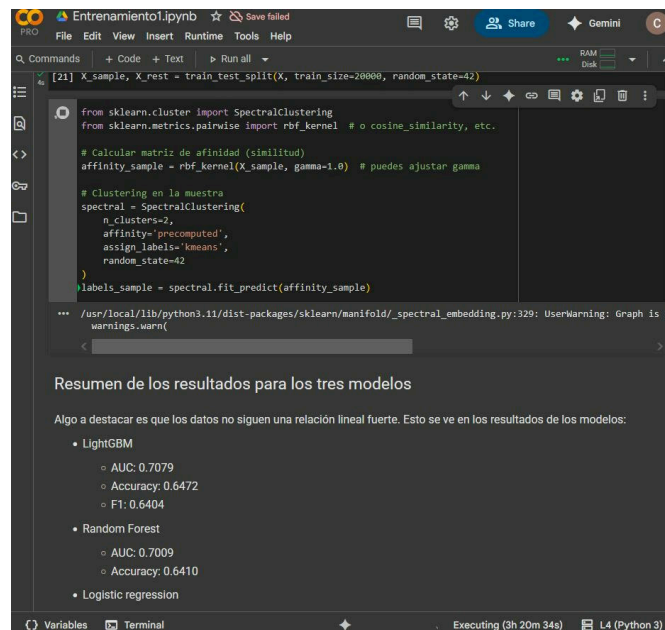
4.3.2.2. Problemas de implementación

A pesar de que el spectral clustering es altamente efectivo para identificar agrupamientos no lineales y estructuras complejas en los datos —precisamente porque no asume que los clústeres sean convexos—, este enfoque tiene una desventaja crítica cuando se aplica a conjuntos de datos a gran escala: su alto costo computacional. Esto se debe a que, en su forma básica, el algoritmo requiere construir matrices de afinidad y Laplacianas de tamaño completo, lo cual implica una complejidad temporal de $O(N^3)$ y espacial de $O(N^2)$ para un conjunto de datos con N objetos. Como resultado, el método se vuelve impráctico cuando se trabaja con volúmenes masivos de datos.

Para abordar esta limitación, en [24] se propusieron dos variantes diseñadas con escalabilidad en mente: U-SPEC (Ultra-Scalable Spectral Clustering) y su versión de ensamble, U-SENC (Ultra-Scalable Ensemble Clustering). Estas metodologías conservan muchas de las fortalezas teóricas del spectral clustering tradicional, pero incorporan mecanismos que permiten reducir drásticamente el consumo de tiempo y memoria.

En lo que sigue, explicaremos cómo funciona U-SPEC, mostrando las estrategias clave que permiten que el algoritmo mantenga la precisión sin incurrir en los costos computacionales del enfoque clásico. Luego introduciremos U-SENC, que extiende la idea de U-SPEC mediante un ensamble de múltiples soluciones base para mejorar aún más la robustez y estabilidad del resultado final.

Para ver el artículo original donde se habla de estos algoritmos vaya a [25].



```
[21] X_sample, X_rest = train_test_split(X, train_size=20000, random_state=42)

from sklearn.cluster import SpectralClustering
from sklearn.metrics.pairwise import rbf_kernel # o cosine_similarity, etc.

# Calcular matriz de afinidad (similitud)
affinity_sample = rbf_kernel(X_sample, gamma=1.0) # puedes ajustar gamma

# Clustering en la muestra
spectral = SpectralClustering(
    n_clusters=2,
    affinity="precomputed",
    assign_labels="kmeans",
    random_state=42
)

labels_sample = spectral.fit_predict(affinity_sample)

... /usr/local/lib/python3.11/dist-packages/sklearn/manifold/_spectral_embedding.py:329: UserWarning: Graph is not
warnings.warn(
```

Resumen de los resultados para los tres modelos

Algo a destacar es que los datos no siguen una relación lineal fuerte. Esto se ve en los resultados de los modelos:

- LightGBM
 - AUC: 0.7079
 - Accuracy: 0.6472
 - F1: 0.6404
- Random Forest
 - AUC: 0.7009
 - Accuracy: 0.6410
- Logistic regression

Figure 1: La implementación de SpectralClustering en scikit-learn (véase [26]) no sigue estrictamente el algoritmo propuesto por [23], sino que se alinea más con enfoques posteriores como los presentados por [27] y [28], enfocados en la segmentación multiclase y estrategias eficientes de partición espectral. Aunque todas estas propuestas comparten la idea central de usar los vectores propios del Laplaciano normalizado del grafo de similitud, scikit-learn no aplica la normalización fila a fila del embedding espectral antes de la asignación por K-Means, como lo hace el algoritmo original de Ng et al. En cambio, permite al usuario seleccionar entre distintas estrategias de asignación de etiquetas mediante el parámetro `assign_labels`:

- “kmeans” es el enfoque por defecto y consiste en aplicar el algoritmo de K-Means sobre los vectores propios obtenidos del Laplaciano; esta estrategia es más flexible y puede capturar particiones más finas, pero depende de inicializaciones aleatorias, por lo que puede producir resultados distintos en ejecuciones sucesivas si no se fija el `random_state`.
- “discretize” implementa una técnica basada en rotación ortogonal seguida de asignación de etiquetas por máximos, lo que genera particiones más simétricas y geoméricamente uniformes; además, es totalmente determinista.
- “cluster_qr” es una opción más reciente que aplica una factorización QR con pivoteo sobre el embedding espectral, permitiendo una asignación de etiquetas también determinista pero con mejor fidelidad geométrica que discretize, lo que resulta en particiones visualmente más coherentes en muchos casos prácticos.

Se intentó ejecutar esta técnica sobre un conjunto de datos de 8.921.483 registros, pero debido a la complejidad cúbica del cálculo espectral, la operación resultó computacionalmente inviable: después de más de cuatro horas de ejecución, el proceso no finalizó en múltiples pruebas. Además, se emitieron advertencias como “Graph is not fully connected, spectral embedding may not work as expected”, lo cual indica que la matriz de similitud generó un grafo no conexo, comprometiendo la validez del embedding espectral y, por ende, la calidad del agrupamiento.

4.3.2.3. Descripción del algoritmo U-SPEC (Ultra-Scalable Spectral Clustering)

U-SPEC es un algoritmo diseñado para aplicar clustering espectral en datasets de gran escala, abordando los principales cuellos de botella computacionales de los métodos clásicos. Esto lo logra mediante técnicas que reducen la dimensionalidad operativa y el tamaño del grafo involucrado, manteniendo al mismo tiempo una alta calidad en los resultados.

En primer lugar, dado que trabajar con todos los datos en clustering espectral clásico es inviable a gran escala, U-SPEC introduce una forma eficiente de elegir un conjunto reducido de representantes. Esta selección se realiza combinando selección aleatoria que es muy rápida, pero con riesgo de baja calidad y K-means sobre una muestra aleatoria, lo cual mejora la representatividad sin incurrir en el alto costo de aplicar K-means a todo el dataset.

Concretamente, se toma una muestra aleatoria de tamaño p' (usualmente 10 veces p), y sobre ella se aplica K-means para obtener p representantes finales. Así se equilibra velocidad y calidad.

Luego, para cada dato del dataset original, U-SPEC necesita estimar sus K representantes más cercanos, pero sin construir una matriz de afinidad completa de tamaño $N \times p$, lo cual sería inviable. Por eso, se propone el siguiente enfoque aproximado de dos etapas:

1. **Preprocesamiento:** Se agrupan representantes en clústeres locales mediante K-means (esta cantidad de clústeres es mucho menor que p) y luego se calculan los K' vecinos más cercanos para cada representante, donde K' es típicamente 10 veces K .
2. **Búsqueda aproximada:** Para cada objeto:
 - Se identifica el clúster de representantes más cercano.
 - Se busca el representante más próximo dentro de ese clúster.
 - A partir de él y sus K' vecinos, se seleccionan los K representantes más cercanos al objeto.

Con los K representantes más cercanos obtenidos para cada objeto, se puede construir la submatriz de afinidad dispersa $B = \{b_{ij}\}$ como se muestra a continuación. En [24], es mencionado que se utiliza el kernel gaussiano como kernel de similitud. Esta matriz B tiene en cada fila solo k elementos no nulos.

$$b_{ij} = \begin{cases} \exp\left(-\frac{\|x_i - r_j\|^2}{2\sigma^2}\right), & \text{if } r_j \in N_K(x_i) \\ 0, & \text{otherwise} \end{cases}$$

Figure 2: Definición matriz de afinidad dispersa B. Aquí r_j es uno de los K representantes más cercanos

Básicamente, U-SPEC representa la relación entre objetos y representantes mediante un grafo bipartito $G = \{X, R, B\}$, donde X es el conjunto de datos, R los representantes, y B la matriz de afinidad cruzada. Llegado este punto, en vez de resolver directamente el problema de autovalores sobre este grafo completo (de tamaño $N + p$), que sería muy costoso, se utiliza

una técnica llamada transfer cut. Esta permite reducir el problema a un grafo más pequeño que solo incluye los representantes (G_r) y resolver el problema espectral allí ($L_r v = \lambda D_r v$), lo que es mucho más eficiente.

A continuación, se reconstruyen las representaciones para los datos originales (u_x) a partir de los autovectores de los representantes (v), evitando así calcular autovectores para millones de puntos.

Una vez obtenidos los k autovectores relevantes para los datos originales, estos se usan como nuevas características. Se aplica K -means sobre ellas para obtener las etiquetas finales de clustering. El número de clústeres k se define a partir del problema o se asume conocido.

La complejidad computacional del algoritmo USPEC es $O(N \cdot (\sqrt{p \cdot d} + K \cdot d + K \cdot k + K^2 + k^2 \cdot t))$, donde N es el número total de datos (ej. puntos, objetos), p es el número de representantes seleccionados, d es la dimensión de los datos (número de características por punto), K es el número de representantes vecinos considerados para cada punto, k es el número de clústers deseados y t es el número de iteraciones del algoritmo k -means.

4.3.2.4. Consideraciones sobre los parámetros de USPEC

Los siguientes parámetros son clave en el comportamiento y eficiencia de USPEC. A continuación, se explican sus efectos y se ofrecen recomendaciones basadas en experimentos y análisis hechos al algoritmo:

- **p (Cantidad de representantes):** Número de puntos seleccionados como representantes del conjunto de datos.

Un p muy pequeño puede llevar a una representación pobre del espacio de datos, lo que afecta negativamente la calidad del clustering.

Un p muy grande mejora la calidad pero incrementa significativamente el costo computacional, especialmente al calcular distancias y construir la matriz bipartita.

► Recomendación:

- Para datasets grandes como el aquí estudiado (casi nueve millones de registros), valores entre 500 y 1000 suelen ser un buen punto de partida.
- Con $p = 500$, ya se está en un rango razonable para pruebas rápidas; si el tiempo de cómputo lo permite, puede probarse también con $p = 1000$.

- **Knn (Número de vecinos por dato):** Número de representantes vecinos a considerar para cada dato al construir la matriz de afinidad.

Valores pequeños (ej. 3–5) reducen el costo y generan grafos más dispersos, mientras que valores grandes mejoran la conectividad del grafo pero aumentan memoria y tiempo de cómputo.

► Recomendación:

- Para un submuestreo del 10%, un $Knn = 5$ es adecuado. Si se aumenta la muestra o el número de representantes, podrías subir a 7–10 para mejorar calidad.

- **maxTcutKmIters y cntTcutKmReps:** maxTcutKmIters es el número máximo de iteraciones para KMeans en el corte espectral. Por otro lado, cntTcutKmReps es la cantidad de repeticiones de KMeans con inicializaciones distintas.

Valores más altos aumentan la probabilidad de converger a una buena solución pero hacen el proceso más lento.

► Recomendación:

- Para el dataset aquí trabajado con submuestras con $p = 500$, use maxTcutKmIters = 100 y cntTcutKmReps = 5 como valores equilibrados.

- **sample_fraction (Fracción de datos por submuestra):** Porción del dataset completo sobre la que se aplica USPEC.

Reduce la carga computacional al evitar procesar todo el dataset. Además, puede introducir variabilidad, especialmente si la fracción es muy baja y no representa bien la estructura de clases.

► Recomendación:

- Con 8.9 millones de filas, un sample_fraction de 0.01 a 0.1 (1% a 10%) es lo usual para pruebas razonables.
- Para análisis más confiables, podría tratar de subir ese valor

- **num_iter (Número de repeticiones con submuestras):** Número de veces que se aplica el algoritmo sobre distintas submuestras para evaluar la robustez de los resultados.

Aumenta la estabilidad estadística del resultado (promedio de NMI, desviación).

► Recomendación:

- num_iter = 10 está bien. Si los resultados son muy variables, considere aumentar a 20.

4.3.2.5. Descripción del algoritmo Ultra-Scalable Ensemble Clustering (U-SENC)

Ultra-Scalable Ensemble Clustering (U-SENC) es una extensión del enfoque U-SPEC que busca mejorar la robustez del proceso de agrupamiento al combinar múltiples resultados de clustering en una única solución más estable y precisa. Esta técnica se apoya en los principios del ensemble clustering, el cual, según el trabajo citado, se compone de dos etapas principales: la generación del conjunto de agrupamientos base y la construcción de una función de consenso que sintetiza estos resultados en una partición final refinada.

En la fase de generación del conjunto, se emplean múltiples ejecuciones independientes de U-SPEC como agrupamientos base. Esto significa que se generan m instancias de U-SPEC (siendo m típicamente igual a 20), cada una actuando como una agrupación preliminar distinta. Para asegurar la diversidad entre estas instancias —lo cual es fundamental para obtener una buena combinación final— se introducen dos fuentes de variación: primero, los representantes utilizados en cada instancia se seleccionan de forma independiente a través de la estrategia híbrida de U-SPEC, la cual incluye una preselección aleatoria seguida de una agrupación con K-means, ambas no deterministas; segundo, el número de clústers k para cada instancia se elige de manera aleatoria dentro de un rango definido por valores mínimos y máximos, a través de

una fórmula que incorpora una variable aleatoria continua. Esto garantiza que cada $U-SPEC_i$ agrupe los datos en un número distinto de grupos, aumentando así la diversidad del conjunto total.

Una vez generado este conjunto de m particiones, se procede a la fase de consenso mediante la construcción de un grafo bipartito. En este grafo, los nodos representan tanto a los datos originales como a los clústers individuales producidos por las m instancias de U-SPEC. Se forma una matriz de afinidad cruzada que conecta cada objeto con los m clústers a los que pertenece (uno por cada instancia), dando lugar a una representación dispersa que requiere solo $O(Nm)$ memoria. Esta matriz permite transformar el problema de encontrar una agrupación final en la resolución de un problema espectral sobre un grafo reducido, que contiene únicamente los clústers como nodos. A través del análisis de los vectores propios de este grafo reducido, se obtienen nuevas representaciones para cada objeto, sobre las cuales se aplica K -means una última vez para definir la agrupación final.

Desde el punto de vista computacional, el costo total de U-SENC se puede dividir en dos partes: la generación de las m instancias de U-SPEC y la función de consenso. Bajo las condiciones típicas en las que m , k y K son mucho menores que p y p es mucho menor que N , el costo temporal de la fase de generación es del orden de $O(Nm\sqrt{p \cdot d})$, siendo este el término dominante. La función de consenso, por su parte, tiene una complejidad de $O(N(m^2 + mk + k^2 \cdot t) + kC^3)$, aunque este costo suele ser menor en comparación. En cuanto al uso de memoria, la fase de generación necesita aproximadamente $O(N\sqrt{p})$, mientras que la fase de consenso requiere $O(Nm)$, lo cual es razonable incluso en conjuntos de datos a gran escala.

Finalmente, un aspecto que queda sin resolver en el diseño de U-SENC es la determinación del número de clústers k cuando este no se conoce previamente. Aunque U-SPEC y U-SENC permiten trabajar con valores variables de k en las instancias base (dentro de un rango definido por K_{\min} y k_{\max}), el algoritmo no propone una solución para estimar automáticamente el número óptimo de clústers. Esta limitación representa un desafío importante cuando se busca aplicar estos métodos a conjuntos de datos masivos y sin información previa.

4.3.2.5.1. Parámetros del algoritmo USENC(fea, k, m, distance, p=1000, Knn=5, lowK=2, highK=5)

- fea: matriz de características con forma (N, d) . Son los datos de entrada.
- k: número total de clústers que se desea al final. Se usa en la etapa final de consenso. Si se conoce el número real de clases, se debe usar aquí.
- m: número de clusterings base (tamaño del ensamble). Se hacen m ejecuciones de U-SPEC. Se recomienda un valor de 20 para buena robustez, en caso de ser posible (un valor de 20 puede ser costoso computacionalmente).
- distance: tipo de distancia para U-SPEC. Comúnmente se usa 'euclidean'.
- p: cantidad de representantes en U-SPEC. Se recomienda que $p \ll N$, por ejemplo 50, 100, 500, 1000, en el casos del dataset aquí tratado.
- Knn: número de vecinos más cercanos por punto. Es usado para construir la matriz de afinidad. Valor recomendado: 5.
- lowK y highK: Rango para el número de clústers en cada instancia U-SPEC. Se selecciona un k_i aleatorio para cada instancia base. Ejemplo típico: lowK = 5, highK = 10.

4.3.3. Estudio de la estructura de los datos usando USPEC y USENC

En este estudio se implementaron y evaluaron dos algoritmos de clustering espectral escalables: U-SPEC (Ultra-Scalable Spectral Clustering) y U-SENC (Ultra-Scalable Ensemble Clustering), con base en la implementación disponible públicamente en [29]. Ambos métodos fueron probados sobre un conjunto de datos al que se le aplicaron distintos tratamientos respecto a los valores nulos: imputación y eliminación directa.

La experimentación incluyó, además, ejecuciones adicionales sobre submuestras aleatorias del 10% y 50% de los datos, con el objetivo de facilitar la elección de hiperparámetros adecuados y observar el comportamiento de los algoritmos en escenarios de reducción de tamaño. A pesar de estas variaciones, los resultados no evidenciaron diferencias significativas en las métricas obtenidas (NMI promedio). Tanto en las versiones con imputación como eliminación, y tanto con el conjunto completo como con las submuestras, los valores del NMI promedio se mantuvieron en rangos muy bajos, cercanos a cero.

En particular, la comparación directa entre USPEC y USENC mostró que USENC consistentemente obtuvo valores ligeramente superiores de NMI promedio en todos los casos evaluados, aunque dichas diferencias siguen siendo marginales y no permiten afirmar una buena recuperación de estructura por parte de ningún algoritmo. También se observó que reducir el tamaño del conjunto de datos (hasta un 10% de los registros) no alteró sustancialmente los resultados, lo que sugiere que el comportamiento de los algoritmos es consistente ante reducciones de tamaño, pero también que no están capturando ninguna estructura latente clara en los datos.

4.4. Implementación: Microsoft Malware Prediction

4.4.0.1. Primer entregable

Dada la magnitud y complejidad del conjunto de datos original, cuya carga completa excede la capacidad de procesamiento eficiente en memoria, se optó por un enfoque por muestreo sistemático para realizar el análisis exploratorio inicial. Este enfoque no solo mitigó las restricciones computacionales, sino que también permitió obtener una perspectiva más representativa y estructurada del comportamiento general de los datos.

Se definió una estrategia de análisis basada en la extracción de cinco muestras aleatorias de 500 filas cada una. Las muestras fueron seleccionadas de manera aleatoria a partir del conjunto de datos ya filtrado, evitando aquellas entradas identificadas previamente como erróneas o ilegibles. Esto asegura que los datos analizados sean confiables y útiles para la posterior modelación y evaluación.

A cada una de estas muestras se le aplicó un Análisis de Componentes Principales (PCA), con el objetivo de obtener una primera visión de la estructura interna de los datos, identificar posibles agrupamientos, redundancias o direcciones principales de variabilidad. Si bien en esta fase aún no se ha definido una métrica formal para la interpretación cuantitativa de los componentes, la observación de las proyecciones resultantes permite evaluar de manera cualitativa ciertos aspectos relevantes, tales como:

- Distribución y dispersión de los puntos en el espacio reducido.

- Posibles patrones.
- Existencia de outliers u observaciones atípicas.
- Influencia de las dimensiones originales en los componentes principales.

Este análisis por bloques también proporciona un marco para comparar la estabilidad estructural del conjunto de datos: si las distintas muestras presentan comportamientos similares bajo PCA, se fortalece la hipótesis de que el conjunto tiene una distribución coherente. En cambio, variaciones abruptas entre muestras pueden indicar la presencia de subgrupos diferenciados, sesgos o la necesidad de normalización adicional.

El uso de múltiples muestras en lugar de una única instancia permite obtener una perspectiva más robusta y confiable del comportamiento general del dataset, sentando las bases para las siguientes etapas de limpieza, modelado y evaluación.

Como parte del proceso exploratorio, también se implementó un primer intento de modelo de predicción utilizando el algoritmo Random Forest. Este modelo no fue concebido como una solución final, sino como una herramienta preliminar para familiarizarse con el flujo de entrenamiento, validación y evaluación dentro del contexto del problema. La experimentación con Random Forest en esta fase proporcionó información valiosa sobre la factibilidad del modelado y sobre las transformaciones previas requeridas para optimizar el desempeño en fases posteriores.

Los resultados obtenidos en esta etapa y las conclusiones derivadas pueden consultarse en el archivo **E1Descripcion_Codigo_Analisis_Exploratorio**.

4.4.0.2. Segundo entregable

Cabe señalar que en etapas previas del proyecto no fue posible aplicar de manera extensiva las técnicas de reducción de dimensionalidad sobre el conjunto de datos completo, debido a limitaciones computacionales en el entorno local de trabajo. Estas restricciones obligaron a realizar tanto el análisis exploratorio como los entrenamientos iniciales sobre subconjuntos reducidos del dataset, lo cual, si bien permitió establecer una línea base funcional, comprometía en cierta medida la generalización y profundidad del análisis. Sin embargo, en esta fase final del estudio se logró disponer de una infraestructura más robusta mediante el uso de Google Colab Pro, lo que permitió aprovechar recursos computacionales adicionales como aceleración por GPU y mayor capacidad de memoria RAM. Esta mejora hizo posible la ejecución del pipeline completo de preprocesamiento y entrenamiento sobre la totalidad del conjunto de datos, incluyendo técnicas demandantes como la estimación robusta de covarianza mediante el algoritmo FAST-MCD, la detección de outliers multivariantes basada en distancias de Mahalanobis, y la posterior reducción de dimensionalidad con PCA sobre datos robustamente transformados. Este salto en capacidad computacional no solo habilitó un análisis más exhaustivo y realista, sino que también permitió evaluar el impacto de estos métodos robustos en la calidad predictiva de distintos modelos de clasificación, ofreciendo conclusiones más sólidas y representativas.

Análisis exploratorio y diseño del preprocesamiento Dentro de la fase experimental, como primer paso se desarrolló un análisis exploratorio detallado, registrado en el notebook `AnalisisExploratorioDatosALA`, para comprender la estructura, tipo de variables, valores nulos y distribución de los atributos del dataset proporcionado por Microsoft. Esta

exploración fue esencial para diseñar una estrategia de preprocesamiento efectiva y adaptada a la naturaleza de los datos. A partir de este análisis se definió la función `preprocess_data`, implementada en los notebooks Entrenamiento1 y Entrenamiento2. Esta función incorpora la imputación de valores faltantes y preparación de los datos para su posterior uso con modelos de machine learning. Además, una vez aplicada, se complementa con la codificación categórica (One-Hot Encoding para variables nominales con pocas clases únicas y Target Encoding para categóricas ordinales o de alta cardinalidad)

4.4.0.2.0.1. Entrenamiento de modelos: Fase 1 – Baseline

En la primera etapa experimental se entrenaron tres modelos base con el objetivo de establecer un punto de referencia (“baseline”) para el rendimiento esperado sin técnicas de reducción robusta de dimensionalidad. Los modelos entrenados fueron:

1. Regresión logística, como primera aproximación para evaluar linealidad.
2. Random Forest, como modelo de tipo ensemble con capacidad para manejar datos tabulares.
3. LightGBM, un modelo de boosting basado en histogramas que ha demostrado un rendimiento competitivo en múltiples competencias y escenarios reales.

Los resultados mostraron que la regresión logística no logró capturar relaciones significativas en los datos, mientras que LightGBM alcanzó un AUC de 0.7079, superando a los demás y estableciendo así el baseline del proyecto. Entrenamiento de modelos: Fase 2 – Integración de PCA clásico y robusto En la segunda fase experimental, se integraron dos enfoques distintos de reducción de dimensionalidad:

1. PCA clásico, tras estandarización de variables numéricas.
2. PCA robusto, que incorpora:
 - Estimación robusta de la media y covarianza usando el algoritmo Minimum Covariance Determinant (MCD).
 - Detección de outliers multivariantes mediante distancia de Mahalanobis.
 - Remoción opcional de outliers (no implementada por crash del entorno).
 - Escalado robusto basado en Σ_{robusta} .
 - PCA posterior aplicado sobre datos limpios.

Este procedimiento fue implementado en el método `preprocesar_con_mahalanobis_y_pca`.

4.4.0.2.0.2. Evaluación de modelos avanzados

Los modelos evaluados fueron:

- LightGBM + PCA Robusto²
 - AUC: 0.6952
 - Accuracy: 0.6378
 - F1: 0.6275

²Realmente se hicieron dos entrenamientos para cada uno de los modelos LightGBM + PCA Robuto y Random Forest + PCA Robusto. En cada entrenamiento hay una diferencia en el preprocesamiento de los datos; sin embargo, los resultados fueron muy similares (para más detalles vea el notebook Entrenamiento1)

- Random Forest + PCA Robusto
 - AUC: 0.6911
 - Accuracy: 0.6351
- LightGBM + PCA clásico
 - AUC: 0.7052
 - Accuracy: 0.6449

Aunque el PCA robusto mostró mejoras respecto al baseline en cuanto a estabilidad frente a outliers, el PCA clásico combinado con LightGBM logró resultados casi equivalentes al modelo base, lo que sugiere que un preprocesamiento cuidadoso puede reducir la necesidad de métodos robustos si el dataset no presenta una contaminación extrema.

4.5. Conclusiones

- El análisis de datos sobre detección de malware permitió identificar variables relacionadas con la seguridad del sistema, el hardware y la ubicación geográfica de los dispositivos.
- El dataset original es muy extenso (cerca de 8 millones de registros), lo cual impuso restricciones computacionales y exigió el uso de muestras para realizar un análisis exploratorio efectivo inicialmente
- Se evidenció alta correlación entre varias variables numéricas categóricas.
- El modelo de regresión logística mostró bajo desempeño, indicando que no existe una relación lineal fuerte entre las variables predictoras y la variable objetivo.
- LightGBM se consolidó como el modelo con mejor rendimiento general, sirviendo como baseline para comparación con otros modelos entrenados
- El PCA robusto, aunque no superó al baseline, mostró buen rendimiento, validando su uso en contextos con alta presencia de valores atípicos.
- La variable objetivo está balanceada, lo cual simplificó el entrenamiento de los modelos sin necesidad de aplicar técnicas de balanceo.
- Se realizó un preprocesamiento riguroso, incluyendo la selección de variables numéricas con significado, imputación por media y moda, y eliminación de nulos en distintos enfoques, entre otros métodos descritos en los notebooks de análisis exploratorio y de entrenamiento
- El proyecto evidencia la necesidad de usar herramientas y técnicas especializadas para manejar grandes volúmenes de datos y mejorar la eficiencia en análisis de ciberseguridad.
- Los hallazgos obtenidos a partir de las muestras deben ser validados posteriormente sobre el dataset completo para garantizar la robustez de los modelos y las inferencias.
- A partir de los resultados en el estudio con el clustering espectral obtenidos se puede concluir que ni USPEC ni USENC lograron descubrir una estructura de agrupamiento relevante en los datos analizados, lo cual se refleja en los valores extremadamente bajos del NMI promedio. Este comportamiento se mantuvo incluso al variar el tratamiento de los valores faltantes y al aplicar submuestreo aleatorio. Por tanto, es razonable suponer que el conjunto de datos carece de una estructura de clusters bien definida, al menos desde la perspectiva

de estos métodos espectrales. Alternativamente, también es posible que las características consideradas no sean las más apropiadas para captar agrupamientos significativos, o que las distancias utilizadas no reflejen adecuadamente la similitud entre los objetos del conjunto.

En cualquier caso, los resultados obtenidos justifican una exploración adicional con otras técnicas de reducción de dimensión, normalización, o selección de variables, así como la evaluación de métodos de agrupamiento alternativos que puedan modelar mejor la posible estructura subyacente.

4.6. Glosario

- **AUC (Area Under the Curve):** Métrica de evaluación usada en clasificación binaria. Representa el área bajo la curva ROC y mide la capacidad del modelo para distinguir entre clases.
- **Aprendizaje automático (machine learning):** Subcampo de la inteligencia artificial que desarrolla algoritmos capaces de aprender patrones a partir de datos, con o sin supervisión humana.
- **Bosque aleatorio (Random Forest):** Algoritmo de clasificación que combina múltiples árboles de decisión para mejorar la precisión y controlar el sobreajuste.
- **Cadenas de texto:** Fragmentos legibles de caracteres encontrados en archivos ejecutables que pueden dar pistas sobre el comportamiento del software (por ejemplo, URLs, rutas o comandos).
- **Clasificadores supervisados:** Algoritmos de aprendizaje automático que aprenden a partir de ejemplos etiquetados para predecir clases futuras (e.g., benigno o malicioso).
- **Dataset:** Conjunto de datos estructurados que contienen ejemplos (muestras) con sus respectivas características y etiquetas para el entrenamiento y evaluación de modelos.
- **Feature extraction (extracción de características):** Proceso mediante el cual se identifican y seleccionan atributos relevantes de los datos (ya sea estáticos o dinámicos) para representar cada muestra.
- **Firmas estáticas:** Identificadores únicos (como hashes o patrones de código) usados tradicionalmente por antivirus para reconocer malware conocido.
- **Malware:** Programa o código malicioso diseñado para dañar, infiltrarse o controlar sistemas informáticos sin autorización.
- **Malware metamórfico:** Tipo de malware que reescribe su propio código internamente en cada ejecución para evitar su detección.
- **Malware polimórfico:** Malware que cifra o modifica su código externo (por ejemplo, por medio de packers u ofuscadores) para cambiar de apariencia sin alterar su funcionalidad.
- **Naïve Bayes:** Clasificador probabilístico basado en el teorema de Bayes. Asume independencia entre características, lo que lo hace rápido pero limitado en ciertos contextos.

- **Obfuscation (ofuscación):** Técnica para dificultar el análisis de un programa, alterando su código o estructura sin cambiar su funcionalidad.
- **Opcode (código de operación):** Instrucción que especifica la operación que debe realizar el procesador. Su análisis permite inferir patrones de ejecución de un binario.
- **PCA (Principal Component Analysis):** Técnica estadística de reducción de dimensionalidad que transforma los datos originales en un conjunto reducido de variables no correlacionadas (componentes principales) que retienen la mayor varianza posible.
- **Precisión (accuracy):** Métrica de evaluación que indica la proporción de predicciones correctas sobre el total de muestras.
- **Recall (sensibilidad):** Métrica que mide la proporción de verdaderos positivos identificados correctamente por el modelo respecto al total real de positivos.
- **Redes Neuronales Artificiales (RNA):** Modelos inspirados en la estructura del cerebro humano, capaces de aprender relaciones complejas entre datos de entrada y salida.
- **Sandbox:** Entorno controlado y aislado donde se ejecutan programas sospechosos para observar su comportamiento sin poner en riesgo el sistema real.
- **SVM (Support Vector Machine):** Algoritmo de clasificación que encuentra el hiperplano óptimo para separar clases en el espacio de características.

Bibliography

- [1] G. Strang, *Introduction to Linear Algebra*, 5th ed. Wellesley-Cambridge Press, 2016.
- [2] P. J. Davis, “The Theory of Spectral Representations.” [Online]. Available: <https://math.mit.edu/~dav/spectral.pdf>[◦]
- [3] D. C. Lay, *Matrix Algebra Useful for Statistics*, 1st ed. Hoboken, NJ: Wiley-Interscience, 2012.
- [4] S. Vatansever and R. K. Gupta, “Can I use principal component and factor even though eigenvalue is lower than 1?” [Online]. Available: https://www.researchgate.net/post/Can_I_use_principal_component_and_factor_even_though_eigenvalue_is_lower_than_1[◦]
- [5] M. Bleicher, “Range Rule for Standard Deviation.” [Online]. Available: <https://www.thoughtco.com/range-rule-for-standard-deviation-3126231>[◦]
- [6] N. Research, “Robust PCA: How to deal with outliers in dimensionality reduction?.” [Online]. Available: <https://nirpyresearch.com/robust-pca/>[◦]
- [7] M. Pourahmadi, “Modified Cholesky Decomposition and Covariance Estimation,” *Biometrika*, vol. 86, no. 3, pp. 587–602, 1999, doi: 10.1093/biomet/86.3.587[◦].
- [8] X. Kang, J. Lian, and X. Deng, “On Block Cholesky Decomposition for Sparse Inverse Covariance Estimation,” *Statistica Sinica*, 2021, doi: 10.5705/ss.202020.0211[◦].
- [9] P. J. Rousseeuw and K. Van Driessen, “A Fast Algorithm for the Minimum Covariance Determinant Estimator,” *Technometrics*, vol. 41, no. 3, pp. 212–223, 1999, doi: 10.1080/00401706.1999.10485670[◦].
- [10] M. Yuan, “High dimensional inverse covariance matrix estimation via linear programming,” *Journal of Machine Learning Research*, vol. 9, pp. 2261–2289, 2007, [Online]. Available: <https://www.jmlr.org/papers/volume9/yuan08a/yuan08a.pdf>[◦]
- [11] R. A. Maronna, D. R. Martin, and V. J. Yohai, *Robust Statistics: Theory and Methods*. Wiley, 2006.
- [12] O. Ledoit and M. Wolf, “A well-conditioned estimator for large-dimensional covariance matrices,” *Journal of Multivariate Analysis*, vol. 88, no. 2, pp. 365–411, 2004, doi: 10.1016/S0047-259X(03)00096-4[◦].
- [13] P. J. Rousseeuw, “Least median of squares regression,” *Journal of the American Statistical Association*, vol. 79, no. 388, pp. 871–880, 1984, doi: 10.2307/2288718[◦].
- [14] M. Hubert, P. J. Rousseeuw, and K. Vanden Branden, “ROBPCA: A new approach to robust principal component analysis,” *Technometrics*, vol. 47, no. 1, pp. 64–79, 2005, doi: 10.1198/004017004000000563[◦].
- [15] “Malware - Wikipedia.” [Online]. Available: <https://es.wikipedia.org/wiki/Malware>[◦]
- [16] N. Autor, “A static malware detection system using data mining methods,” *Nombre de la Revista o Conferencia*, no. Número, p. páginas, 2008, doi: DOI-si-lo-tenes[◦].
- [17] F. Toolan and J. Carthy, “Feature selection for spam and phishing detection,” *eCrime Researchers Summit (eCrime)*, pp. 1–12, 2010, doi: 10.1109/eCrime.2010.5668493[◦].
- [18] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, “A Survey on Automated Dynamic Malware Analysis Techniques and Tools,” *ACM Computing Surveys*, vol. 44, no. 2, pp. 1–42, 2012, doi: 10.1145/2089125.2089126[◦].
- [19] D. Suhartono and others, “Neural Network with Principal Component Analysis for Malware Detection,” in *Proceedings of the 16th International Conference on Security and Cryptography*, 2019, pp. 270–277. [Online]. Available: <https://www.scitepress.org/Papers/2019/99089/99089.pdf>[◦]
- [20] A. Mahindru *et al.*, “Enhancing malware detection with feature selection and scaling techniques,” *Scientific Reports*, vol. 15, no. 1, p. 10724, 2025, doi: 10.1038/s41598-025-93447-x[◦].
- [21] A. Kabakus and others, “Evaluation of Principal Component Analysis Variants to Assess Their Suitability for Mobile Malware Detection,” *ResearchGate*, 2022, [Online]. Available: <https://www>.

researchgate.net/publication/361523264_Evaluation_of_Principal_Component_Analysis_Variants_to_Assess_Their_Suitability_for_Mobile_Malware_Detection^o

- [22] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007, doi: 10.1007/s11222-007-9033-y^o.
- [23] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an Algorithm," *Advances in Neural Information Processing Systems*, vol. 14, pp. 849–856, 2002.
- [24] D. Huang, C.-D. Wang, J.-S. Wu, J.-H. Lai, and C.-K. Kwoh, "Ultra-Scalable Spectral Clustering and Ensemble Clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 6, pp. 1212–1226, Jun. 2020, doi: 10.1109/TKDE.2019.2903410^o.
- [25] Guava, "Spectral clustering for large scale datasets (Part 1)." 2023.
- [26] scikit-learn developers, "Spectral Clustering — scikit-learn documentation." 2024.
- [27] S. X. Yu and J. Shi, "Multiclass spectral clustering," in *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003, pp. 313–319. doi: 10.1109/ICCV.2003.1238361^o.
- [28] A. Damle, V. Minden, and L. Ying, "Simple, direct, and efficient multi-way spectral clustering," *SIAM Journal on Scientific Computing*, vol. 41, no. 4, pp. A2965–A2989, 2019, doi: 10.1137/18M1222313^o.
- [29] Guava, "Spectral clustering for large scale datasets (Part 2)." 2023.

Index of Figures

Figure 1 La implementación de SpectralClustering en scikit-learn (véase [26]) no sigue estrictamente el algoritmo propuesto por [23], sino que se alinea más con enfoques posteriores como los presentados por [27] y [28], enfocados en la segmentación multiclase y estrategias eficientes de partición espectral. Aunque todas estas propuestas comparten la idea central de usar los vectores propios del Laplaciano normalizado del grafo de similitud, scikit-learn no aplica la normalización fila a fila del embedding espectral antes de la asignación por K-Means, como lo hace el algoritmo original de Ng et al. En cambio, permite al usuario seleccionar entre distintas estrategias de asignación de etiquetas mediante el parámetro `assign_labels`:

- “kmeans” es el enfoque por defecto y consiste en aplicar el algoritmo de K-Means sobre los vectores propios obtenidos del Laplaciano; esta estrategia es más flexible y puede capturar particiones más finas, pero depende de inicializaciones aleatorias, por lo que puede producir resultados distintos en ejecuciones sucesivas si no se fija el `random_state`.
- “discretize” implementa una técnica basada en rotación ortogonal seguida de asignación de etiquetas por máximos, lo que genera particiones más simétricas y geoméricamente uniformes; además, es totalmente determinista.
- “cluster_qr” es una opción más reciente que aplica una factorización QR con pivoteo sobre el embedding espectral, permitiendo una asignación de etiquetas también determinista pero con mejor fidelidad geométrica que discretize, lo que resulta en particiones visualmente más coherentes en muchos casos prácticos.

Se intentó ejecutar esta técnica sobre un conjunto de datos de 8.921.483 registros, pero debido a la complejidad cúbica del cálculo espectral, la operación resultó computacionalmente inviable: después de más de cuatro horas de ejecución, el proceso no finalizó en múltiples pruebas. Además, se emitieron advertencias como “Graph is not fully connected, spectral embedding may not work as expected”, lo cual indica que la matriz de similitud generó un grafo no conexo, comprometiendo la validez del embedding espectral y, por ende, la calidad del agrupamiento. 35

Figure 2 Definición matriz de afinidad dispersa B. Aquí r_j es uno de los K representantes más cercanos 36