UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ACATLÁN

INTRODUCCIÓN AL ANÁLISIS DE CLÚSTER, AL MODELO PREDICTIVO DE LOS K-VECINOS MÁS CERCANOS Y SU IMPLEMENTACIÓN EN EL SECTOR FINANCIERO

TESINA Y EXAMEN PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE
LICENCIADO EN ACTUARÍA

PRESENTA:
RICARDO FIGUEROA BRACAMONTES

DRA. INGRID CHANTAL TORRES RAMOS

Santa Cruz Acatlán, Naucalpan, Estado de México. 2023

Índice general

Prefacio

Justificación y planteamiento del problema

Este trabajo nace por la búsqueda de atender una necesidad en el área en la que laboro, sin embargo por cuestiones inherentes a los estragos de la pandemia no se lograron implementar del todo los hallazgos obtenidos. Sin embargo, por mi inquietud científica y constructiva de la carrera, decidí continuar con el planteamiento y solución del problema, poniendo a prueba los conocimientos adquiridos durante mi formación académica y en el desarrollo del trabajo.

La necesidad en cuestión se resume en brindar pronósticos confiables sobre el desempeño a futuro de los empleados operativos de la organización con la finalidad de tener un criterio cuantitativo para la liberación de promociones, aumentos salariales y prestaciones.

Por otro lado, si bien existe bibliografía orientada a la introducción teórica de las diferentes técnicas de análisis de datos, no son así de numerosos los trabajos que aborden de manera integral la aplicación de la misma en casos prácticos dentro del ámbito profesional y aún menos enfocados en el sector financiero. En particular, gran parte de la licenciatura se desarrollan ejemplos basados en escenarios hipotéticos e ideales y, por consiguiente, la elaboración de esta propuesta cobra una gran relevancia ya que el trabajo cubrirá una reseña histórica, el sustento teórico del análisis de clúster hasta su implementación en una necesidad real y con la correspondiente evaluación de su eficacia. Adicionalmente, se revisará y empleará el lenguaje de programación R, al igual que paqueterías que se emplean para el uso de técnicas de análisis de datos.

Objetivos generales y particulares

Como objetivos generales en este trabajo, se busca introducir los modelos de clasificación y técnicas de análisis de datos mediante sus conceptos básicos y el sustento teórico en los que se basan. Por otro lado, se busca ilustrar la implementación de diversos métodos como el t-SNE, técnicas y algoritmos como el k-nearest neighbours, prescindiendo de condiciones ideales y, al mismo tiempo, atender una necesidad real en el campo profesional.

En particular, se busca:

Brindar una introducción a ciertas técnicas de análisis de datos desde sus conceptos básicos hasta la teoría estadística en la que se sustentan. Mostrar los paquetes (librerías) y comandos del lenguaje de programación R existentes para su implementación. Ilustrar el proceso de implementación de dichas técnicas.

A continuación se enlista una descripción breve de cada uno de los capítulos que se abordarán en el presente.

1. **Introducción** En este capítulo se brinda un reforzamiento de los conceptos de clasificar y pronosticar así como del impacto que tienen en nuestra

vida diaria sin que nos demos cuenta. Asimismo, se da un breve repaso histórico de la evolución de una de las clasificaciones más antiguas; la clasificación taxonómica de los seres vivos. Finalmente se enlistan las aportaciones más valiosas a la estadística y los autores de las mismas, rama de donde nacen prácticamente todas las técnicas abordadas en el presente trabajo.

2. **Sustento teórico** Se brinda una introducción teórica a las diversas técnicas y conceptos que se aplican en la resolución del problema práctico que se desea atacar: El análisis de clúster. Partiendo de los conceptos básicos de distancia claves para poder comprender el análisis de clúster y abarcando tanto las técnicas de partición como las jerárquicas. Por otro lado se aborda el modelo predictivo K-nearest Neighbors y la técnica t-SNE para reducción de dimensión.

3. **Aplicación** En él se ataca el problema práctico mediante las diversas técnicas descritas anteriormente mediante lenguaje de programación R.

4. **Conclusiones** Se brindan las principales conclusiones que se desprenden del estudio de estas técnicas y resolución del problema mediante las mismas.

Capítulo 1

Introducción

1.1. Antecedentes

1.1.1. La necesidad de clasificar

La palabra *clasificar* es asociada con la acción de ordenar, organizar o situar ciertos objetos o sujetos dentro de alguna categoría de acuerdo con determinados criterios. La necesidad de clasificar ha nacido a raíz de la numerosa y creciente cantidad de elementos con los que el ser humano ha interactuado a lo largo de la historia.

Hoy en día, y casi de forma inconsciente, las clasificaciones abarcan prácticamente cualquier aspecto de la vida en sociedad; desde la ropa y calzado que uno decide ponerse al iniciar el día (deportiva, formal, casual, entre otros) hasta el género de la película con la que pretende entretenerse en los ratos de ocio (acción, ciencia ficción, suspenso, etcétera). El simple hecho de nombrar un objeto es clasificarlo; por ejemplo un *automóvil* es un objeto que posee ciertos atributos que no cumple una *bicicleta*, es decir, ambos pertenecen a categorías distintas a pesar de que ambos pertenecen a la clase de objetos denominados *vehículos*.

En años recientes, incluso han surgido algoritmos que estiman las preferencias de una persona de acuerdo con sus características (edad, estrato social, grado de estudios, sexo, entre otros) uno de los objetivos es brindar publicidad personalizada a través de distintos medios, llegando a generar controversia e incluso levantando sospechas de *espionaje*; sin embargo, únicamente se trata de algoritmos clasificatorios modernos surgiendo la siguiente pregunta natural ¿cómo se ha llegado a este punto?

1.1.2. La clasificación taxonómica de los seres vivos

Dada la inmensa cantidad de disciplinas en las que se han aplicado técnicas de clasificación a lo largo de la historia, el profundizar en cada una de ellas ameritaría una investigación en sí misma. Con fines prácticos se abordará la trayectoria de la taxonomía de los seres vivos, rama de la biología y quizá una de los esquemas de clasificación con mayor antigüedad, por naturaleza el ser humano ha clasificado las plantas y animales desde que tuvo uso raciocinio. En general, el mundo vivo está objetivamente estructurado por tal motivo un objetivo de la biología es el descubrir y documentar esta estructura, una forma de hacerlo es por medio de la clasificación, es en este punto que la taxonomía biológica cobra importancia. La idea básica de las estructuras taxonómicas actuales en el ámbito biológico consiste en hallar patrones y semejanzas entre la gran variedad de organismos existentes y sus características. No sería demasiado aventurado pensar que esta ha sido la mecánica que el ser humano ha empleado para clasificar los organismos de su entorno desde la prehistoria pues incluso los animales son capaces de hacer clasificaciones inconscientes y de forma muy primitiva entre organismos comestibles y organismos que pudiesen representar un peligro como lo son los depredadores.

Si bien, la idea base es relativamente simple a lo largo de la historia han existido diversas clasificaciones cada una definida por el conocimiento y necesidades de la época aunque en general siempre se buscó cumplir dos estándares; englobar a todos los seres vivos conocidos en su época y, que un sujeto no pudiese pertenecer a dos clases distintas. Dado que de forma continua se han ido descubriendo nuevas especies, estos criterios debieron irse modificando a lo largo de la historia.

Si se piensa en el llamado *Hombre de las cavernas* puede intuirse que era de vital importancia el identificar qué plantas y animales representaban un peligro y cuáles una presa o sustento potencial, teniendo así una clasificación primitiva de los seres vivos en su entorno. Sin embargo, los primeros registros que abordan la materia se remontan al año 350 a.C. con Aristóteles, quien observó y dividió alrededor de 520 especies animales en 2 grupos: los que tienen sangre roja y los que no, esta clasificación concuerda con el concepto actual de vertebrados e invertebrados. Unos años después, alrededor del 320 a.C. Treofrasto, discípulo de Aristóteles, estudia la anatomía de las plantas y las clasifica por su tamaño y estructura en su libro *"De Historia Plantarum"*.

Al igual que en muchas otras ramas de la ciencia durante la Edad Media, se tuvieron varios siglos de estancamiento en el desarrollo de la taxonomía. No fue si no hasta el siglo XVIII cuando Carl von Linné sentó las bases de la taxonomía moderna, misma que aún sigue vigente a grandes rasgos, asignando a cada organismo viviente en una categoría taxonómica mediante una clasificación jerárquica definiendo primero el *Reino* al que pertenece, posteriormente el *Filo, Clase, Orden, Familia, Género* y, finalmente, su *Especie*. Cada especie tiene una nomenclatura única en función de su género y un epíteto específico, este último usualmente está definido por la localización geográfica característica de

la especie o el nombre de quien la descubre o describe. El género inicia con mayúscula y el epíteto con minúscula ambas en cursivas o subrayadas teniendo así un sistema de nomenclatura universal.

A pesar de los avances logrados, hasta este punto los únicos *Reinos* existentes eran *Plantae* y *Animalia*, es decir, todos los seres vivos estudiados se clasificaban como animales o como plantas. Fue hasta que empezó a descubrirse y estudiarse la vida unicelular que los científicos se dieron cuenta de que no siempre era posible clasificar a estos organismos como miembros de uno u otro reino dado que sus características no eran totalmente afines a ellos. En 1866, Ernst Haeckel propuso crear el reino *Protista* en el que se clasificaba a todos los organismos unicelulares.

A mediados del siglo XX con los avances tecnológicos y científicos se fue descubriendo que aún dentro del mundo microscópico había diferencias entre los organismos. Se descubrió que había células eucariontes que poseen núcleos y organelos y células procariontes que carecen de ellos. Dadas estas diferencias, Herbert Copeland propuso, en 1956, un cuarto reino llamado *Monera* en el que se clasificaba a las bacterias por ser todas procariontes.

Siguiendo la inercia de los avances realizados se fueron conociendo a profundidad ciertas características de los hongos que los descalificaron como integrantes del reino *Plantae*. En el año 1969 Robert H. Whittaker brindo un esquema de criterios que refinaba la clasificación ya dada; en él se definieron cinco reinos en función de los siguientes criterios:

- Tipo celular: procariontes y eucariontes.
- Nivel de organización: unicelular y pluricelular
- Tipo de nutrición: autótrofa y heterótrofa.
- Tipo de reproducción: sexual y asexual.

Dados estos criterios, propuso un quinto reino llamado *Fungi* en el que se engloba a los hongos pues poseen ciertas características incompatible con el reino *Plantae* en el que se les clasificaba anteriormente.

Al igual que en muchas áreas, estos conceptos no son inalterables si no que se encuentran sujetos a nuevos descubrimientos y consensos científicos. Sin embargo, en esencia, la clasificación que se utiliza sigue estos principios.

Y así podría enumerarse una larga lista de ejemplos en los que se muestra que la clasificación ha acompañado al ser humano a lo largo de su historia. En el caso de la taxonomía y por la antigüedad de esta necesidad se tienen pautas establecidas que determinan las diferentes clases existentes; sin embargo, resultaría de utilidad el contar con un método que pudiese clasificar determinada variedad de objetos o sujetos a pesar de no disponer de criterios establecidos

como en el caso anterior, o bien, que la clasificación permanezca estable al ingresar nuevas observaciones.

Con el surgimiento y evolución de las computadoras, se ha accedido a una potencia de cálculo anteriormente inimaginable para el ser humano. Esta función resulta muy útil al ejecutar análisis numéricos especializados en el ***reconocimiento de patrones*** que de forma manual sería difícil ejecutar, por no decir imposible. Es en este punto que surge el *análisis de cluster* que es capaz de establecer clases y criterios que definan partiendo de un conjunto de objetos individuales a su vez descritos por un conjunto de características, en otras palabras, brinda criterios universales de clasificación. Para una introducción más profunda a las técnicas de análisis de cluster y su aplicación en R, consultar [8].

1.1.3. La necesidad de pronosticar

Ya lo decía el conocido filósofo Chino, Confucio (551 - 478 a.C.) "Estudia el pasado si quieres pronosticar el futuro", idea que si bien fue expuesta hace ya más de dos mil años se mantiene el día de hoy como un axioma sustentado por una ingente teoría estadística pues efectivamente, el realizar pronósticos (uno de los objetivos más importantes de esta rama) se logra mediante el estudio de datos recopilados inherentes al pronóstico que se desea hacer.

El alcance de contar con pronósticos fiables y sustentados matemáticamente es inmenso. Si bien para el grueso de la población podría ser relevante en decisiones cotidianas como el tomar precauciones ante alertas de tormenta en días posteriores o decidir invertir en determinado sector por buenos pronósticos de crecimiento, para una persona con alto rango de poder (gobernadores, directores, empresarios, etc.) la materia cobra aún más relevancia por la necesidad de dar sustento a decisiones que determinen el éxito o fracaso de la organización.

Si bien en la antigüedad ya se tenían nociones numéricas y estadísticas, estas últimas consistían en registros orientados en llevar un control descriptivo de poblaciones, producción agrícola y cálculo de recursos. Es decir, la estadística no adquiría en ese entonces el objetivo de realizar pronósticos. Se tiene registro de métodos similares para llevar control de los recursos y censos del imperio romano e, inclusive, durante el siglo XVI se llevaron registros demográficos relacionados con las epidemias sufridas en ese entonces.

Fue hasta entrado el siglo XVII que los matemáticos franceses Blaise Pascal y Pierre de Fermat establecen fundamentos y nociones sobre la probabilidad mediante juegos de azar, fundamentos sobre los que Christian Huygens (físico, matemático y astrónomo danés) publicó en 1656 el primer tratado sobre teoría de probabilidad en su obra titulada *"Sobre el razonamiento en los juegos de azar"*.

Durante el sigo XVIII la estadística descriptiva vuelve a cobrar relevancia en asuntos sociales y económicos, y es a finales de ese siglo y comienzos del XIX

cuando, con los trabajos de Joseph Louis Lagrange, Pierre Simon de Laplace, Carl Friedrich Gauss y Simeón-Denis Poisson, la teoría de la probabilidad comienza a tomar forma. Cabe señalar también el descubrimiento de la distribución normal por Abraham de Moivre, distribución que será posteriormente "redescubierta" por Gauss y Poisson.

Si bien ya se comenzaba a formar un sustento teórico de esta rama, fue Jacques Quételect el precursor de la aplicación de la misma al utilizar el principio de promedios y variabilidad para la descripción de fenómenos sociales.

Durante el transcurso del siglo XIX la teoría siguió desarrollándose con el nacimiento de los conceptos de error (Laplace y Gauss), la teoría de los mínimos cuadrados (Laplace, Gauss y Legendre), el concepto de la correlación entre variables acuñado por Sir Francis Galton, terminó que siguió desarrollando Karl Pearson dando a luz el concepto *coeficiente de correlación*.

Los siguientes avances destacados en la materia fueron aportados por Francis Galton y Kurt Pearson dando a la ciencia un enfoque más aplicado empleándola en análisis de experimentos mediante el test de Chi-cuadrada que sigue aplicándose y estudiándose en la actualidad, enfoque que profundizaron Egos Pearson y Jerzy Neyman fundando las pruebas modernas de contraste de hipótesis.

Puede decirse que Sir Ronald Arnold Fisher fue clave para consolidar la estadística tal y como se conoce en la actualidad dado que en su libro *"Sobre los fundamentos de la estadística teórica"* publicado en 1922. En dicho texto estableció los principios a partir de los cuales se desarrolló la teoría que se utiliza hoy en día así como los llamados *métodos clásicos*.

Las figuras destacadas en la materia no provinieron únicamente de Norteamérica y Europa, personas originarias de Rusia como Pafnuty Chebichev, Andrei Harkov, Alexander Khinchin y Andrey Kolmogorov hicieron grandes aportes con el desarrollo de diversas técnicas que, del mismo modo, son ampliamente aplicadas el día de hoy y llevan sus nombres.

Ya en la era contemporánea el método científico se ha extendido como una herramienta universal en el estudio de cualquier fenómeno natural o social y con ello se ha obtenido en la estadística una herramienta indispensable en la mayoría de investigaciones pues logra reducir a términos matemáticos la información y no deja lugar a ambigüedades.

Tal ha sido la implemetación de la estadística en ciertas áreas que en la actualidad no es posible la liberación de un nuevo fármaco si no ha sido sujeto a las más rigurosas pruebas estadísticas y es el lenguaje universal al hablar de datos económicos, políticos, sociales, psicológicos, biológicos y físicos.

Como la rama madura que es, es posible segmentar la estadística en diversas áreas, una muy amplia es la estadística descriptiva que, como su nombre hace intuir, se centra en la descripción de los datos. En el presente trabajo no es este el enfoque de interés si no el de la pronosticación.

Capítulo 2

Sustento teórico

Aunque la estadística tiene sus orígenes desde el siglo XVIII, en años recientes ha surgido gran diversidad de métodos y algoritmos que, aunque están sustentados en buena medida sobre teoría estadística, no puede decirse que todos sean 100 % métodos estadísticos si no que se sirven de la inmensa capacidad para realizar cálculos que poseen las computadoras, en el reconocimiento de patrones sin necesidad de ajustar los datos a distribuciones de probabilidad. Estos métodos han demostrado ser altamente precisos y de práctica implementación, es por ello que son materia del presente.

2.1. Análisis de cluster

El concepto de análisis de cluster no implica un método *per se*, si no que engloba diversos procedimientos estadísticos multivariantes que, partiendo de una muestra de datos que describa a un grupo de objetos individuales y sus características, buscan brindar clases de ellos que compartan ciertos atributos. Estas clases son llamadas ***clusters***. Lo que destaca al análisis de cluster es que no es necesario (aunque si útil) conocer a priori la naturaleza de las clases, estos métodos buscan la clasificación óptima a modo de minimizar la similitud entre miembros de distintas clases y maximizar la de los miembros de la misma clase.

Si bien la estadística y la mayoría de sus ramas han tenido tiempo para madurar, las técnicas relacionadas con el análisis de cluster son relativamente recientes. Una de las primeras nociones que se tiene en la materia data de la década de 1960 por Sokal y Sneath quienes expusieron la necesidad de ampliar el esquema tradicional de clasificación de los seres vivos surgiendo la llamada *Taxonomía numérica* que proponía establecer grados de similitud entre individuos y agruparlos en taxones que contenían a los individuos más semejantes entre sí.

Aunque el surgimiento de este nuevo enfoque impulsó el desarrollo del análisis de

cluster, fue la aparición y el boom de las computadoras lo que se ha convertido en el pilar de la rama. Pues, como ejemplo, si se piensa en una muestra de 500 objetos de la que se requiere calcular el grado de similitud de cada uno con los demás sería necesario hacer 4,950 cálculos comparativos, creciendo de forma exponencial este número de cálculos conforme crezca la muestra, sin mencionar el aumento de la complejidad de los mismos al aumentar el número de variables que describan a cada objeto. Es por ello que, si bien la teoría es lo que le da cuerpo a esta rama, en la práctica no resultaría accesible la aplicación de estas técnicas sin los recursos computacionales con los que se dispone hoy día.

Si bien se ha comprobado la efectividad de estos métodos, al ser una rama reciente el sustento teórico continúa en desarrollo y se tiene aún un largo camino por recorrer para comprenderse del todo. Es por ello que no hay un método infalible si no que distintos métodos pueden generar clasificaciones distintas sobre un mismo conjunto de datos dependiendo de su estructura y es necesario ejecutar más de un método con la finalidad de poder elegir el resultado más adecuado.

2.1.1. Medidas de distancia

El primer factor a considerar al aplicar un método de clustering será la medida de distancia que se usará dependiendo de la estructura o fuente de los datos, con base en ella se definirá el criterio de similitud entre los elementos a clasificar.

Dados $x = (x_1, \ldots, x_n)$ y $y = (y_1, \ldots, y_n)$ dos vectores de dimensión n, las medidas usadas comúnmente son:

1. Distancia Ecludiana
$$d_{euc}(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$
2. Distancia Manhattan
$$d_{man} = \sum_{i=1}^{n} |x_i - y_i|$$
3. Distancia por correlación de Pearson
$$d_{cor}(x, y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Obsérvese que la medida basada en correlación considera que dos objetos son similares si se correlacionan entre sí aunque eso no necesariamente implique similitud bajo el criterio de distancia euclidiana, este tipo de medidas son usualmente empleadas en genética y marketing pues son útiles al identificar perfiles similares independientemente de la magnitud de las variables *per se*.

Definida la métrica a aplicar es necesario ***estandarizar*** las variables especialmente cuando son representadas por escalas distintas dado que de no hacerse, el criterio de similitud podría verse afectado. Lo que se busca es escalar las variables a modo de hacer que tengan media 0 y desviación estándar 1 para poder compararlas entre sí, esto puede lograrse mediante la siguiente transformación:

$$x_i' = \frac{x_i - \bar{x}}{sd(x)}$$

Una vez solventadas la estandarización y la medida de distancia (además de la correspondiente *limpieza* de los datos), será posible implementar alguna técnica de cluster. Si bien existen diversos métodos, estos se pueden diferenciar en dos grupos: jerárquicos y de partición.

2.1.2. Clusters de partición

Esta variante de técnicas se usa para clasificar observaciones en diversos grupos de acuerdo con sus similitudes y requieren que el analista especifique el número de grupos a ser generados. A continuación se darán nociones básicas de los más aplicados.

2.1.2.1. K-Means Clustering

K-means clustering es uno de los algoritmo de aprendizaje no supervisado más usado para particionar una muestra de datos en k clusters. La idea general de esta técnica es que cada cluster es representado por su centroide que se corresponde con la media de los puntos que integran cada cluster (recordando que en estos algoritmos, cada observación se representa como un punto en el espacio de las variables que definen dichas observaciones, es decir, si se tienen k variables, el punto tendrá k dimensiones).

El objetivo que busca k-means es definir los clusters a modo de minimizar la variación intra-cluster total, es decir, la suma de las disimilitudes entre los integrantes de cada cluster y su correspondiente centroide.

Existe más de un algorítmo o *aproximación* para k-means que se sustentan bajo este esquema, sus orígenes se remontan al año 1956 cuando el matemático Hugo Steinhaus publicó en francés la primera versión del algorítmo en el *Bulletin de l'Académie Polonaise des Sciences* en el atículo titulado *Sur la Division des Corps Matériels en Parties* [8] en el que se abordaba el problema de particionar un sólido heterogéneo mediante la selección adecuada de particiones mencionando aplicaciones en antropología e industria, es por ello que puede decirse que fue el primero en proponer un algorítmo explícito en espacios multidimensionales que brindaba una solución a esta necesidad.

El algoritmo siguió evolucionando mediante aproximaciones brindadas por Stuart Lloyd (de Bell laboratories), quien en el artículo titulado *Least Squares Quantization in PCM* [9] abordó la necesidad de trasmitir una señal aleatoria X en un espacio multidimensional. Lloyd Trabajó en el campo de las comunicaciones y la electrónica y su algoritmo fue presentado como una técnica para la modulación de códigos de impulsos.

Posteriormente fue James MacQueen (Department of Statistics of the University of California) quien en su artículo *Some Methods for Classification and Analysis of Multivariate Observations* [10] propuso un algorítmo para particionar un conjunto en clusters buscando que la variancia de cada cluster fuera mínima y fue él quien acuñó por primera vez el término *k-means*

Otra evolución en el algoritmo fue propuesta por Hartigan-Wong (1979), quien define la variación intra-cluster como la suma de los cuadrados de las distancias euclidianas entre los miembros del cluster y su correspondiente centroide, es decir

$$W(C_k) = \sum_{x_i \in C_k} (d_{euc}(x_i, \mu_k))^2 \tag{2.1}$$

donde x_i representa al i-ésimo punto perteneciente al cluster C_k y μ_k es la media de los puntos que pertenecen al cluster C_k. Recordando que al tratarse de espacios multivariados tanto x_i como μ_k son a su vez vectores multivariados por lo que (suponiendo que se tienen n variables) $x_i = (x_1^i, \ldots, x_n^i)$ y $\mu_k = (\mu_1^k, \ldots, \mu_n^k)$. Es decir, la expresión 2.1 puede reescribirse como

$$W(C_k) = \sum_{x_i \in C_k} \sum_{j=1}^{n} (x_j^i - \mu_j^k)^2$$

Haciendo énfasis en que siendo un espacio multivariado μ_k será un vector en el que cada variable está dada por el promedio de dicha variable para todos los puntos pertenecientes al cluster.

Siguiendo esta definición, cada punto x_i será asignado a un cluster de tal modo que la suma de distancias cuadradas a su centroide sea la mínima, definiendo la variación intra-cluster

$$\sum_{k=1}^{K} W(C_k) = \sum_{k=1}^{K} \sum_{x_i \in C_k} (d_{euc}(x_i, \mu_k))^2$$

en k-means se busca que esta variación total sea lo más pequeña posible. En resumen, la secuencia del algoritmo es la siguiente:

1. Determinar el valor de k, es decir, el número de clusters a construir.

2. Definir aleatoriamente k observaciones como los centroides de los k clusters iniciales.

3. Empleando la distancia euclidiana, asignar cada observación con el centroide inicial más cercano.

4. Redefinir los k centroides como la media de todos los integrantes de su correspondiente cluster. Teniendo en cuenta que cada observación está definida por n variables, este centroide será un vector de n dimensiones, donde la i-ésima dimensión será la media de la i-ésima variable de todos los elementos del cluster.

5. Iterar los pasos 3 y 4 hasta minimizar la variación intra-cluster

La fortaleza de este algoritmo se encuentra en su simplicidad pues es muy eficiente al trabajar con cantidades de datos masivas teniendo recursos computacionales limitados. Por otro lado, se tienen ciertas desventajas empezando por la necesidad de definir el número de clusters a construir ya que aunque hay ciertas técnicas que brindan una estimación, el hacerlo requiere de un conocimiento previo de los datos. Otra desventaja del algoritmo es que, al depender de un evento aleatorio al definir los centroides iniciales, se pueden obtener clusters resultantes distintos cada vez que se ejecute el proceso. Finalmente, otra desventaja de k-means es que, al utilizar la distancia euclidiana y medias para definir centroides, se presenta alta sensibilidad a outliers.

Otra forma de abordar el algoritmo k-means es por medio de un enfoque probabilístico, es decir, tratar de entender el algoritmo como un caso específico de modelo de mezcla finita de distribuciones Gaussianas (Gaussian Mixture Models o GMM abreviando).

Los GMM surgen a raíz de la necesidad de describir o ajustar distribuciones de probabilidad a muestras de datos multimodales a los que una distribución simple *per se* no se podría ajustar [1, p. 110-113], en este caso es conveniente describir la distribución como una superposición o combinación lineal de dos o más distribuciones (Gaussianas en este caso), la densidad de esta distribución compuesta estará descrita por:

$$p(x) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \sum_k) \tag{2.2}$$

en donde $N(x|\mu_k, \sum_k)$ representa a cada densidad Gaussiana, nombradas *componentes* de la mezcla; μ_k y $\sum_k$ representan la media y covarianza de cada componente, respectivamente.

Los π_k que son llamados *coeficientes de mezcla* o *pesos de la mezcla* pueden interpretarse como la probabilidad de tomar el k-ésimo componente y cumplen $\sum_{k=1}^{K} \pi_k = 1$. Ello se obtiene al integrar la expresión 2.2 con respecto a x. De lo que se concluye que $0 \leq \pi_k \leq 1$ puesto que $p(x) \geq 0$ y $N(X_i|\mu_k, \sum_k) \geq 0$.

En cambio en el algoritmo k-means, a cada punto x_i se la asigna un cluster $z_i = k$ en donde la distribución de cada z_i será una distribución categórica con k parámetros $\pi_k = p(z_i = k)$. De lo cual, cada punto provendrá de una distribución Gaussiana con media μ_{z_i} y covarianza $\sum_{z_i}$ con probabilidad condicionada $p(x_i|z_i = k) = N(X_i|\mu_K, \sum_K)$. Teniendo en cuenta el objetivo de k-means, será de interés *invertir* esta probabilidad, es decir, la probabilidad de que un conjunto de puntos con cierta varianza y media pertenezca a un determinado cluster, es decir, $p(z_i = k|x, \mu_k, \sum_k)$ llamada probabilidad de correspondencia.

Cuando los parámetros de un GMM $z, \mu, \pi, \sum$ son desconocidos se utiliza el algoritmo Expectation-Maximitation.

El algortimo Expectation - Maximitation (EM) inicialmente propuesto por Dempster, Laird y Rubin en 1977 [11], es una técnica iterativa que generalmente se utiliza para adecuar o estimar un modelo estadístico en los casos con datos incompletos o varibles desconocidas (como en el caso de este trabajo). Para trabajar con este algorítmo EM es necesario estimar los parámetros de los componentes de la mezcla.

En otras palabras, este algoritmo alterna entre el paso E (expectativa) que asigna los clusters por medio de las probabilidades de correspondencia, manteniendo los parámetros fijos, y el paso M (maximización) que recalcula los parámetros del cluster manteniendo fija la asignación, es decir:

1. Paso E: Dados los parámetros estimados para los clusters, se calculan las probabilidades de correspondencia

$$\gamma_{i,k} = p(z_i = k|x_i; \mu_k, \sum_k) = \frac{\pi_k N(x_i|\mu_k, \sum_k)}{\sum_{j=1}^{K} \pi_j N(x_i|\mu_j, \sum_j)}$$

2. Paso M: Calcula los parámetros que maximizan la verosimilitud del conjunto de datos $p(\chi|\pi, \mu, \sum, z)$, es decir, la probabilidad del conjunto de datos bajo el GMM

$$p(\chi|\pi, \mu, \sum, z) = \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k N(x_i|\mu_k, \sum_k) \tag{2.3}$$

   Maximizando la expresión 2.3 con respecto a cada parámetro se obtiene:

$$S_k = \sum_{i=1}^{N} \gamma_{i,k}$$

$$\mu_k = \frac{\sum_{i=1}^{N} \gamma_{i,k} x_i}{S_k}$$

$$\pi_k = \frac{S_k}{N}$$

$$\sum_k = \frac{\sum_{i=1}^{N} \gamma_{i,k}(x_i - \mu_k)(x_i - \mu_k)^T}{S_k}$$

se observa que para cada iteración del algoritmo, crece o mantiene la función de verosimilitud.

Considérese el caso especial de GMM en el que las matrices de covarianza son esféricas y compartidas entre componentes, es decir, $\sum_k = \sigma l$ para $k = 1, \ldots, K$ con l la matriz identidad y $\sigma > 0$ la varianza. Asumiendo que σ es una constante conocida, el paso E se simplifica como

$$\gamma_{i,k} = \frac{\pi_k e^{-\frac{||x_i - \mu_k||_2^2}{2\sigma}}}{\sum_{j=1}^{K} \pi_j e^{-\frac{||x_i - \mu_j||_2^2}{2\sigma}}}$$

cuando σ tiende a 0 la ecuación $\gamma_{i,k}$ se aproxima a 1 para el valor más cercano a x_i, es decir, el componente para el que la distancia Euclidiana sea mínima y el resto de componentes tendrá probabilidad de correspondencia 0 y sus correspondientes π_k dejan de tener influencia alguna y dado que tanto las distancias como σ son, en todo caso, mayores que cero

$$\begin{aligned}\lim_{\sigma \to 0} \gamma_{i,k} &= \lim_{\sigma \to 0} \frac{\pi_k e^{-\frac{\|x_i - \mu_k\|}{2\sigma}}}{\sum_{j=1}^{K} \pi_j e^{-\frac{\|x_i - \mu_j\|}{2\sigma}}} \\ &= \lim_{\sigma \to 0^+} \frac{\pi_k e^{-\frac{\|x_i - \mu_k\|}{2\sigma}}}{\sum_{j=1}^{K} \pi_j e^{-\frac{\|x_i - \mu_j\|}{2\sigma}}}\end{aligned}$$

Suponiendo que x_k es el elemento más cercano a x_i, los π_j proceden a anularse salvo para $j = k$ en cuyo caso $\pi_j = 1$ de lo cual:

$$\begin{aligned}\lim_{\sigma \to 0} \gamma_{i,k} &= \lim_{\sigma \to 0^+} \frac{(1)e^{-\frac{\|x_i - \mu_k\|}{2\sigma}}}{(0)e^{-\frac{\|x_i - \mu_1\|}{2\sigma}} + (0)e^{-\frac{\|x_i - \mu_2\|}{2\sigma}} + \ldots + (1)e^{-\frac{\|x_i - \mu_k\|}{2\sigma}} + (0)e^{-\frac{\|x_i - \mu_{k+1}\|}{2\sigma}} + \ldots + (0)e^{-\frac{\|x_i - \mu_K\|}{2\sigma}}} \\ &= \lim_{\sigma \to 0^+} \frac{e^{-\frac{\|x_i - \mu_k\|}{2\sigma}}}{e^{-\frac{\|x_i - \mu_k\|}{2\sigma}}} \\ &= 1\end{aligned}$$

Como se puede observar, el paso E del algoritmo EM coincide con el paso de asignación de k-means. Por otro lado, mientras π_k carezca de influencia, el paso

M del algoritmo recalcula únicamente los parámetros de media μ_k que corresponde a la media muestral de los datos que sean más cercanos al componente.

En síntesis, si se asume un modelo probabilístico GMM para el conjunto de datos con matrices de covarianzas fijas, idénticas y esféricas a través de todos los clusters y se toma el límite de la varianza de los clusters para σ muy pequeñas el algoritmo EM equivale al algoritmo k-means.

2.1.2.2. K-medoids y CLARA Clustering

Existe un algoritmo con un enfoque similar a K-means que, si bien no se abordará a detalle, es conveniente tener en cuenta, este es el algoritmo k-medoids.

k-medoids tiene una estructura muy parecida a la de k-means solo que en este caso el centroide de cada cluster corresponde a un integrante del mismo y no la media de sus integrantes como en k-means, estos centroides reciben el nombre de medoids. El medoid de cada cluster será definido como el punto cuya distancia promedio al resto de puntos del cluster es la mínima, al ser un miembro del cluster el centroide y no la media de sus miembros se tiene menor sensibilidad a outliers con este enfoque.

Al igual que en k-means, en k-medoids es necesario predefinir el número de clusters a construir. Del mismo modo, se tiene un algoritmo estándar que es el más usado al aplicar k-means, este es el algoritmo PAM (Partitioning Around Medoids, Kaufman & Rousseeuw, 1990).

La idea en que se fundamenta el algoritmo PAM es la búsqueda de los k puntos más representativos del conjunto de datos. Una vez seleccionados, los clusters se construyen asignando cada dato al cluster representado por el medoid más cercano. Posteriormente se asigna a otro integrante del cluster como un nuevo medoid y se calcula la suma de las distancias entre todos los miembros del cluster y este nuevo medoid, este proceso es iterado hasta hallar la asignación que minimice esta suma de distancias. Para este algoritmo suelen emplearse la distancia de Manhattan y la Euclidiana, en caso de presentarse outliers la distancia de Manhattan deberá tener un mejor desempeño.

En resumen, la secuencia del algoritmo es la siguiente:

1. Definir los medoids iniciales
2. Calcular la matriz de distancias
3. Asignar cada observación al cluster que corresponda al medoid más cercano
4. Buscar en cada cluster si existe algún miembro del mismo cuya distancia promedio con el resto sea menor, en dado caso, definirlo como el nuevo mediod del cluster. En caso de haber más de un punto con una distancia promedio menor, elegir el que tenga la menor.

5. Si al menos un medoid ha cambiado, iterar los pasos 3 y 4, en caso contrario finalizar.

Finalmente, abordando las desventajas del algoritmo se tiene que será necesario que el analista defina el número de clusters a construir y que por la estructura del algoritmo, se requiere un mayor recurso computacional que con otras alternativas. Ello cobra mayor relevancia al trabajar con grandes cantidades de datos.

Es justo en este tipo de casos que algoritmo CLARA - Clustering Large Aplications cobra relevancia. Éste puede pensarse como una extensión de k-medoids. La idea básica de este algoritmo se puede simplificar como la aplicación del algoritmo PAM a una muestra del universo de datos a clasificar y extender la clasificación obtenida a todo este universo. De este modo se obtienen las ventajas de PAM clustering sin el consumo de los recursos computacionales que el mismo requiere.

La estructura del algoritmo es la siguiente:

1. Obtener n muestras aleatorias de tamaño fijo del conjunto de datos a clasificar
2. Aplicar el algoritmo PAM a cada una de las muestras obtenidas, cada subconjunto devolverá k medoids.
3. Dado el subconjunto i, con $i \in [i, n]$ y sus k_i se tendrán sus correspondientes k medoids resultantes, asignar cada punto del conjunto total de datos a su medoid más cercano. Este proceso se hará para cada uno de los n conjuntos de medoids obtenidos.
4. Calcular la suma (o promedio) de distancias entre el total de observaciones y su medoid más cercano para los n conjuntos de medoids.
5. El conjunto de medoids óptimo será el que minimice esta distancia total.

2.1.3. Clustering jerárquico

Como se analizó anteriormente, en el clustering de partición es necesario especificar el número de clusters a construir lo cual no siempre resulta obvio y no hay un método infalible que indique el número óptimo si este se desconoce, adicionalmente dado que se parte de una selección aleatoria de centroides, el resultado final puede variar en mayor o menor medida dependiendo de esta asignación inicial. En este contexto resulta de interés conocer el clustering jerárquico el cual no requiere predefinir este número, sin embargo, si requiere definir la medida con la que se mida la similitud o diferencia entre objetos y clases.

Tal como su nombre lo sugiere, con este tipo de algoritmos se obtienen representaciones jerárquicas de los clusters en los que cada nivel está conformado por

agrupaciones de clusters del nivel inferior. Esto puede ilustrarse más claramente con diagramas de árbol llamados dendogramas. Véase la Figura 2.1.

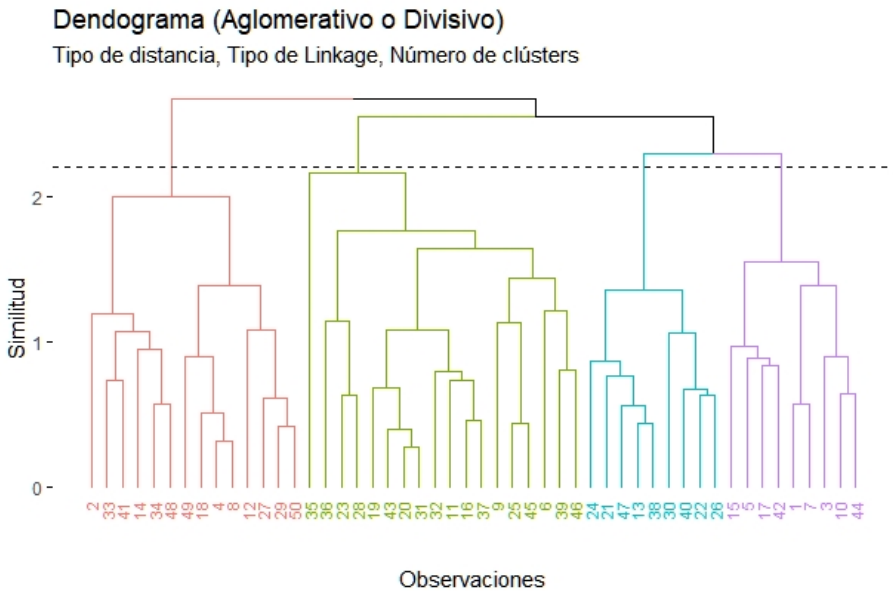


Figura 2.1: Ejemplo de dendograma

Esta representación deja a elección del analista el nivel jerárquico más adecuado para la clasificación, dependiendo del nivel elegido se obtendrá determinado número de clusters.

Bajo estos preceptos, se tienen dos enfoques

1. Clustering Aglomerativo: Se parte suponiendo que cada objeto constituye un cluster per se, posteriormente se identifican similitudes entre clusters para ir formando agrupaciones cada vez más grandes.
2. Clustering Divisivo: Se parte del supuesto de que todos los objetos pertenecen a un único cluster, posteriormente se identifican agrupaciones internas del mismo y se generan nuevos clusters partiendo de ellas, así sucesivamente hasta obtener clusters más pequeños y homogeneos.

A continuación se abordará con más detalle cada enfoque.

2.1.3.1. Clustering Aglomerativo

Como ya se explicó en el resumen, los algoritmos de clustering aglomerativo parten del supuesto de que cada objeto es un cluster y en cada paso agrupan a

los más parecidos entre si para formar un cluster más grande. Dada la estructura del algoritmo, será necesario definir la métrica con la que el algoritmo medirá esta similitud entre clusters y entre objetos, por lo que ya no será suficiente el concepto de medidas anteriormente expuestas que se aplicaban únicamente para objetos, es decir que habrá que extender este concepto para medir similitud entre grupos, a continuación las medidas o *linkages* más populares:

Sean A y B dos clusters. Si $d_{ii'}$ es el conjunto de distancias (ya sea la medida euclidiana, de Manhattan, etc.) entre cada punto $i \in A$ y cada punto $i' \in B$.

1. **Complete Linkage**: Define la distancia entre clusters como la distancia entre sus miembros más lejanos.

$$d_{CL}(A, B) = max\{d_{ii'} | i \in A, i' \in B\} \tag{2.4}$$

2. **Single Linkage**: Define la distancia entre clusters como la distancia entre sus miembros más cercanos.

$$d_{SL}(A, B) = min\{d_{ii'} | i \in A, i' \in B\} \tag{2.5}$$

3. **Average Linkage**: Se define como el promedio de distancias entre todos los puntos de A y todos los de B. Es decir, sean N_A y N_B el número de puntos pertenecientes a A y B respectivamente:

$$d_{AL}(A, B) = \frac{1}{N_A N_B} \sum_{i \in A} \sum_{i' \in B} d_{ii'}$$

4. **Centroid Linkeage**: Se define la distancia entre clusters como la distancia entre sus respectivos centroides (definidos como la media de sus integrantes).

5. **Ward Linkeage**: La selección de clusters a fusionar será definida por el par de clusters que, al fusionarse, minimicen la suma total de varianza intra-cluster.

La distancia y el linkeage a utilizar dependerá de la estructura de los datos, no existe una combinación de medidas que sea infalible para cualquier conjunto de datos y en ocasiones una servirá mejor para describir la similutud entre datos y agrupaciones de datos que otra aunque existen médotos que ayudan a sustentar esta decisión. Una vez definidas las métricar a utilizar se procede a aplicar el algoritmo cuya estructura es la siguiene:

1. Calcular la similitud entre cada par de puntos a clasificar

2. Emplear dicha medida de similitud para agrupar los puntos más similares en clusters.

3. Seleccionar una función linkage a conveniencia y, con ella, agrupar los clusters más similares generando así una estructura agrupada de forma jerárquica para los datos hasta llegar a un cluster que englobe a todos los datos.

4. Elegir a conveniencia el punto de corte del dendograma resultante, ello generará un determinado número de clusters.

Los clusters resultantes pueden variar dependiendo de la medida y función linkage que se utilicen.

2.1.3.2. Clustering Divisivo

Un algoritmo muy extendido con el enfoque divisivo es el propuesto por Macnaughton Smith (1965) [12] . Su estructura es la siguiente:

1. Supóngase A como el cluster inicial, es decir, el que contiene todos los puntos a clasificar.

2. Se calcula (por medida euclidiana, Manhattan, etc.) la matriz de distancias entre todos los puntos del cluster y se identifica al que tenga una distancia promedio mayor, este punto será el primer integrante de un nuevo cluster B.

3. Se identifican los puntos de A que tengan una distancia mayor a los puntos restantes de A que a los del nuevo cluster B y se transfieren a él.

4. Se itera el mismo criterio con los puntos restantes de A hasta que no quede ningún punto en él cuya distancia promedio sea mayor a los puntos de A que a los de B. Resultando 2 nuevos y más pequeños clusters.

5. Repetir el procedimiento con cada uno de los nuevos clusters y sus subsecuentes particiones, estructurando así los datos en una estructura jerárquica hasta llegar a una segmentación atómica de los datos en los que cada punto represente un cluster per se.

6. Elegir a conveniencia el punto de corte del dendograma resultante, ello generará un determinado número de clusters.

Tal como se puede observar, en algoritmos de clustering divisivo no es necesaria una función linkage dado que no se mide distancia entre grupos si no únicamente entre puntos.

Una ventaja que tiene este algoritmo frente a k-means es que si bien, los dos segmentan el universo de datos en grupos más pequeños, en este caso no es necesario predefinir el número de ellos y, adicionalmente, no se depende de una asignación inicial arbitraria de clusters.

Finalmente, una ventaja del clustering divisivo frente al aglomerativo es que se desempeña mejor identificando clusters grandes, es decir, cuando es de interés segmentar los datos en un número de clusters reducido.

2.2. Modelos predictivos

En el ámbito práctico, el análisis predictivo es una de las herramientas más valoradas dada la inmensidad de aplicaciones que puede tener la disciplina y el provecho que una empresa puede obtener de ellas. Por ejemplo, en el sector financiero es indispensable el contar con un modelo de score que evalúe la probabilidad de que un solicitante incumpla con el pago de su crédito y con base en ello se define si se le otorga o no o a qué tasa de interés. Existen también modelos para la prevención de fraudes, para la estimación de portafolios, etc.

Existen dos tipos de modelos predictivos: modelos de clasificación y de regresión. Los modelos de regresión con mayor antigüedad dependen de ajustar los datos a alguna distribución de probabilidad, sin embargo en décadas recientes ha surgido un enfoque distinto en el que se puede prescindir de este ajuste y puede pronosticarse cierta variable de los datos observando las demás, un modelo con este enfoque es el de K-Nearest neighbors (k-vecinos más cercanos).

2.2.1. k-nearest neighbors

El refrán popular *dime con quién andas y te diré quién eres* define a la perfección la lógica de este modelo pues muy a grandes razgos consiste en definir agrupaciones de datos, observar patrones de comportamiento en estas agrupaciones y pronosticar determinado comportamiento en nuevas observaciones dependiendo de la agrupación más cercana a ella y su probabilidad de pertenecer a esa agrupación.

Para ilustrar la estructura del algoritmo, piénsese en el conjunto de datos como pares ordenados $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ en donde x_i (con $i \in \{1, , N\}$) denominado *input* es un vector de D dimensiones llamadas atributos o características y y_i llamado *output* o *etiqueta* corresponde a la variable objetivo, es decir, la variable a pronosticar.

Los atributos pueden estar compuestos por variables continuas o discretas y el output en caso de ser continuo reducirá el pronóstico a una regresión, en caso de ser discreto este pronóstico será de clasificación.

Partiendo de un conjunto de datos de entrenamiento del que se conoce tanto el input como el output se tiene como objetivo pronosticar el output de un nuevo conjunto de datos del que únicamente se conoce el input. Con k-nearest neighbors ello se logra *observando* el output que tienen los k datos del conjunto de entrenamiento más cercanos a cada dato del nuevo conjunto y asignado un

output a estos últimos basándose en el output de estos k vecinos más cercanos. Si se trata de clasificación, esta asignación será igual al output de la mayoría de estos k vecinos y en caso de tratarse de regresión, será el promedio de estos outputs.

Para la ejecución del algoritmo será necesario determinar previamente el valor de k, es decir, el número de vecinos a observar que se considere adecuado para la asignación, así como la métrica de distancia para determinar cuales son los k datos más similares.

Una vez definido el conjunto de datos de entrenamiento e ingresado un nuevo punto a clasificar el funcionamiento del algoritmo es el siguiente:

1. Definir la métrica de distancia a utilizar y el valor de k, ello dependerá de la estructura de los datos tal como se abordó en 2.1.1. Los datos deben ser estandarizados puesto que escalas distintas entre variables podrían alterar el criterio de similitud entre puntos.
2. Calcular la matriz de distancias entre el punto a clasificar y cada uno de los puntos del conjunto de entrenamiento.
3. Ordenar las distancias para elegir los k puntos más cercanos.
4. Etiquetar el nuevo dato con la misma etiqueta de la mayoría de los vecinos (si se presenta un empate, se etiqueta al azar) en caso de requerirse clasificación. Para regresión se asigna el promedio de las etiquetas de los K vecinos.

Como puede observarse, k-means es un algoritmo de fácil implementación. Sin embargo, dado que cada nuevo dato debe compararse con el conjunto de entrenamiento se requerirán altos recursos computacionales para grandes cantidades de datos.

2.3. Reducción de dimensión

Un problema cada vez más habitual en la actualidad no es la falta de información si no el exceso de ella tanto en cantidad de registros como en número de variables. Si bien los recursos computacionales se han democratizado en la actualidad, si se tiene una base de datos muy extensa no cualquier computadora será capaz de ejecutar algoritmos con ella, en especial si se trata de algoritmos complejos. Adicionalmente, el realizar un estudio visual de la información se torna complicado cuando esta consta de múltiples variables.

Es bajo este contexto que surge la necesidad de alguna técnica destinada a reducir la dimensión de los datos. Existe más de una que tiene esta finalidad, una muy extendida es el Análisis de Componentes Principales (PCA por sus

siglas en inglés) basada y limitada a combinaciones lineales de los datos. Otra alternativa es t-Distributed stochastic Neighbor Embedding (t-SNE) propuesto por Hinton y Van der Maaten en 2008 [4] que al ser un método no lineal tiene un mejor desempeño en diversos escenarios.

Para ejemplificar, piénsese en un conjunto de datos tridimensionales representado los el gráfico 2.2 y supóngase que es menester el poder representar estos datos de forma bidimensional ya sea para eficientar recursos computacionales al querer realizar cálculos con ellos o simplemente para tener una comprensión más amplia de los mismos.

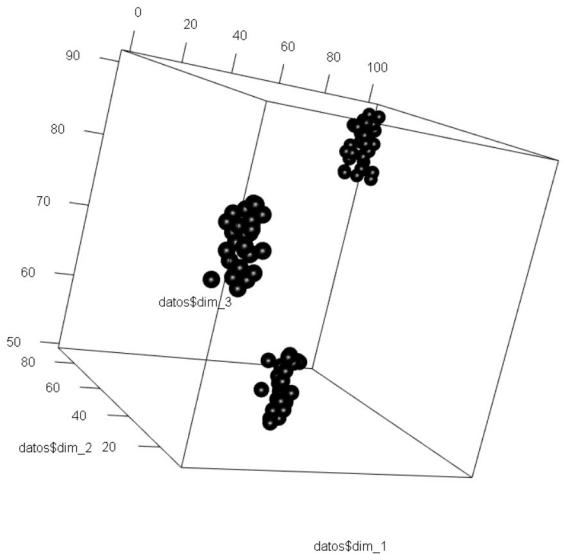


Figura 2.2: Datos originales en 3 dimensiones

En la figura 2.3 se tiene esta representación mediante el método t-SNE.

Mientras que en la figura 2.4 se tiene otra representación de los datos mediante el método PCA.

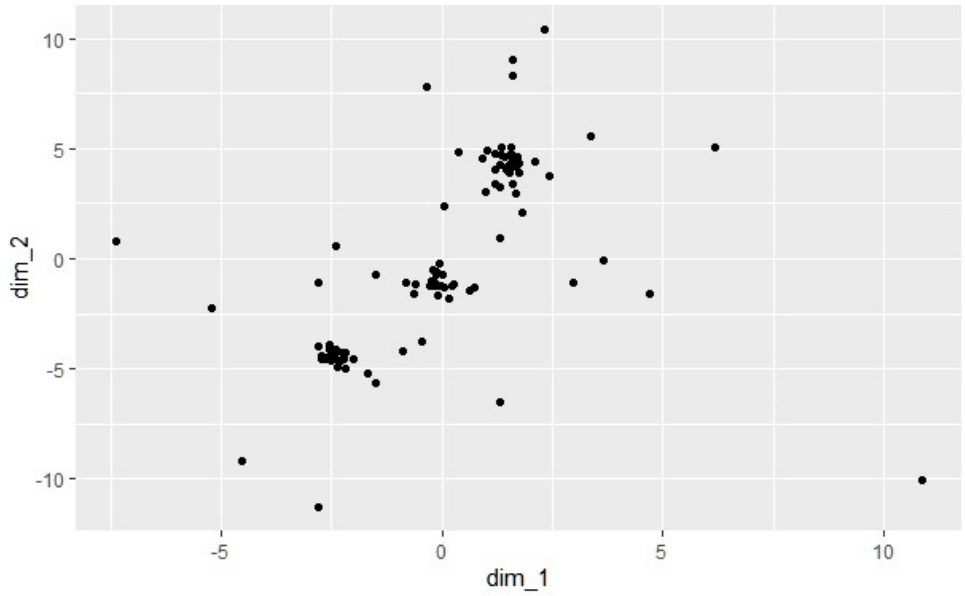


Figura 2.3: Reducción a dos dimensiones con t-SNE

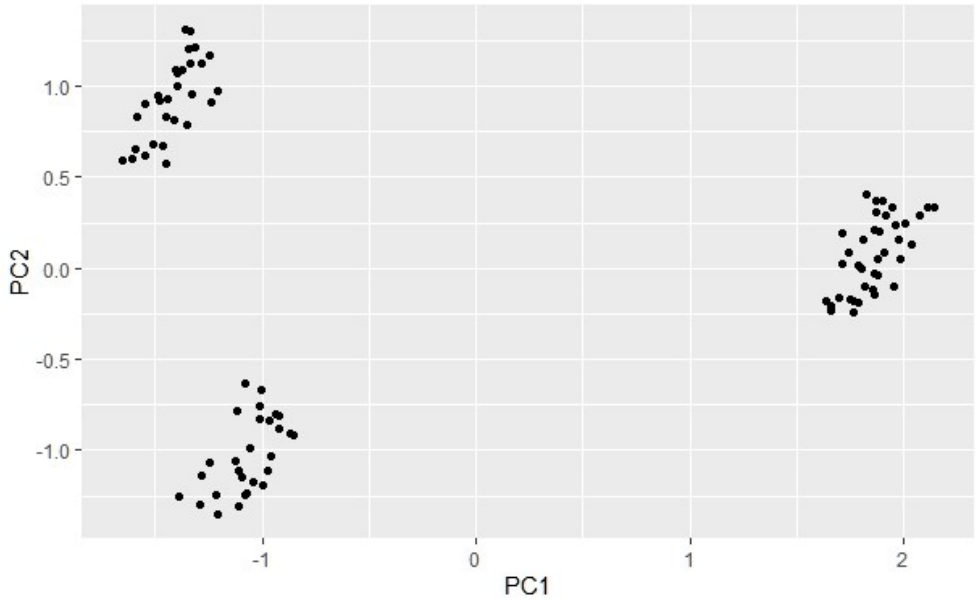


Figura 2.4: Reducción a dos dimensiones con PCA

2.3.1. TSNE

t-SNE técinca surge como sucesora de la técnica Stochastic Neighbor Embedding o *SNE* (Hinton & Roweis, 2002), estando t-SNE mejor optimizada que su predecesora y "produce visualizaciones significativamente mejores" [4]. Con t-SNE es posible revelar estructuras en diversas escalas, finalidad que es muy útil al trabajar con datos de multiples dimensiones.

La idea base de t-SNE y su predecesor es la de medir la similitud entre puntos por probabilidades condicionales en vez de usar la métrica de distanca Euclidiana, es decir, la similitud entre puntos está dada por la probabilidad condicional $p_{j|i}$ en la que los vecinos x_j de x_i serán seleccionados mediante la suposición de la existencia de una distribución Gaussiana centrada en x_i. Para los puntos cercanos a x_i será alto el valor de $p_{j|i}$ y mientras más lejanos sean este indicador irá disminuyendo de forma considerable dada la estructura de las distribuciones Gaussianas hasta alcanzar valores ínfimos. Esta probabilidad condicional $p_{j|i}$ está dada por

$$p_{j|i} = \frac{e^{\frac{-\|x_i - x_j\|^2}{2\sigma_i^2}}}{\sum_{k \neq i} e^{\frac{-\|x_i - x_k\|^2}{2\sigma_i^2}}}$$

donde la distribución Gaussiana supuesta $p_{i|i} = 0$ centrada en x_i tiene varianza σ_i.

Dadas las contrapartes de x_i y x_j de menor dimensión y_i y_j es posible calcular una probabilidad condicional similar denotada por $q_{j|i}$, asignando a la varianza de la distruición supuesta en este caso como $\frac{1}{\sqrt{2}}$, la similitud entre puntos será

$$p_{j|i} = \frac{e^{-\|y_i - y_j\|^2}}{\sum_{k \neq i} e^{-\|y_i - y_k\|^2}}$$

donde nuevamente se define $q_{i|i} = 0$. Para que los puntos y_i y y_j representen de forma precisa el espacio de mayor dimensión, las probabilidades condicionales $p_{j|i}$ y $q_{j|i}$ deben ser iguales, es por ello que se busca el espacio de menor dimensión que minimice la diferencia entre $p_{j|i}$ y $q_{j|i}$. El método SNE minimiza esta diferencia mediante un método de gradiente descendente aplicado a una métrica de fidelidad llamada divergencia de Kullback-Leibler cuya función costo está dada por

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}$$

El parámetro que resta por definir es la varianza σ_i de la distribución Gaussiana supuesta con centro en cada punto x_i, el problema con este parámetro es que no

existe un valor para él que sea óptimo para cualquier conjunto de datos, si los puntos son muy cercanos entre si, el valor de σ_i debería ser pequeño y de forma inversa, si los datos están dispersos se esperaría un valor grande para σ_i. SNE resuelve esta incógnita hallando un σ_i que produzca una P_i con una *perplexity* fija y estimada por el analista. Esta *perplexity* está definida como

$$Perp(P_i) = 2^{H(P_i)}$$

en donde

$$H(P_i) = -\sum_j p_{j|i} log_2 p_{j|i}$$

el valor típico de esta variable se ubica entre 5 y 50.

El punto que distingue el método t-SNE de su predecesor es que para convertir las distancias del espacio de menor dimensión en probabilidades supone una distribución t-Student con un grado de libertad, la cual posee colas con mayor peso que la distribución Gaussiana. Ello permite modular las distancias del espacio de dimensión superior en distancias de mayor magnitud de forma fiable en el espacio de dimensión inferior dispersando así los puntos que en el espacio de dimensión superior presentan disimilitudes mínimas, lo que facilita el reconocimiento de patrones. Otra ventaja de esta distribución, es que es más rápido el evaluar densidades de probabilidad bajo ella al no integrar funciones exponenciales como en el caso de la distribución Gaussiana.

Bajo este supuesto base, t-SNE minimiza la divergencia Kullback-Leibler entre las probabilidad conjuntas p_{ij} en el espacio de dimensión superior y las probabilidades conjuntas q_{ij} en el espacio de dimensión inferior. Los valores p_{ij} son definidos como las probabilidades condicionales simétricas

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

Mientras que los q_{ij} están definidos por las medias de la distribución t-Student con un grado de libertad

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}} \tag{2.6}$$

Se define $p_{ii} = q_{ii} = 0$ y la divergencia Kullback-Leibler entre las disribuciones P y Q queda definida por

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}}$$

$$= \sum_i \sum_j p_{ij} log p_{ij} - p_{ij} log q_{ij} \tag{2.7}$$

Haciendo un cambio de variable para reducir notación

$$d_{ij} = \|y_i - y_j\|$$

$$Z = \sum_{k \neq l} (1 + d_{kl}^2)^{-1}$$

De lo cual, el gradiente de C con respecto a y_i está dado por

$$\frac{\partial C}{\partial y_i} = \sum_j (\frac{\partial C}{\partial d_{ij}} + \frac{\partial C}{\partial d_{ji}})(y_i - y_j)$$

$$= 2 \sum_j \frac{\partial C}{\partial d_{ij}} (y_i - y_j) \tag{2.8}$$

Es posible obtener el gradiente $\frac{\partial C}{\partial d_{ij}}$ partiendo de la ecuación 2.7 haciendo la observación de que la primera parte de la expresión es una constante, de lo cual

$$\begin{aligned}
\frac{\partial C}{\partial d_{ij}} &= -\sum_{k \neq l} p_{kl} \frac{\partial (log q_{kl})}{\partial d_{ij}} \\
&= -\sum_{k \neq l} p_{kl} \frac{\partial (log q_{kl} Z - log Z)}{\partial d_{ij}} \\
&= -\sum_{k \neq l} p_{kl} (\frac{1}{q_{kl} Z} \frac{\partial ((1 + d_{kl}^2)^{-1})}{\partial d_{ij}} - \frac{1}{Z} \frac{\partial Z}{\partial d_{ij}})
\end{aligned}$$

Nótese que el gradiente $\frac{\partial ((1+d_{kl}^2)^{-1})}{\partial d_{ij}}$ es distinto de cero únicamente cuando $k = i$ y $l = j$, de lo cual $\frac{\partial C}{\partial d_{ij}}$ queda dado por

$$\frac{\partial C}{\partial d_{ij}} = 2 \frac{p_{ij}}{q_{ij} Z} (1 + d_{ij}^2)^{-2} - 2 \sum_{k \neq l} p_{kl} \frac{(1 + d_{ij}^2)^{-2}}{Z}$$

Dado que $\sum_{k \neq l} p_{kl} = 1$, la expresión anterior se simplifica como

$$\frac{\partial C}{\partial d_{ij}} = 2 p_{ij} (1 + d_{ij}^2)^{-1} - 2 q_{ij} (1 + d_{ij}^2)^{-1}$$

$$= 2(p_{ij} - q_{ij})(1 + d_{ij}^2)^{-1}$$

Finalmente, se sustituye esté término en la expresión 2.8, obteniendo

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(1 + \|y_i - y_j\|^2)^{-1}(y_i - y_j) \tag{2.9}$$

En resumen, la estructura del algoritmo es la siguiente:

1. Dado un conjunto de datos $X = \{x_1, x_2, \ldots, x_n\}$ se define el *perplexity* $Perp$ y el número de iteraciones T
2. Se calculan los valores $p_{j|i}$ haciendo uso del valor $Perp$ obteniendo $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$. Se propone la muestra solución inicial $\gamma^{(0)} = \{y_1, \ldots, y_n\}$ de $N(0, 10^{-4}I)$
3. Haciendo uso de la expresión 2.6 se calculan los q_{ij} y el correspondiente gradiente $\frac{\partial C}{\partial \gamma}$ aplicando la expresión 2.9. Se asigna $\gamma^{(t)} = \gamma^{(t-1)} + \eta \frac{\partial C}{\partial \gamma} + \alpha(t)(\gamma^{(t-1)} - \gamma^{(t-2)})$ para $1 \leq t \leq T$ donde η es la *tasa de aprendizaje* y $\alpha(t)$ el *momentum*. Con ello se ha obtenido el espacio de dimensión reducida.

2.3.1.1. Aproximación Barnes-Hut

Si bien se han mostrado los útiles alcances del método t-SNE, dada su estructura se tiene que su complejidad escala de forma cuadrática al número de datos en el conjunto analizado, es por ello que su aplicación está limitada a usuarios con altos recursos computacionales o a conjuntos que no rebasen algunos miles de datos. Es por ello que surge la necesidad de hallar alguna forma de obtener los resultados de t-SNE o una aproximación cercana de ellos de una forma mejor optimizada.

Es posible aproximar (Barnes & Hut, 1986) las medidas de similitud p_{ij} sin mayor afectación al resultado final calculando hallando los $\lfloor 3u \rfloor$ vecinos más cercanos para cada uno de los N datos del conjunto analizado (donde u es el *perplexity* de las distribuciones condicionales) y redefiniendo la similitud entre pares, p_{ij} como:

$$p_{i|j} = \begin{cases} \frac{exp(-d(x_i, x_j)^2 / 2\sigma_i^2)}{\sum_{k \in N_i} exp(-d(x_i, x_k)^2 / 2\sigma_i^2)} & sij \in N_i \\ 0 & Cualquierotrocaso \end{cases}$$

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \tag{2.10}$$

en donde N_i representa el conjunto de los $\lfloor 3u \rfloor$ vecinos más cercanos de x_i, σ_i se establece a modo de satisfacer la *perplexity* predefinida u. Los conjuntos N_i serán definidos para todo el conjunto de datos, posteriormente se calculan los valores p_{ij} con las expresiones de arriba.

Posteriormente, la forma en que éste método aproxima el gradiente de t-SNE es pensándolo como la suma de 2 *fuerzas*, es decir, reescribiendo la expresión 2.9 como

$$\frac{\partial C}{\partial y_i} = 4(F_{attr} + F_{rep}) = 4(\sum_{j \neq i} p_{ij} q_{ij} Z(y_i - y_j) - \sum_{j \neq i} q_{ij}^2 Z(y_i - y_j))$$

donde se denota como F_{attr} la suma de las *fuerzas de atracción* representada por la suma izquierda de la ecuación anterior y F_{rep} r-denota la suma de las *fuerzas de repulsión* representada por la suma derecha de la ecuación anterior y

$$Z = \sum_{k \neq l} (1 + \|y_i - y_j\|^2)^{-1}$$

Si bien, el cálculo de F_{attr} puede ser calculado de forma relativamente simple mediante 2.10, el calculo de F_{rep} sigue representando una limitante por su complejidad y es en este fragmento en el que este método enfoca la aproximación.

Si se consideran tres puntos y_i, y_j y y_k tales que $\|y_i - y_j\| \approx \|y_i - y_k\| \gg \|y_j - y_k\|$ la influencia de y_j y y_k sobre F_{rep} será casi la misma. El algoritmo Barnes-Hut toma ventaja de esta observación agrupando los datos y eligiendo alguno de cada agrupación como representante para *resumir* la contribución de los puntos de este grupo sobre F_{rep}.

Ello lo logra definiendo un *quadtree*, un árbol en el que cada nodo representa una *celda* rectangular con determinados ancho y alto. Los nodos que no son hojas se particionan en cuatro celdas más pequeñas, los nodos que no se particionan u *hojas* representan celdas que contienen al menos un punto del conjunto y el nodo inicial representa la celda que contiene todos los datos del conjunto total. En cada nodo se almacenan el *centro de masa* y_{cell} de los puntos contenidos en la celda correspondiente y el número de puntos N_{cell} contenidos en dicha celda.

La idea principal es que si una celda es lo suficientemente pequeña y lejana al punto y_i, las contribuciones $q_{ij}^2 Z(y_i - y_j)$ a F_{rep} serán muy similares para todos los puntos y_j pertenecientes a dicha celda. De este modo se aproxima dichas contribuciones mediante $-N_{cell} q_{i,cell}^2 Z(y_i - y_{cell})$ donde y_{cell} y N_{cell} están dados como se definió anteriormente y $q_{i,cell} Z = (1 + \|y_i - y_{cell}\|^2)^{-1}$.

Inicialmente se aproxima $F_{rep} Z = -q_{ij}^2 Z^2 (y_i - y_j)$ evaluando en cada nodo del árbol si es posible *resumir* del modo expuesto anteriormente todos los puntos pertenecientes a la celda correspondiente. Del mismo modo se construye una

estimación para $Z = \sum_{i \neq j}(1 + \|y_i - y_j\|^2)^{-1}$. Ambas estimaciones brindarán en conjunto una aproximación para F_{rep} como $F_{rep}\frac{F_{rep}Z}{Z}$.

El método propuesto por Barnes y Hut brinda un criterio para decidir cuándo puede resumirse una celda, este criterio compara la distancia entre la celda y el punto objetivo con el tamaño de la celda, es decir

$$\frac{r_{cell}}{\|y_i - y_{cell}\|^2} < \theta$$

en donde r_{cell} representa la longitud de la diagonal de la celda en cuestión y θ es un valor predefinido de acuerdo con la precisión deseada (valores grandes de θ brindarán una mayor velocidad de cómputo pero con resultados más burdos, valores pequeños serán más precisos pero con un tiempo de cómputo mayor).

De este modo se obtiene una aproximación fiable de los resultados del método t-SNE sin la complejidad del mismo y los recursos computacionales que ello demanda.

Capítulo 3

Aplicación

3.1. Introducción

En cierta institución financiera **dedicada al préstamo de créditos grupales**, una de las figuras principales del negocio es el *asesor de crédito* cuya función es la venta de créditos y su correspondiente gestión (cobranza, retención, entre otros). Por tal motivo, un interés natural para la empresa en cuestión es elevar la productividad del asesor de crédito.

En general, para una institución financiera existe gran interés en la cantidad de indicadores orientados al tamaño o crecimiento de la cartera así como de su calidad y gestión, muchos de ellos se definirán más adelante.

3.2. Planteamiento del problema

Un objetivo es definir una métrica adecuada para evaluar el desempeño de los empleados y clasificarlos con base en ella. Dado que el ingreso bruto que un asesor aporta a la empresa con su trabajo es el monto de intereses cobrados se propone usar esta variable como métrica de productividad.

Usualmente suele definirse la productividad de los empleados como el cociente del capital que aportan para la utilidad de la empresa entre el capital que la empresa invierte en ellos. En este caso no se ocupa esta definición dado que la empresa otorga comisiones con base en sus métricas de interés por lo que un asesor que perciba montos altos en comisiones pudiera calificarse como poco productivo a pesar de que sus indicadores sean óptimos y el análisis pasaría a colocar en la cima de la clasificación a los asesores que no necesariamente tuviesen los mejores indicadores pero cobraran poco estancando así el crecimiento y metas de la empresa.

Una vez definida la métrica, se desea clasificar a los empleados en *niveles* de acuerdo con su desempeño y, a su vez, estimar cual será el *desempeño* o nivel del asesor en un futuro. Con ello se busca atacar la siguiente problemática:

- Supóngase un empleado que durante toda su estancia en la empresa, no ha logrado sobrepasar cierto umbral de productividad pero en determinado mes, por una combinación de factores, logra superarlo. Una clasificación simple, que determine el *nivel* del empleado basándose únicamente en el desempeño más reciente del empleado provocaría que al asesor en cuestión se le otorgasen beneficios pensados únicamente para el personal más productivo. Dado que existen beneficios que no es posible retirar más adelante (aumentos de sueldo, ascensos, entre otros) esto supondría una pérdida para la empresa si el asesor en cuestión regresa a los niveles deficientes de productividad que había tenido históricamente.

- Supóngase ahora el caso de un asesor que siempre se ubica en los niveles más altos de productividad, sin embargo en determinado mes su productividad se cae. Es de interés para la empresa detectar el origen del problema del asesor (en términos de las variables del negocio) y con ello darle pauta al asesor para que enfoque sus esfuerzos en el área de mejora y así retome su nivel de desempeño y que no sea catalogado como empleado poco productivo.

- Finalmente supóngase el caso de dos asesores de nuevo ingreso. Al final de determinado periodo se tiene que ambos se mantienen en los niveles más bajos de productividad, sin embargo uno de ellos ha tenido cierto crecimiento y la tendencia muestra que al cabo de un periodo corto de tiempo debería alcanzar niveles aceptables mientras que para el otro no se estima que tenga mejora en mediano plazo. Es de interés para la empresa el ser capaz de detectar dichas tendencias para apreciar estas diferencias más sutiles entre empleados y enfocar de forma óptima recursos para programas de capacitación, beneficios, etc.

Abordando el problema desde un enfoque estadístico, lo que se necesita es un modelo compuesto, inicialmente se requiere analizar qué variables son las que más influyen en la productividad del empleado, o bien, concentrar la información que aporta el total de variables y, por otro lado, se debe estimar la productividad del empleado en un futuro a partir de estas variables.

3.3. Exploración de los datos

Una vez abordado el problema, se procede a obtener los datos a usar, estructurarlos y hacer una exploración preliminar de los mismos.

Por medio de las bases de datos de la empresa en cuestión, se tiene un listado de variables correspondientes a diversos indicadores de cada asesor a la fecha de cierre de cada mes desde enero del 2018 a Julio del 2020. A continuación un glosario de los mismos, abordando conceptos del negocio para su definición:

- **mes**: Indica el mes al que corresponde la información en formato aaaamm. Cada empleado tendrá un registro en cada mes que haya estado activo.
- **Fecha_información**: Indica el día al que corresponde la información, en este caso corresponderá al último día de la variable *mes*
- **no_empleado**: Id del empleado al que corresponden los indicadores del mes. Puede defiirse el id de un registro como una llave compuesta por el núero de empleado y el mes de registro.
- **Clientes_nuevos**: Créditos otorgados durante el mes, contabilizados por número de clientes que no habían tenido un crédito previamente con la empresa.
- **Clientes_renovados**: Créditos vendidos durante el mes, contabilizados por número de clientes que ya habían tenido un crédito previamente con la empresa, el plazo de su crédito previo vencía en el mes.
- **Clientes_anticipados**: Créditos vendidos durante el mes, contabilizados por número de clientes que ya habían tenido un crédito previamente con la empresa, el plazo de su crédito previo vencía durante el mes posterior al registro.
- **Intereses**: Monto monetario que corresponde a los intereses cobrados durante el mes.
- **ODV**: Monto monetario que corresponde a la cartera con 0 días de atraso. (La cartera es el monto monetario que consta de todo el capital en préstamo asignado al empleado en cuestión para su gestión. Por definición, los créditos que constituyen la cartera tienen menos de 180 días de atraso en pago).
- **Buckets**: Conjunto de variables expresadas en montos monetarios que segmentan la cartera del empleado de acuerdo con sus días de atraso; P1_14 cartera de créditos que tienen entre 1 y 14 días de atraso, P15_29 entre 15 y 29, P30_44 entre 30 y 44, P45_59 entre 45 y 59, P60 entre 60 y 89, P90 entre 90 y 119, P120 entre 120 y 149 y P150 entre 150 y 179
- **Atraso_promedio**: Valor numérico que resulta de promediar los días de atraso de todos los grupos gestionados por el empleado.
- **Ciclo_promedio**: Resultado numérico de promediar el número de ciclo de todos los grupos gestionados por el empleado. Entiéndase por ciclo al número de créditos que se le han otorgado al grupo, por ejemplo: un grupo

nuevo será ciclo 1, un grupo con ciclo 3 será aquel al que se le ha otorgado un crédito después de haber liquidado 2 anteriormente.

- **Grupos**: Número de grupos asignados al empleado para su gestión.
- **Clientes_totales**: Cartera contabilizada por número de clientes
- **Clientes_21**: Cartera con menos de 21 días de atraso, contabilizada por número de clientes.
- **Tasa_promedio**: Promedio de las tasas otorgadas a los créditos que el empleado tuvo en cartera en el mes en cuestión.
- **Antigüedad**: Tiempo que el trabajador tenía laborando en la empresa en el cierre del mes en cuestión.

Una vez obtenidos y definidos los datos, se procede a importarlos a R para su correspondiente análisis.

```
#Importación de datos
datos<-read.csv(file.choose())
```

Como ya se estableció, la clasificación de asesores se hará con base en los montos de intereses que hayan cobrado durante el mes. Se tiene que cada registro corresponde a un empleado en determinado mes por lo que se procede a calcular el monto promedio de cada empleado para así atenuar outliers.

```
#Clustering para segmentación de grupos por intereses
library(cluster)
library(factoextra)
```

```
## Loading required package: ggplot2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3
```

```
library(ggplot2)
#Se mide sobre el promedio por empleado para atenuar outliers
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

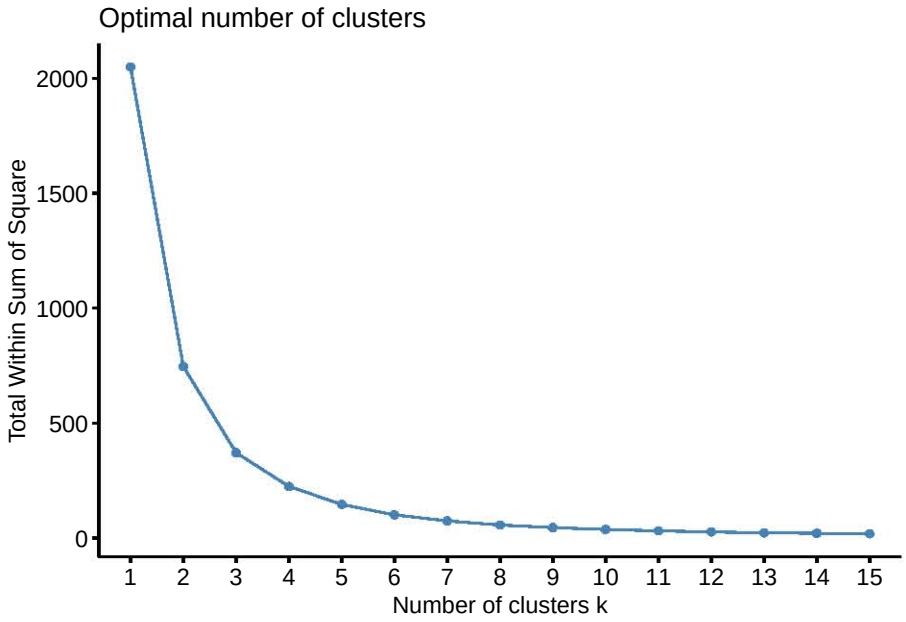
```
datos_p<-datos[!duplicated(datos$no_empleado), ][,3]
datos_p<-as.data.frame(datos_p)

for (i in 1:2050) {
  datos_p[i,2]<-mean(filter(datos,
                            no_empleado == datos_p[i,1])[,10])
}
names(datos_p)<-c("Empleado", "Int_prom")
```

Ya estructurados los datos (promedio por cada empleado) se procede a determinar los rangos óptimos de clasificación para los empleados de acuerdo con esta métrica. Ello se hará mediante clustering, empero, como ya se ha hecho mención no existe una técnica de clustering infalible.

Como primera alternativa se propone aplicar el algoritmo de k-means, para ello será necesario determinar el número de clusters a construir (dato que se desconoce, aunque en la práctica un buen intervalo sería al menos tres grupos y no más de seis). Una forma de determinar este parámetro es mediante una gráfica de inercia, en ella se brinda de forma visual la suma total de distancias internas cuadradas que resultarían dependiendo del número de clusters, resultará obvio que a mayor número de clusters menor será la suma de distancias, sin embargo lo que se busca en la gráfica es el punto óptimo o de quiebre en el que de disminuir la cantidad de clusters represente un aumento considerable en la suma de distancias y el aumentar dicha cantidad no afecte de forma importante el parámetro.

En otras palabras, lo que se buscaría de forma visual en la gráfica es un punto que se asemeje a un *codo*, es decir, en el que el cambio de pendiente sea notable y esté alejado en el eje vertical de los puntos a su izquierda y la distancia en el eje vertical con los puntos a su derecha sea casi nula.

```
x_int<-datos_p[2]
x_int<-scale(x_int)
x_int<-data.frame(x_int)
#Clustering por k-means con medida euclidiana
fviz_nbclust(x = x_int, FUNcluster = kmeans, method = "wss",
      k.max = 15, diss = get_dist(x_int, method = "euclidean"),
      nstart = 50)
```



Lo que se observa en la gráfica obtenida es un descenso progresivo en las distancias a partir de la formación de 3 clusters, lo que este descenso progresivo indica es que, aunque hay una reducción considerable con respecto a la suma de distancias obtenida al crear solo 2 clusters, el crear 4 clusters sigue disminuyendo de forma importante el parámetro y así sucesivamente por lo que no se puede afirmar que sea evidente el número óptimo de clusters para aplicar k-means.

Este resultado no representa mayor inconveniente pues, como ya se ha explicado, existen diversas alternativas. Se brinda como segunda propuesta obtener los rangos de clasificación mediante clustering jerárquico aglomerativo. Considerando que para aplicar este tipo de algorítmo es necesario definir el linkeage a aplicar, la incógnita en este caso es cuál es mejor. Una forma de establecer un criterio de decisión es empleando el coeficiente de correlación entre las distancias del dendrograma (altura de los nodos) y la matriz de distancias original. Entre más alta sea la correlación, mejor representa el dendrograma la verdadera similitud entre las observaciones. En R, la función cophenetic() calcula las distancias del clustering jerárquico, se propone comparar el average linkeage y el complete linkeage.

```
#########CLUSTERS JERÁRQUICOS######
x_int<-datos_p[2]
x_int<-scale(x_int)
x_int<-data.frame(x_int)
# Matriz de distancias euclídeas
mat_dist <- dist(x = x_int, method = "euclidean")
```

```
# Dendrogramas con linkage complete y average
hc_euclidea_complete <- hclust(d = mat_dist, method = "complete")
hc_euclidea_average  <- hclust(d = mat_dist, method = "average")
#Correlaciones con distancias
#cophenetic del dendrograma (altura de los nodos)
cor(x = mat_dist, cophenetic(hc_euclidea_complete))
```

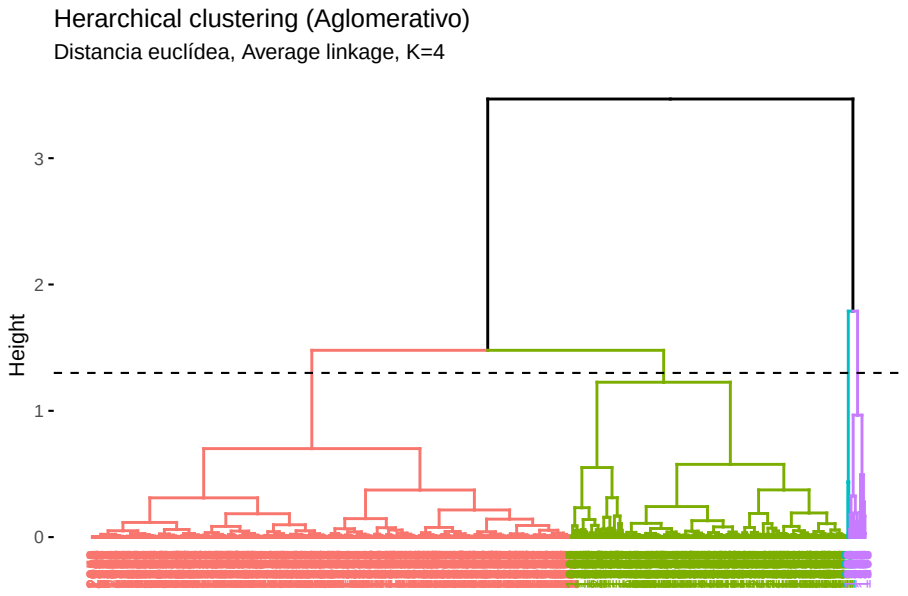
```
## [1] 0.7709246
```

```
cor(x = mat_dist, cophenetic(hc_euclidea_average))
```

```
## [1] 0.7989189
```

En este caso, el linkeage por promedio (average linkeage) arroja mejores resultados, por lo que se propone para determinar los clusters.

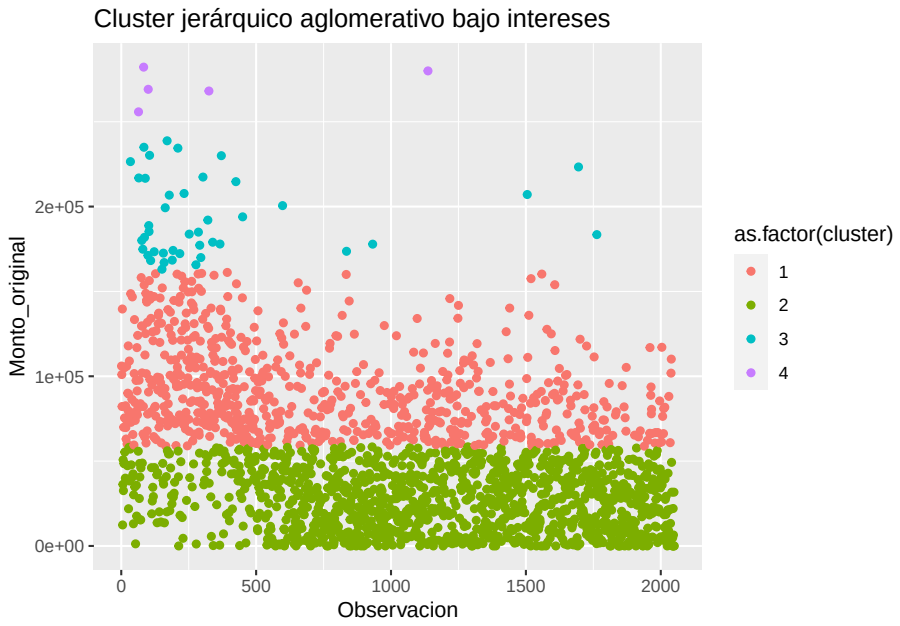
```
hc_euclidea_avg <- hclust(d = dist(x = x_int,
                  method = "euclidean"),method = "average")

fviz_dend(x = hc_euclidea_avg, k = 4, cex = 0.6) +
  geom_hline(yintercept = 1.3, linetype = "dashed") +
  labs(title = "Herarchical clustering (Aglomerativo)",
       subtitle = "Distancia euclídea, Average linkage, K=4")
```



Analizando el dendograma se propone realizar el corte a la altura de 1.3 dado

que de hacerse más arriba se estaría trabajando con 2 o 3 clusters y la mayoría de los datos se encasillarían solo en 1. En caso de hacerse más abajo comienza a crecer el número de clusters lo cual, en la práctica, no es conveniente (se debería manejar un alto número de perfiles de empleados). Al cortar el árbol en el punto propuesto, se obtienen 4 clusters visualizandose del siguiente modo con los montos originales:

```
x_int$cluster<-cutree(hc_euclidea_avg, k = 4)
x_int$Monto_original<-datos_p$Int_prom
x_int$Observacion<-1:2050
####Análisis de clusters resultantes
ggplot(x_int, aes(Observacion, Monto_original,
                  color = as.factor(cluster))) + geom_point() +
  ggtitle("Cluster jerárquico aglomerativo bajo intereses")
```



Se obtienen los rangos obtenidos y se etiqueta el total de datos de acuerdo con esta clasificación mediante la variable *Nivel*

```
#Se obtienen los máximos por cluster para etiquetar
#el universo de datos
aggregate(x_int, by = list(x_int$cluster), FUN = max)
```

```
##   Group.1  Int_prom cluster Monto_original Observacion
## 1       1 2.4695928       1      161191.93        2039
## 2       2 0.1143795       2       58437.99        2050
```

```
## 3       3 4.2476153       3      238764.02        1763
## 4       4 5.2425148       4      282169.79        1137
```

```
datos_t<-data.frame(datos$mes, datos$no_empleado, datos$Intereses)
datos_t$nivel<-ifelse(datos$Intereses < 58500, 1,
               ifelse(datos$Intereses < 160000, 2,
               ifelse(datos$Intereses < 238000, 3, 4)))
names(datos_t)<-c("Mes", "Empleado", "Intereses", "Nivel")
```

3.4. Estructuración de los datos e implementación del forecast

Con la finalidad de implementar el forecasting, la forma en la que se trabajarán los datos es obtener una matriz que contenga, para cada empleado, toda la trayectoria de niveles alcanzados en cada mes de registro, posteriormente se procederá a *recorrer* los registros a modo de que todos los empleados *inicien* con cierto dato y el resto sea su evolución, de este modo algunos empleados tendrán una trayectoria más larga que otros. Posteriormente, se formarán cuaternas de datos en las que cada una contendrá una *fotografía* de 4 meses consecutivos de un empleado (un empleado con 5 registros, tendrá 2 cuaternas, una de sus primeros 4 meses y la otra del segundo mes al quinto). Por último, se implementará el forecasting observando el primer registro de nivel (tiempo t) y ciertas variables asociadas a este y, con ello, se entrenará y evaluará un modelo que pronostique el nivel en el cuarto registro (tiempo $t+3$).

Se inicia por crear esta matriz de niveles mensuales por empleado.

```
serie<-data.frame(datos[!duplicated(datos$no_empleado), ][,3])
names(serie)<-c("Empleado")
#Lo que hace el for es, si el mes (columna) es menor al mínimo
#o mayor al máximo registrado, se pone un cero,
#en caso contrario se toma el registro
for (i in 1:2050) {
  for (j in 2:32) {
    serie[i,j]<-ifelse(ifelse(j<=13, 201800 + j - 1,
                ifelse(j<=25, 201900 + j - 13, 202000 + j - 25))
                < min(filter(datos_t, Empleado == serie[i,1])[,1])
                | ifelse(j<=13, 201800 + j - 1,
                ifelse(j<=25, 201900 + j - 13, 202000 + j - 25))
                >max(filter(datos_t, Empleado == serie[i,1])[,1]),0,
                filter(datos_t, Empleado == serie[i,1],
                Mes ==  ifelse(j<=13, 201800 + j - 1,
                ifelse(j<=25, 201900 + j - 13, 202000 + j - 25)))[,4])
```

```
  }}

names(serie)<-c("Empleado", "Enero_18", "Febrero_18", "Marzo_18",
                "Abril_18", "Mayo_18", "Junio_18", "Julio_18",
                "Agosto_18", "Septiembre_18", "Octubre_18",
                "Noviembre_18", "Diciembre_18","Enero_19",
                "Febrero_19", "Marzo_19", "Abril_19",
                "Mayo_19", "Junio_19", "Julio_19",
                "Agosto_19", "Septiembre_19", "Octubre_19",
                "Noviembre_19", "Diciembre_19", "Enero_20",
                "Febrero_20", "Marzo_20", "Abril_20",
                "Mayo_20", "Junio_20", "Julio_20")
serie[1:2,1:6]
```

```
##   Empleado Enero_18 Febrero_18 Marzo_18 Abril_18 Mayo_18
## 1 80144547        2          2        2        2       2
## 2 90507591        2          2        2        2       2
```

Hasta este punto se ha logrado obtener un dataframe en el que cada renglón corresponde a un empleado y cada columna a un mes determinado comenzando en enero 2018 y finalizando en julio 2020, si un determinado empleado no estuvo activo en un mes específico o en un rango de meses, se tendrá un cero en dicho mes o intervalo.

Si bien se tienen diversas variables descritas anteriormente, por el modo en el que se planteará el modelo predictivo no resulta práctico manejar todas las variables por separado, por lo que se implementará reducción de dimensión mediante TSNE (en específico mediante la aproximación de Barnes Hut por la cantidad de datos). De este modo en vez de crear 15 dataframes o más como el de arriba, se requerirá la creación de únicamente 1 o 2 adicionales.

```
library(Rtsne)
datos_rtsne <- data.matrix(datos[-c(1:3, 7:9, 27)])
#Variables relevantes para el modelo, se retiran variables
#redundantes o de identificación
names(datos[-c(1:3, 7:9, 27)])
```

```
##  [1] "Clientes_nuevos"      "Clientes_renovados"   "Clientes_anticipados"
##  [4] "Intereses"            "ODV"                  "P1_14"
##  [7] "P15_29"               "P30_44"               "P45_59"
## [10] "P60"                  "P90"                  "P120"
## [13] "P150"                 "Atraso_promedio"      "Ciclo_promedio"
## [16] "Grupos"               "Clientes_totales"     "Clientes_21"
## [19] "Tasa_promedio"        "Antiguedad"
```

```
set.seed(2001)
#Mediante el siguiente comanto se condensan los datos
#en 2 dimensiones
rtsne3 <- Rtsne(X = datos_rtsne, is_distance = FALSE, dims = 2,
                check_duplicates = FALSE, max_iter = 10000)

resultados <- as.data.frame(rtsne3$Y)
colnames(resultados) <- c("dim_1", "dim_2")

#Se almacenan estas 2 dimensiones en el dataframe "resultados",
#se añaden variables de identificación
resultados$empleado<-datos$no_empleado
resultados$mes<-datos$mes
resultados[1:4,]
```

```
##        dim_1      dim_2  empleado    mes
## 1  13.756204  68.027556  80144547 201801
## 2   1.525487  39.094793  90507591 201801
## 3 -30.140452 -47.724027  90557696 201801
## 4 -75.877414  -1.960299 200471717 201801
```

Teniendo condensada toda la información en estas 2 dimensiones identificadas por mes y empleado, se procede a crear 2 dataframes (uno por cada dimensión) análogos al ya creado (dataframe *serie*)

```
#Se trabaja del mismo modo, #dejando en cero los meses
#en los que no se tenga registro del empleado
serie_d1<-data.frame(serie$Empleado)
names(serie_d1)<-c("Empleado")
for (i in 1:2050) {
  for (j in 2:32) {
    serie_d1[i,j]<-ifelse(ifelse(j<=13, 201800 + j - 1,
            ifelse(j<=25, 201900 + j - 13, 202000 + j - 25)) <
            min(filter(resultados, empleado == serie_d1[i,1])[,4])
            | ifelse(j<=13, 201800 + j - 1,
            ifelse(j<=25, 201900 + j - 13, 202000 + j - 25))
            >max(filter(resultados, empleado == serie_d1[i,1])[,4]),0,
            filter(resultados, empleado == serie_d1[i,1],
            mes ==  ifelse(j<=13, 201800 + j - 1,
            ifelse(j<=25, 201900 + j - 13, 202000 + j - 25)))[,1])
  }}


names(serie_d1)<-c("Empleado", "Enero_18", "Febrero_18", "Marzo_18",
```

```
                    "Abril_18", "Mayo_18", "Junio_18", "Julio_18",
                    "Agosto_18", "Septiembre_18", "Octubre_18",
                    "Noviembre_18", "Diciembre_18","Enero_19",
                    "Febrero_19", "Marzo_19", "Abril_19", "Mayo_19",
                    "Junio_19", "Julio_19", "Agosto_19",
                    "Septiembre_19", "Octubre_19", "Noviembre_19",
                    "Diciembre_19", "Enero_20", "Febrero_20",
                    "Marzo_20", "Abril_20", "Mayo_20",
                    "Junio_20", "Julio_20")
```

Habiendo replicado el primer dataframe, lo que procede es *recorrer* los registros a modo de que todos los empleados *inicien* en la primera columna. Cuando se tenga una cadena de ceros se registrará únicamente el primero y se tomará como un nuevo *nivel* que representará el mes en el que el empleado causó baja.

```
#Data frame que registra solo los ceros inmediatos y descarta
#cadenas de ceros. En el primer for se convierten los ceros en NA
serie_d11<-data.frame(serie_d1$Empleado)
for (j in 2:32){
  for (i in 1:2050) {
    serie_d11[i,j]<-ifelse(serie_d1[i,j]==0, NA, serie_d1[i,j])
  }}
#En el segundo for se indica que el primer NA después de una
#cadena numérica se tome como cero.
#Es correcto tener series que no tengan registro en todos los meses
#y el último no sea un 0,
#estos son los casos en los que el empleado ingresó después de
#enero 2018 y no registró baja hasta julio 2020
for (j in 2:32){
  for (i in 1:2050) {
    serie_d11[i,j]<-ifelse(is.na(serie_d11[i,j])
                           & serie_d11[i,j-1] != 0 &
                           ifelse(j==32, TRUE,
                           is.na(serie_d11[i,j+1])) &
                           j>2, 0, serie_d11[i,j])
  }}
names(serie_d11)<-c("Empleado", "M1", "M2", "M3", "M4", "M5",
                    "M6", "M7", "M8", "M9", "M10", "M11",
                    "M12", "M13", "M14", "M15", "M16",
                    "M17", "M18", "M19", "M20", "M21",
                    "M22", "M23", "M24", "M25", "M26", "M27",
                    "M28", "M29", "M30", "M31")

#En el siguiente dataframe se indicará el número
#de columnas que deberán recorrerse para cada empleado,
```

```
#los registros cuando se inicia con ceros consecutivos
c<-data.frame(serie$Empleado)

for (i in 1:2050){
  c[i,1]<-ifelse(serie[i,2]>0, 0,
          ifelse(serie[i,3]>0, 1, ifelse(serie[i,4]>0, 2,
          ifelse(serie[i,5]>0, 3, ifelse(serie[i,6]>0, 4,
          ifelse(serie[i,7]>0, 5, ifelse(serie[i,8]>0, 6,
          ifelse(serie[i,9]>0, 7, ifelse(serie[i,10]>0, 8,
          ifelse(serie[i,11]>0, 9, ifelse(serie[i,12]>0, 10,
          ifelse(serie[i,13]>0, 11, ifelse(serie[i,14]>0, 12,
          ifelse(serie[i,15]>0, 13, ifelse(serie[i,16]>0, 14,
          ifelse(serie[i,17]>0, 15, ifelse(serie[i,18]>0, 16,
          ifelse(serie[i,19]>0, 17, ifelse(serie[i,20]>0, 18,
          ifelse(serie[i,21]>0, 19, ifelse(serie[i,22]>0, 20,
          ifelse(serie[i,23]>0, 21, ifelse(serie[i,24]>0, 22,
          ifelse(serie[i,25]>0, 23, ifelse(serie[i,26]>0, 24,
          ifelse(serie[i,27]>0, 25, ifelse(serie[i,28]>0, 26,
          ifelse(serie[i,29]>0, 27, ifelse(serie[i,30]>0, 28,
          ifelse(serie[i,31]>0, 29, ifelse(serie[i,32]>0, 30,
          31))))))))))))))))))))))))))))))))}

#En este for se construye el nuevo data frame recorriendo
#las columnas indicadas por el for anterior,
#el resto de registros se dejan en NA
serie_d12<-data.frame(serie_d11$Empleado)
for (j in 2:32) {
  for (i in 1:2050){
    serie_d12[i,j]<-ifelse(j+c[i,1]>32, NA, serie_d11[i,j+c[i,1]])
  }
}
#A los empleados que tengan un NA precedido y sucedido por algún
#valor numérico, se les sustituye este NA por un 1
#(cercano a la media de los datos)
for (j in 2:32) {
  for (i in 1:2050){
    serie_d12[i,j]<-ifelse(j==32,serie_d12[i,j],
                    ifelse(is.na(serie_d12[i,j]) & serie_d12[i,j-1]>=0 &
                    serie_d12[i,j+1]>=0, 1, serie_d12[i,j]))
  }
}
serie_d12[4:8,1:5]
```

```
##   serie_d11.Empleado         V2         V3        V4        V5
## 4          200471717 -75.877414 -49.818141 -52.40963 -61.26131
```

```
## 5          240391562 -50.795790 -45.624881 -57.11817 -57.05530
## 6          260474251 -43.521392  37.105587  39.57350  45.70960
## 7          290070848   2.855026   2.919842 -12.55739   0.00000
## 8          470385707 100.958311   0.000000        NA        NA
```

Una ves *recorridos* los registros, se procede a crear las cuaternas de datos como ya se explicó.

```
#Vectorización de estaos
serie_k<-serie_d12[,-1]

#Vector de estados, cada 14 es un empleado
#2050 empleados * 31 variables
serie_k2<-c()
for (i in 1:(31*2050)) {
  serie_k2[i]<-serie_k[floor((i-1)/31)+1, i-(31*floor((i-1)/31))]
}

#Se construyen las cuaternas, en este vector se combinan resultados
#de distintos empleados que habrá que omitir al final
#31 meses, 4 variables y 2050 empleados
serie_k3<-c()
for (i in 1:(31*4*2050)) {
  serie_k3[i]<-serie_k2[i-(3*floor((i-1)/4))]
}

#Se crea el arrego como data frame
serie_k4<-matrix(serie_k3, nrow = 31*2050, ncol = 4, byrow = TRUE)
serie_k4<-as.data.frame(serie_k4)

#El siguiente for deja en NA las ternas que combinan registros
#de empleados distintos
for (j in 1:4){
  for (i in 1:(31*2050)) {
    serie_k4[i,j]<-ifelse((i-29)/31==floor((i-29)/31) |
                            (i-30)/31==floor((i-30)/31) |
                            (i-31)/31==floor((i-31)/31), NA,
                          serie_k4[i,j])
  }}
serie_dim1<-na.omit(serie_k4)
serie_dim1[1:3,]
```

```
##          V1        V2        V3          V4
## 1 13.756204 18.130876 -7.887334 64.48850835
## 2 18.130876 -7.887334 64.488508 37.00755394
## 3 -7.887334 64.488508 37.007554  0.03824536
```

Se ha obtenido un dataframe que, como ya se definió, contiene una serie de meses consecutivos con su respectiva primera dimensión para un empleado en específico. Lo que procede es obtener un dataframe análogo para la segunda dimensión y para los *Niveles*. Dado que únicamente se replica el proceso se omitirán comentarios.

```
####################DIMENSION 2###################
serie_d1<-data.frame(serie$Empleado)
names(serie_d1)<-c("Empleado")
for (i in 1:2050) {
  for (j in 2:32) {
    serie_d1[i,j]<-ifelse(ifelse(j<=13, 201800 + j - 1,
            ifelse(j<=25, 201900 + j - 13, 202000 + j - 25)) <
            min(filter(resultados, empleado == serie_d1[i,1])[,4])
            | ifelse(j<=13, 201800 + j - 1,
            ifelse(j<=25, 201900 + j - 13, 202000 + j - 25))
            >max(filter(resultados, empleado == serie_d1[i,1])[,4]),0,
            filter(resultados, empleado == serie_d1[i,1],
            mes ==  ifelse(j<=13, 201800 + j - 1,
            ifelse(j<=25, 201900 + j - 13, 202000 + j - 25)))[,2])
  }}


names(serie_d1)<-c("Empleado", "Enero_18", "Febrero_18", "Marzo_18",
                   "Abril_18", "Mayo_18", "Junio_18", "Julio_18",
                   "Agosto_18", "Septiembre_18", "Octubre_18",
                   "Noviembre_18", "Diciembre_18","Enero_19",
                   "Febrero_19", "Marzo_19", "Abril_19",
                   "Mayo_19",  "Junio_19", "Julio_19",
                   "Agosto_19", "Septiembre_19", "Octubre_19",
                   "Noviembre_19", "Diciembre_19", "Enero_20",
                   "Febrero_20", "Marzo_20", "Abril_20",
                   "Mayo_20", "Junio_20", "Julio_20")
serie_d11<-data.frame(serie_d1$Empleado)
for (j in 2:32){
  for (i in 1:2050) {
    serie_d11[i,j]<-ifelse(serie_d1[i,j]==0, NA, serie_d1[i,j])
  }}
for (j in 2:32){
  for (i in 1:2050) {
    serie_d11[i,j]<-ifelse(is.na(serie_d11[i,j])
                    & serie_d11[i,j-1] != 0 & ifelse(j==32, TRUE,
                    is.na(serie_d11[i,j+1])) &
                    j>2, 0, serie_d11[i,j])
  }}
names(serie_d11)<-c("Empleado", "M1", "M2", "M3", "M4", "M5",
```

```
                    "M6", "M7", "M8", "M9", "M10", "M11", "M12",
                    "M13", "M14", "M15", "M16", "M17", "M18",
                    "M19", "M20", "M21", "M22", "M23", "M24",
                    "M25", "M26", "M27", "M28", "M29", "M30", "M31")
serie_d12<-data.frame(serie_d11$Empleado)
for (j in 2:32) {
  for (i in 1:2050){
    serie_d12[i,j]<-ifelse(j+c[i,1]>32, NA, serie_d11[i,j+c[i,1]])
  }
}
for (j in 2:32) {
  for (i in 1:2050){
    serie_d12[i,j]<-ifelse(j==32,serie_d12[i,j],
              ifelse(is.na(serie_d12[i,j]) & serie_d12[i,j-1]>=0 &
              serie_d12[i,j+1]>=0, 1, serie_d12[i,j]))
  }
}
serie_k<-serie_d12[,-1]
serie_k2<-c()
for (i in 1:(31*2050)) {
  serie_k2[i]<-serie_k[floor((i-1)/31)+1, i-(31*floor((i-1)/31))]
}
serie_k3<-c()
for (i in 1:(31*4*2050)) {
  serie_k3[i]<-serie_k2[i-(3*floor((i-1)/4))]
}
serie_k4<-matrix(serie_k3, nrow = 31*2050, ncol = 4, byrow = TRUE)
serie_k4<-as.data.frame(serie_k4)
for (j in 1:4){
  for (i in 1:(31*2050)) {
    serie_k4[i,j]<-ifelse((i-29)/31==floor((i-29)/31) |
                            (i-30)/31==floor((i-30)/31) |
                            (i-31)/31==floor((i-31)/31), NA,
                          serie_k4[i,j])
  }}
serie_dim2<-na.omit(serie_k4)
serie_dim2[1:3,]
```

```
##         V1       V2       V3       V4
## 1 68.02756 71.30741 60.49390 10.31231
## 2 71.30741 60.49390 10.31231 60.10222
## 3 60.49390 10.31231 60.10222 69.02214
```

Se ha replicado el dataframe de cuaternas para la segunda dimensión, se procede a replicarlo para los niveles

```
################################NIVELES#############################
serie_d11<-data.frame(serie$Empleado)
for (j in 2:32){
  for (i in 1:2050) {
    serie_d11[i,j]<-ifelse(serie[i,j]==0, NA, serie[i,j])
  }}
for (j in 2:32){
  for (i in 1:2050) {
    serie_d11[i,j]<-ifelse(is.na(serie_d11[i,j]) &
                    serie_d11[i,j-1] != 0 & ifelse(j==32,
                    TRUE, is.na(serie_d11[i,j+1])) &
                    j>2, 0, serie_d11[i,j])
  }}
names(serie_d11)<-c("Empleado", "M1", "M2", "M3", "M4", "M5",
                    "M6", "M7", "M8", "M9", "M10", "M11", "M12",
                    "M13", "M14", "M15", "M16", "M17", "M18",
                    "M19", "M20", "M21", "M22", "M23", "M24",
                    "M25", "M26", "M27", "M28", "M29", "M30", "M31")
serie_d12<-data.frame(serie_d11$Empleado)
for (j in 2:32) {
  for (i in 1:2050){
    serie_d12[i,j]<-ifelse(j+c[i,1]>32, NA, serie_d11[i,j+c[i,1]])
  }
}
for (j in 2:32) {
  for (i in 1:2050){
    serie_d12[i,j]<-ifelse(j==32,serie_d12[i,j],
              ifelse(is.na(serie_d12[i,j]) & serie_d12[i,j-1]>=0 &
              serie_d12[i,j+1]>=0, 1, serie_d12[i,j]))
  }
}
serie_k<-serie_d12[,-1]
serie_k2<-c()
for (i in 1:(31*2050)) {
  serie_k2[i]<-serie_k[floor((i-1)/31)+1, i-(31*floor((i-1)/31))]
}
serie_k3<-c()
for (i in 1:(31*4*2050)) {
  serie_k3[i]<-serie_k2[i-(3*floor((i-1)/4))]
}
serie_k4<-matrix(serie_k3, nrow = 31*2050, ncol = 4, byrow = TRUE)
serie_k4<-as.data.frame(serie_k4)
for (j in 1:4){
  for (i in 1:(31*2050)) {
    serie_k4[i,j]<-ifelse((i-29)/31==floor((i-29)/31) |
```

```
                              (i-30)/31==floor((i-30)/31) |
                              (i-31)/31==floor((i-31)/31), NA,
                            serie_k4[i,j])
  }}
serie_4<-na.omit(serie_k4)
serie_4[1:3,]
```

```
##   V1 V2 V3 V4
## 1  2  2  2  2
## 2  2  2  2  2
## 3  2  2  2  2
```

En resumen, lo que se ha logrado con la creación de estos 3 dataframes es el registro de 13,675 series de tiempo de 4 meses en cada uno. En ellas se tienen 2 dimensiones que resumen la información que describe el desempeño del empleado en cada mes y el nivel en el que dicho desempeño posicionó al empleado en el respecivo mes.

Lo que prosigue es la implementación del forecast. El tratamiento que se le ha dado a los datos hace que sea fácil asociar 4 variables:

1. El nivel obtenido por determinado empleado en determinado mes.
2. El valor de todos los KPI's de dicho empleado en el mes en cuestión, condensados en la primera de las dos dimensiones obtenidas mediante t-SNE.
3. El valor de todos los KPI's de dicho empleado en el mes en cuestión, condensados en la segunda de las dos dimensiones obtenidas mediante t-SNE.
4. El nivel obtenido por el empleado en cuestión después de tres meses de la primera observación.

Con ello, los valores de entrada del modelo serán las tres primeras variables descritas (Nivel del primer mes y sus dos dimensiones asociadas), con ellas se hará un pronóstico sobre el nivel esperado después de 3 meses (cuarta variable). El modelo con el que se trabajará será k-nearest neighbours.

```
##########Knearest neighbours CON BAJAS############################
library(kknn)
```

```
## Warning: package 'kknn' was built under R version 4.0.2
```

```
arreglo_knn_d<-data.frame(as.factor(serie_4$V1),
                          as.factor(serie_4$V4),
                          serie_dim1$V1, serie_dim2$V1)
names(arreglo_knn_d)<-c("Mt","Mt3", "D1t", "D2t")

#Fase de entrenamiento
set.seed(400)
indices<-sample(1:length(serie_4$V1),
                round(length(serie_4$V1)*.7),
                replace = FALSE); indices<-sort(indices)
entrenamiento<-arreglo_knn_d[indices, ]
prueba<-arreglo_knn_d[-indices, ]

#Fase de prueba
modelo<-train.kknn(Mt3 ~ ., data = entrenamiento, kmax = 14)
pred<-predict(modelo, prueba[, -2])
CM<-table(prueba[, 2], pred)
CM
```

```
##    pred
##        0    1    2    3    4
##   0    8  141  120    6    1
##   1   13  856  270    2    0
##   2    6  257 1917   83    1
##   3    0    3  138  198   13
##   4    0    1    2   25   42
```

```
(sum(diag(CM)))/sum(CM)
```

```
## [1] 0.7362905
```

Con el análisis anterior se ha obtenido *accuracy* del 74 % lo que, en la práctica, resulta un buen nivel de precisión pues lo que se busca además de ello es poder justificar la toma de decisiones lo cual puede hacerse con este modelo al ser relativamente simple y tener identificadas las variables que ejercen influencia en el resultado lo cual no sería posible implementando modelos o ensambles de modelos más complejos. Adicionalmente, como puede observarse en la matriz de confusión, el error cometido se concentra en los niveles subsecuentes y en las bajas (en el caso de las bajas se presentan de forma aún más fortuita dado que un empleado puede causar baja por ser poco productivo o bien ser tan productivo que sea atraído por la competencia), es decir, el error que se comete no es abismal a modo de encasillar a un empleado poco productivo en los niveles más altos (lo que representaría costos para la empresa) si no que se brinda un criterio fiable sobre el desempeño de los empleados a futuro.

Finalmente, para poner en práctica el modelo se recopilan los intereses cobrados por empleado en el mes de agosto y se cruzan con los registros de nivel y de ambas dimensiones de información correspondientes al mes de mayo para implementar el forecast y ver qué precisión se tuvo. Es decir, se consultará el registro de mayo (tiempo t) y se pronosticará el desempeño de agosto (tiempo $t+3$).

```
#Importación de datos de agosto
agosto<-read.csv(file.choose())
#Se etiqueta mediante la variable Nivel a los empleados
#dependiendo de los intereses cobrados en el mes
agosto$nivel<-ifelse(agosto$Intereses < 58500, 1,
              ifelse(agosto$Intereses < 160000, 2,
              ifelse(agosto$Intereses < 238000, 3, 4)))
names(agosto)<-c("empleado", "Intereses", "Nivel")

#se filtran los registros de mayo
library(dplyr)
names(datos_t)<-c("Mes", "empleado", "Intereses", "Nivel")
mayo<-merge(x = filter(resultados, mes == 202005),
            y = filter(datos_t, Mes == 202005), by = "empleado", all.x = TRUE)

#se cruzan registros de mayo y agosto por empleado
may_ago<-merge(x = mayo[,c(1:3,7)], y = agosto[,c(1,3)],
               by = "empleado", all.x = TRUE)
names(may_ago)<-c("empleado", "D1t", "D2t", "Mt", "Mt3")

#En caso de no tener registro se marca como baja
for (i in 1:526) {
  may_ago[i,5]<-ifelse(is.na(may_ago[i,5]),0,may_ago[i,5])
}

#Se estructuran los datos para implementar el modelo
may_ago_knn<-data.frame(as.factor(may_ago$Mt),
                        as.factor(may_ago$Mt3),
                        may_ago$D1t, may_ago$D2t)
names(may_ago_knn)<-c("Mt","Mt3", "D1t", "D2t")

#Se realiza la predicción sobre el nivel obtenido en agosto
pred<-predict(modelo, may_ago_knn[, -2])
CM<-table(may_ago_knn[, 2], pred)
CM
```

```
##    pred
##       0   1   2   3   4
##   0   1  31  30   1   0
##   1   1 146  98   0   0
```

```
##   2   0  18 174  13   0
##   3   0   0   0   8   2
##   4   0   0   1   0   2
```

```
(sum(diag(CM)))/sum(CM)
```

```
## [1] 0.6292776
```

Se obtiene en la implementación un *accuracy* del 63 %, sigue considerándose un nivel de precisión aceptable volviendo a tener en cuenta que el error cometido se engloba mayoritariamente en los niveles subsecuentes. Adicionalmente, es en estas fechas que empiezan a observarse los efectos de la crisis derivada por la pandemia (reestructuras de plantilla, aumento de clientes morosos, reducción del flujo de efectivo, cierre de sucursales, disminución de créditos otorgados y otros efectos adversos). Esta disminución podrá rectificarse dando mantenimiento al modelo, es decir, insertando estos datos en el conjunto de entrenamiento y dándoles un mayor peso teniendo en cuenta que el comportamiento futuro inmediato tendrá este efecto de contracción económica.

Capítulo 4

Conclusiones

Uno de los principales objetivos del presente era el mostrar un ejemplo de las necesidades que existen en el entorno profesional y como pueden satisfacerse mediante la implementación de los conocimientos adquiridos en el entorno académico. Ello se ha logrado incluso planteando una situación ajena a las áreas laborales típicas de la carrera (seguros y riesgos), lo que además muestra el inmenso abanico de sectores en los que son útiles este tipo de técnicas.

Adicionalmente, se ha dejado patente la necesidad de poner en práctica desde el aula la aplicación de los conocimientos adquiridos fuera de entornos controlados o ideales dado que en la realidad este tipo de entornos son dificiles de hallar pues, como en este caso, ni siquiera la aplicación fue visible de forma trivial dado que los datos requirieron cierto tratamiento antes de poder emplearse y, una vez creado y probado el modelo, se vio seriamente afectado debido a factores externos.

Queda patente también la necesidad de una formación integral dado que en este tipo de implementaciones se necesita no solamente conocimiento técnico sobre los diversos métodos y conceptos adquiridos en el aula y el como poder hacerlos funcionar en conjunto si no que también se necesita cierto conocimiento del negocio o entorno de implementación y por último pero no menos importante, se necesita la habilidad de poder interpretar y comunicar los resultados obtenidos a diversos auditorios que difícilmente dominan los conceptos aplicados para justificar la toma de decisiones.

Finalmente, cabe destacar el valor agregado que generan este tipo de técnicas al segmentar poblaciones que al contar con distintas necesidades requieren un tratamiento específico y al identificar a qué población pertenece cada individuo es posible optimizar los recursos que usualmente son limitados al implementar estrategias de negocio.

Índice de figuras

Referencias

[1] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.*Springer Science+Business Media, LLC.

[2] Brusca, R. C. and Brusca, G. J. (2003). *Invertebrates* (2ª ed.). Sinauer Associates, Inc., Publishers.

[3] Kassambara, A. (2017). *Practical Guide To Cluster Analysis in R - Unsupervised Machine Learning* (1ª ed.). STHDA.

[4] van der Maaten, L. and Hinton, G. (2008). *Visualizing Data using t-SNE. Journal of Machine Learning Research*, 9, 2579-2605. https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

[5] Dalgaard, P. (2002). *Introductory Statistics with R.*Springer

[6] James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013). *An Introduction to Statistical Learning.* (8ª ed.) New York: Springer Science+Business Media.

[7] Friedman, J., Hastie, T. and Tibshirani, R. (2017). *The Elements of Statistical Learning.* (2ª ed.) Springer.

[8] Steinhaus, H. (Octubre 1956). *"Sur la division des corps mat´eriels en parties". Bulletin de L'Academie Polonaise des Sciences, vol. 4, pp. 801-804*

[9] Lloyd, S. (1982). *"Least squares quantization in PCM,". IEEE Transactions on Information Theory, vol. 28, no. 2, pp. 129-137*, doi: 10.1109/TIT.1982.1056489.

[10] MacQueen, J. B. (1967). *"Some Methods for Classification and Analysis of MultiVariate Observations". Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability, pp. 281-297.* University of California Press.

[11] Dempster, A. P., Laird, N. M., Rubin, D. B. (1977). *Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, pp. 1-38.*

[12] Macnaughton-Smith, P. N. M. (1965). *Some Statistical and Other Numerical Techniques for Classifying Individuals. London: H.M.S.O.*

[13] Giordani, P., Brigida Ferraro, M., Martella, F. (2020). *An Introduction to Clustering with R.*(1ª ed.) Springer, Singapore