ESCUELA SUPERIOR POLITECNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Modelo de caracterización de clientes para una empresa de la industria retail donde su principal línea de negocio es la venta de electrodomésticos

PROYECTO DE TITULACIÓN

Previo la obtención del Título de:

Magister en Ciencias de Datos

Presentado por:

Córdova Villagómez, Lizsette María

Hernandez Cano, Danny Leonel

GUAYAQUIL - ECUADOR

Año: 2021

AGRADECIMIENTO

A nuestro padre celestial y virgen María, porque son aquellos quienes dan la facultad de pensar, imaginar y razonar. Aquella persona espiritual que nos ha acompañado a lo largo de nuestra carrera universitaria.

También a nuestras familias, amigos y a todas aquellas personas que nos brindaron sus fundamentos sin limitaciones, y finalmente a quienes de una u otra forma nos contribuyeron con su apoyo.

GRACIAS A TODOS.

DEDICATORIA

Dedicamos este proyecto de graduación en especial al ser omnipotente, porque es el que nos ha dado fuerza y valor para progresar cada día un poco más y terminar con éxito nuestro trabajo.

Dedicamos también a nuestras familias que nos supieron apoyar en todo momento; y a cada una de las personas externas que nos ayudaron a culminarlo.

DECLARACIÓN EXPRESA

"Los derechos de titularidad y explotación, me(nos) corresponde conforme al reglamento de propiedad intelectual de la institución; (nombre de los participantes) y doy(damos) mi(nuestro) consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

(Reglamento de Graduación de la ESPOL)

Lizsette Córdova

Lizsette Córdova Villagomez

Danny Hernández C

Danny Hernandez Cano

COMITÉ EVALUADOR

Firmado electrónicamente por:
EDUARDO SEGUNDO
CRUZ RAMIREZ

Msc, Eduardo Cruz

PROFESOR TUTOR

Firmado electrónicamente por:
JOSE EDUARDO
CORDOVA GARCIA

Phd. José Córdova

PROFESOR EVALUADOR

RESUMEN

Una compañía de retail que principalmente comercializa electrodomésticos dentro del Ecuador, se ha planteado nuevas estrategias comerciales para aumentar sus volúmenes de ventas como lo es; el proporcionar a sus clientes una mejor experiencia de consumo ofreciendo un mejor producto y/o servicio, aumentar la proposición de valor mediante una propuesta focalizada y lograr ubicarse a nivel nacional como líder del sector en el que desarrolla sus operaciones. Esto hace obligatorio mantenerse a la vanguardia en tecnología y uso de herramientas o sistemas inteligentes aplicado al mercado retail.

La empresa realiza campañas de ventas vía Sms, WhatsApp, mail y otros medios digitales, dirigida a todos los clientes posibles. No focaliza las campañas a segmentos específicos. No considera las temporadas más adecuadas en las que sus clientes compran. Este proceso genera pérdida de recursos al aumentar los costos operativos y por mantener una plataforma de contact center recurrentemente comunicándose con los clientes para el ofrecimiento de productos.

En el presente proyecto se realizará el entrenamiento, validación y calibración de un modelo de caracterización de clientes que genera grupos con características similares utilizando algoritmos de aprendizaje supervisados y no supervisados. El entrenamiento se realizará identificando las características más relevantes de las transaccionales de compras de electrodomésticos de los clientes en los diferentes almacenes de la empresa.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

CAPÍTULO 1

1. Análisis de la problemática

Según el Banco Mundial en su publicación de junio 2021, menciona qué después de que empezó la situación crítica del COVID-19, la economía en todo el mundo está en condiciones de empezar su recuperación en 2021 luego de una recesión sólida en 8 décadas, no obstante, se estima que el despunte no será igual para todas las naciones, ya que las economías más transcendentales se prevén estar listas para reconocer un alto crecimiento, aunque algunos países registran economías en desarrollo. [1].

Se estima que el crecimiento económico en el mundo se activará en un 5,6 % en 2021, y esto corresponderá en gran parte a las principales economías de los países de Asía y Norte América. Sin embargo, el crecimiento de ciertos países ha estado reconocido en alza para 2021, algunos siguen enfrentando la situación del COVID-19 con sus muchos efectos. Pero el incremento en el 2021, se espera que el nivel del PIB en el mundo en el 2021 sea de un 3,2% menor a las suposiciones anteriores al COVID-19, y se estima que el PIB per cápita para los países en desarrollo se conservará menor a los niveles grandes a priori a la pandemia transcurrido en un periodo amplio. [1].

Entre las naciones con economías de bajo ingreso, donde el proceso de vacunación se ha puesto lento, el crecimiento económico se ha detenido, y lo que se espera es que sea un crecimiento de un 2,9 %. Si no se toma en cuenta la contracción del año 2020, el año con mayor expansión en pausa en 20 años sería el 2021. Se estima que la producción sea menor que lo trabajado antes del COVID-19 en un 4.9%. Los países con una economía menor están afectados por la inestabilidad y compromiso debido al mayor castigo por el COVID-19, y aquellos que tienen una economía alta per cápita han disminuido en un periodo de 10 años. [1].

1.1 Descripción del problema

El Ecuador por ser una economía en desarrollo aún sigue afrontando los efectos de la pandemia, los cuales serán mantenidos a largo plazo. [2]. En 2020 las empresas que adoptaron la trasformación digital como parte de su cultura empresarial lograron mantenerse activas a pesar de los efectos adversos a causa del COVID-19, este salto tecnológico a la Robótica, Big Data, Inteligencia Artificial, e Internet jugó un rol importante en su posicionamiento y hasta en su permanencia en el mercado, lo que no sucedió con negocios de menor escala, ya que el confinamiento en algunos casos provocó el cierre de negocios de forma definitiva y en otros han cerrado parcialmente su actividad, debido a que para sus procesos de producción y comercialización no realizaron ningún cambio en sus estrategias de ventas hacia medios tecnológicos que se adapten a la realidad que vivió el país a causa de la pandemia.

El presente proyecto estará enfocado en proponer una solución a la problemática de una empresa del sector del comercio minorista con más de 7 décadas en el mercado ecuatoriano que al igual que la mayoría en el Ecuador también se ha visto golpeada por los efectos de la pandemia mundial de COVID-19 provocando una reducción considerable de sus volúmenes de ventas (alrededor de un 43% de contracción durante el año 2020 y el 17% hasta agosto del 2021 en comparación a las ventas antes de la pandemia), por lo que la misma se ha visto obligada a realizar múltiples estrategias para su recuperación como la reducción de costos para optimizar su operación y el planteamiento de nuevos objetivos estratégicos. La empresa dispone de un amplio catálogo de productos, aproximadamente más de 10,000 productos diferentes, teniendo su principal enfoque en la venta de electrodomésticos tanto para venta de contado como con crédito directo, posee al 2021, 57 locales físicos distribuidos entre las ciudades mayormente pobladas del Ecuador como: Guayaquil, Quito, Manta, Portoviejo, Ambato, Quevedo, Santo Domingo, Esmeraldas entre otras y comercializa sus productos por diferentes canales como: comercio electrónico y el ofrecimiento comercial por medio de su centro de atención telefónica.

La empresa ha buscado la forma de renovar también las estrategias comerciales para mejorar el volumen de ventas en la compañía, sin embargo, la empresa actualmente realiza campañas de ventas vía sms, whatsapp, mail y otros medios digitales, dirigida a todos los clientes con los que la empresa cuenta en su banco de clientes históricos, sin focalizar las campañas a segmentos específicos, este proceso

de generar campañas de venta a todos los clientes genera pérdida de recursos para la compañía al aumentar sus costos operativos por mantener una plataforma de contact center recurrentemente comunicándose con los clientes para el ofrecimiento de productos y costos de nómina por el recurso humano utilizado en esta atención. Por lo tanto, es importante identificar el segmento de clientes al que se le puede ofrecer un conjunto de productos específicos al momento de realizar una campaña comercial optimizando el proceso, ahorrando recursos para aumentar sus ingresos y con esto alcanzar el cumplimiento de las metas que la compañía se ha establecido como parte de sus nuevas estrategias como son; lograr posicionarse en el top 70 de ventas de las empresas en el ecuador y ser la mejor opción de compra y referente para sus grupos de interés, así como logar el 40% de participación de ventas en los canales no tradicionales.

1.2 Solución Propuesta

Se propone crear un modelo de caracterización de clientes partiendo de un análisis previo de la información de ventas para conocer el comportamiento de consumo de los clientes, a fin de obtener características que sean diferenciadoras al momento de generar el modelo de caracterización, con esto obtener los segmentos y proporcionar una herramienta de visualización de los resultados para dirigir campañas de manera más eficiente y dirigida a los segmentos objetivo de cada campaña.

Con la segmentación de perfiles la empresa podrá encontrar agrupaciones naturales de clientes según sus hábitos de consumo para establecer estrategias específicas para cada segmento, y lograr el cumplimiento de las nuevas estrategias planteadas por la empresa, entre los beneficios que se podrán obtener se puede mencionar:

- Elaborar productos, servicios y mensajes personalizados.
- Crear y lanzar “campañas” a segmentos/targets específicos.
- Agradar clientes nuevos con un perfil esperado.
- Aumentar el volumen de operaciones.
- Incentivar el consumo de diferentes productos.
- Fidelizar ciertos clientes.

1.3 Objetivos

1.3.1 Objetivos generales

Crear un modelo de categorización de clientes, mediante el análisis de las compras que se realizan en una empresa de la industria retail, para dirigir estrategias de marketing eficientes enfocadas a segmentos con necesidades específicas de productos.

1.3.2 Objetivos específicos

Extraer las características temporales, monetarias y demográficas, mediante el análisis de las transacciones de compras, para caracterizar patrones significativos de los clientes.

Construir un modelo de categorización de clientes, mediante la utilización de algoritmos de aprendizaje supervisado y no supervisado, para predecir segmentos de clientes.

Implementar una herramienta de visualización para presentar de forma interactiva los segmentos de clientes.

1.4 Metodología

En este proyecto se analizan varias técnicas de clasificación de clientes de acuerdo con su comportamiento de consumo y obtener diferentes subconjuntos (o grupos). El principal objetivo es obtener grupos con la mayor similitud entre sus miembros de cada segmento y que a su vez sean diferenciables de los otros grupos.

El desarrollo del proyecto se encuentra dividido en etapas o fases de análisis como se muestra en la Ilustración 1.

En la primera fase se realizará el proceso de ETL (extracción, transformación y carga de los datos de la compañía) con su respectiva selección de atributos. Luego se

limpiará la información eliminando valores outliers, valores ausentes y la corrección del tipo de las características. Este proceso de limpieza se las identifica y visualiza con los gráficos de la distribución de las características cuantitativas, usando gráficos como: Distplot y Boxplot, con la librería Matplotlib.

La segunda fase será del análisis exploratorio de los datos en donde identificaremos el comportamiento de venta de los clientes, su frecuencia, monto, recencia, temporalidad de compra, entre otros. Se agregarán al análisis características de tipo demográficas de los clientes a partir de datos externos obtenidos por la compañía, donde se incorporarán datos como: sexo, estado civil y edad. Posteriormente se realizará la selección de las características más significativas que serán considerados para el modelo de clasificación.

La tercera fase es la creación del modelo de caracterización, en donde se diseñarán, entrenarán, validarán y calibrarán algoritmos supervisados y no supervisados. El análisis de clústers se lo utilizará para encontrar grupos naturales de acuerdo con su comportamiento de compras. Para el proceso de etiquetado de los grupos aplicaremos el algoritmo no supervisado de agrupamiento k-means. El algoritmo k-means trabaja de forma iterativa, actualizando la ubicación de los grupos de manera de ir disminuyendo las distancias entre los individuos de cada grupo y su centroide. Las etiquetas de los grupos serán utilizadas para entrenar algoritmos supervisados de machine learning como: Naive Bayes, KNN, SVM, Arboles Decisión y Redes Neuronales. Posteriormente se seleccionarán el mejor modelo de acuerdo con sus métricas de evaluación para luego segmentar y perfilar a los clientes. Esta fase será desarrollada en el lenguaje de programación Python y su librería scikit-learn.

Y la última fase del proyecto consistirá en el desarrollo de una herramienta BI para visualizar los segmentos de los clientes y sus características.

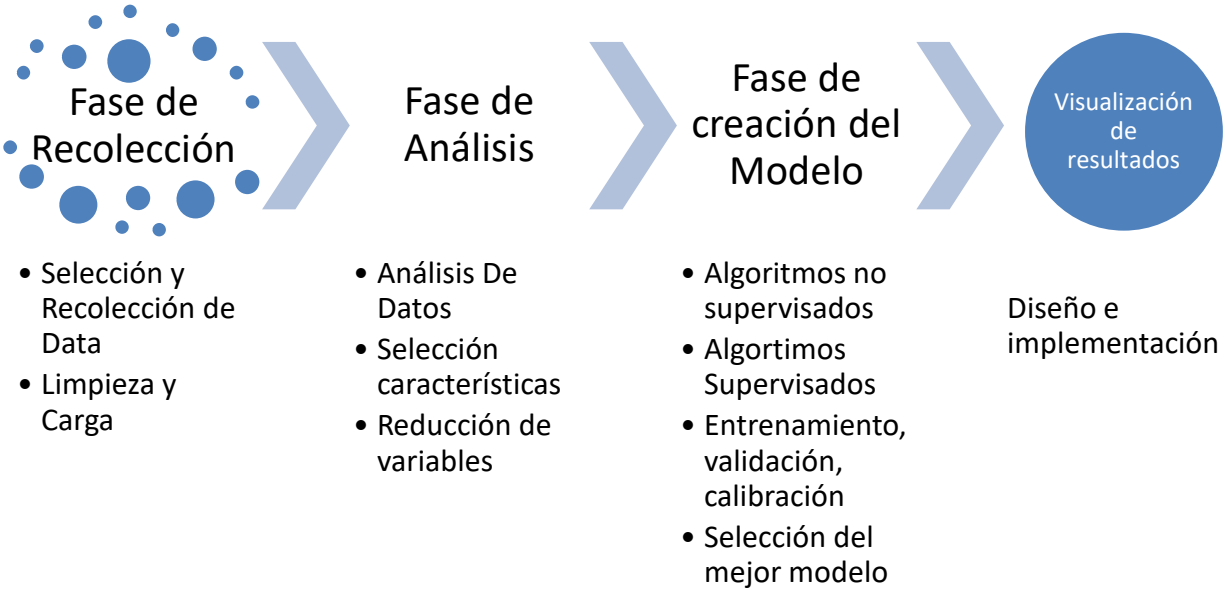


Ilustración 1. Proceso de la metodología aplicada.

1.5 Resultados esperados

El presente proyecto de caracterización de clientes tendrá como resultados la entrega de lo siguiente:

Visualizaciones con graficas estadísticas de la caracterización de los clientes de acuerdo con su comportamiento de compra.

Modelo de caracterización de clientes como producto de la utilización de algoritmos de aprendizaje supervisado y no supervisado.

Herramienta de visualización para la presentación de los resultados de los segmentos de clientes.

1.6 Dataset

Para la implementación del proyecto la empresa dispone de información estructurada en modelos de datos relacionales pertenecientes a los cubos de información de la empresa. Estos modelos de datos se encuentran en el servidor de datawarehouse que posee bases de datos en Microsoft SQLServer. La información de

los modelos de datos de los cubos se actualiza diariamente mediante proceso batch nocturno y mantiene la información actualizada al corte del día anterior.

Para la construcción del modelo de segmentación de clientes se utilizará la información que contiene el modelo de datos del cubo de ventas que tiene un diseño de tipo estrella como se ilustra en la Ilustración 2, el cual contiene registros detallados a nivel de ítem de factura y contiene documentos de ventas realizadas desde enero de 2015 hasta el 2020.

La tabla de hechos del modelo contiene datos relacionados a la transacción de venta como: fecha de transacción, número de factura, canal de venta, identificación del cliente, forma de pago, monto de compra, monto de descuento, monto de intereses, monto de devoluciones, unidades vendidas, código del producto, código de promoción, código de combo, vendedor.

La dimensión de clientes contiene algunos datos demográficos como: sexo, estado civil, ciudad, provincia, promedio de ingresos y fecha de nacimiento.

La dimensión de productos posee características de los productos vendidos; descripción, líneas, grupos, subgrupo, capacidad, marca.

El modelo de datos posee atributos de los almacenes como: descripción, provincia, ciudad y zona, pero no participaran en este análisis.

De este modelo de datos se extraerá la información entre los años 2015 a 2020 como insumo para la obtención de características de tipo temporales, monetaria y de tipo demográficas para intuir el comportamiento de los consumidores.

El dataset contiene 1,761.308 registros y 23 características, de las cuales 17 son categóricas, 5 son numéricas y una de tiempo, es decir, presenta una data mixta o combinada, un total de 439.452 clientes donde un 60% de género masculino y 40% de género femenino.

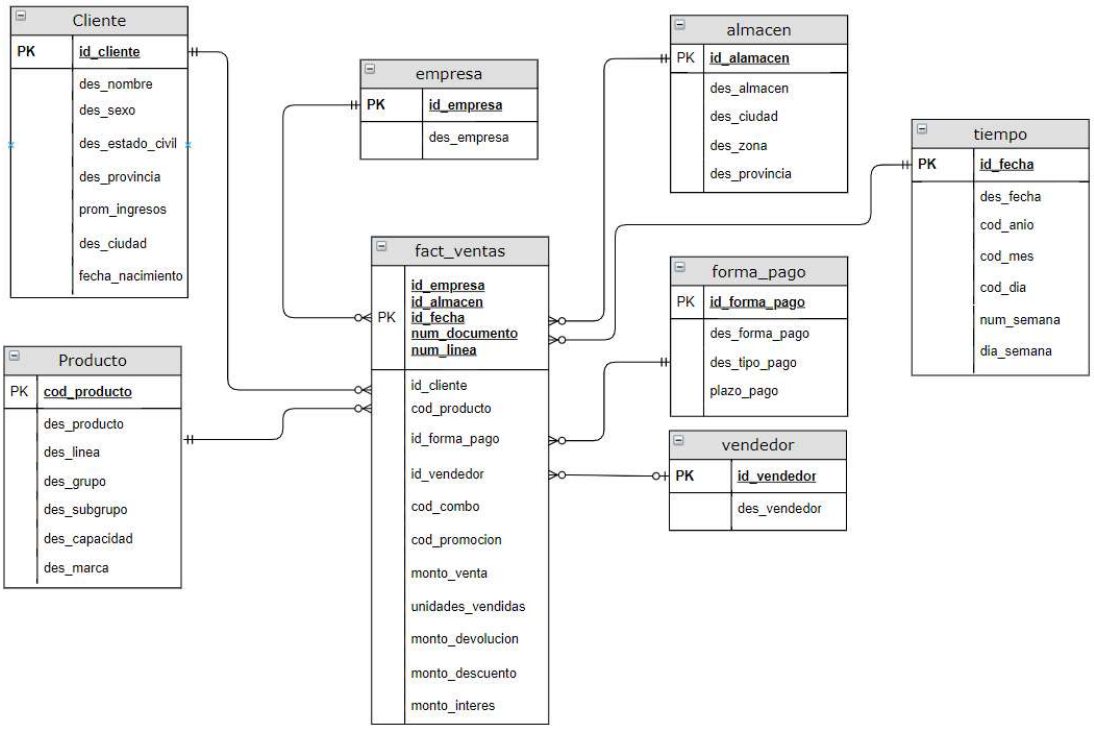


Ilustración 2 Modelo de datos del cubo de ventas

CAPÍTULO 2

2. Estado del Arte

La clasificación supervisada es una de las labores que más comúnmente son llevadas a cabo por los nombrados Sistemas Inteligentes. Consecuentemente, un gran número de paradigmas perfeccionados bien por la estadística como son: Análisis Discriminante, Regresión Logística, o bien por la Inteligencia Artificial: Random Forest, Asociación de Reglas y Redes Neuronales son capaces de ejecutar los trabajos propios de la clasificación. [3]

2.1 Fundamentos del problema

La empresa se ha planteado nuevas estrategias comerciales para aumentar sus volúmenes de ventas como lo es: el proporcionar a sus clientes una mejor experiencia de consumo ofreciendo un mejor producto y/o servicio, aumentar la propuesta de valor a través de la focalización y lograr ubicarse a nivel nacional como líder del sector en el que desarrolla sus operaciones. Esto hace obligatorio mantenerse a la vanguardia en tecnología y uso de herramientas o sistemas inteligentes aplicado al mercado retail.

El proyecto consiste en el desarrollo de un modelo de caracterización de clientes a través de datos transaccionales de ventas de la compañía estableciendo características con mayor importancia de los clientes justo al realizar la compra, e identificar las características que serán parte del modelo de clasificación. Luego con este obtener los grupos con características comunes y construir modelos no supervisados y supervisados, realizar el entrenamiento, validación, calibración para finalmente seleccionar el mejor modelo.

2.2 Soluciones de analítica, aprendizaje y fuentes de datos relacionadas a la caracterización de clientes

El proceso de **caracterizar** o segmentar a clientes, consiste en identificar los grupos de clientes que poseen **características** comunes entre sí y que son del interés para una compañía. La caracterización se logra a través del uso de características de tipo demográfico, geográfico, de comportamiento o intrínsecos. Para este proceso se puede encontrar variedad de soluciones de analíticas y aprendizajes relacionadas que aplican algoritmos de Machine Learning. [4].

Los algoritmos de máquina de aprendizaje trabajan estableciendo sistemas que nivelan patrones oscuros en enormes volúmenes de datos, es decir, que estudian el dataset y están capacitados de predecir. Estos algoritmos contienen una gran ventaja que aprenden instintivamente, es decir, con el tiempo incrementan su capacidad de aprender sin tomar en cuenta a las personas. [4].

Se tiene dos algoritmos de Machine Learning: no supervisada y supervisada.

Algoritmo supervisado. Quiere decir al aprendizaje automatizado en donde ya se conoce la salida al que se quiere llegar de un conjunto de datos ya etiquetados, es decir, ya se parte del conocimiento del valor del objetivo. [4].

Algoritmo no supervisado. Se refiere al aprendizaje de maquina cuando no se conoce el conjunto de salidas y se tiene un volumen de datos no etiquetados, es decir, buscan comportamientos o patrones significativos en los datos. [4].

Sus principales aplicaciones son: buscar anomalías, predecir, clasificar y distinguir patrones:

- Búsqueda de anomalías. - Es encontrar e identificar datos discordantes en relación con los demás.
- Predicción (Regression). - Se basa en encontrar relaciones entre las características para predecir datos futuros.
- Clasificación de categorías. – Radica en determinar en un volumen de datos a que categorías pertenece los nuevos datos.
- Clustering de patrones. – Se trata de agrupar los datos en cada clase. [4].

Análisis de Clustering

El análisis de clústers (AC) es un algoritmo no supervisado dentro del Machine Learning que tiene como finalidad reducir las dimensiones del dataset mediante técnicas multivariantes con el objetivo de agrupar datos considerando las características que lo conforman. El AC clasifica los datos en grupos o conglomerados de cierta manera que cada objeto sea similar a los de los otros grupos. Esta técnica es la que utilizaremos para agrupar a los clientes y clasificar nuestro set de datos. [3].

Método de dendograma

Un dendrograma, es un diagrama que visualiza las distancias de los grupos entre cada par de grupos fusionados de manera secuencial. La distancia que se usó para la realización del diagrama es la media.

Método Gaussian

El método Gaussian GMM, tiene la ventaja que puede adoptar diferentes tamaños de grupos y estructuras de correlación dentro de cada grupo. Dentro del cálculo de medición de distancia del método GMM considera la varianza, es decir, en GMM se calcula la distancia ponderada, una de la diferencia con los métodos de agrupaciones tradicionales como kmean, que calcula la distancia euclidiana convencional.

Método Kmean

El método kmean es el más común y tradicional para agrupar individuos con características o perfiles similares, obtiene grupos homogéneos dentro de ellos y heterogéneos entre cada uno de ellos, al igual que el método GMM también se requiere de un numero de k grupos como parámetro para la segmentación.

Algoritmos supervisados, modelos de clasificación

La clasificación supervisada consiste en entrenar algoritmos de machine learning para identificar comportamiento en los datos, aprender de los clientes y enseñar a la maquina a clasificar un conjunto de datos conocidos en la que se incluyen unos conjuntos de características como entrada para dar categorizaciones deseadas como salidas. Este proceso se realizará hasta que el algoritmo alcance el nivel alto de rendimiento y precisión.

Existen una variedad de modelos de clasificación dentro del área de machine learning, de los cuales se usarán los siguientes: Naive Bayes, KNN, SVM, Arboles de Decisión y Redes Neuronales. Para el desarrollo de estos modelos es necesario determinar la partición de dos subconjuntos de datos, de las cuales un dataset, el más grande, será utilizado para evaluar los parámetros y entrenar el modelo y el otro dataset, más pequeño, para comprobar el comportamiento del del modelo entrenado.

Cross Validation

En un conjunto de datos se particiona en k submuestras, esto se denomina validación cruzada con k iteraciones. En cada interacción se determina como data de prueba a un subconjunto del dataset mientras que las otras submuestras (k-1) se determina como data de entrenamiento. Estas interacciones se repite k veces para cada submuestra del dataset. Por último, se calcula estadísticamente las medias de cada resultado de la interacción para determinar un resultado. Con este método certificamos que el conjunto de datos de prueba y entrenamiento son considerados en cada momento de la interacción, aunque computacionalmente es lento cuando se trata de grandes volúmenes de datos. Para considerar el número óptimo de iteraciones se calcula la medida del conjunto del dataset. [5]. Ver Ilustración 3.

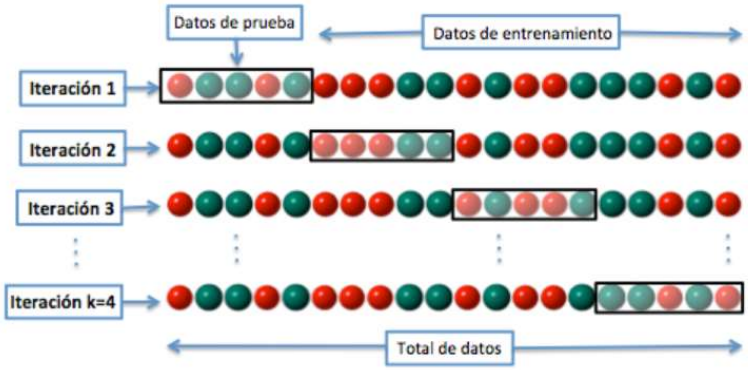


Ilustración 3 Cross-Validation de K iteraciones con K=4.

Naive Bayes Gaussiano

Es un modelo clasificador simple que proporciona aceptables resultados con los hiper-parametros del modelo, no requiere de una gran cantidad de datos para ser robustos. Este modelo es mayormente usado cuando se tiene problemas derivados de la maldición de la dimensión, es decir, muchas características en el dataset, en este proyecto, 21 características dummies.

Naive bayes considera que la participación de una característica dentro del clasificador no está relacionada con las otras características, por ejemplo; un cliente puede ser estimado como “bueno” si tiene frecuencia mayor a 2, monto de venta promedio superior a $400 y compra a crédito, el modelo Naive Bayes considera cada una de estas características en forma independiente para que la probabilidad gaussiana del cliente sea clasificada como “bueno”.

K-Nearest Neighbords

Algoritmo supervisado de Machine learning que se utiliza para predecir valores continuos y para clasificar valores discretos con nuevas categorías, para nuestro caso se utilizará el modelo de “vecinos cercanos” para clasificar los 4 grupos de clientes que ya conocemos de antemano.

Este modelo consiste en buscar observaciones que estén lo más cerca del objetivo que se quiere predecir, para luego clasificar el punto de interés considerando los puntos que lo rodean, es decir, clasifica cada grupo de cliente de acuerdo con las características más cercanas a cada clase etiquetada. Una de las desventajas es la demanda de recursos en memoria y procesamiento en CPU que requiere, esto debido a que utiliza toda la data para entrenar cada punto, por eso el modelo KNN funciona mejor con datasets pequeños, nuestro dataset cumple esta condición.

Support Vector Machine (kernel Linear)

Este algoritmo de aprendizaje supervisado se utiliza para problemas relacionados con la regresión y mayor frecuencia con la clasificación. Se trata en representar cada elemento de los datos en un espacio de p dimensiones, en donde p es la cantidad de características que tiene el dataset, luego se realiza la clasificación

separando los grupos de las clases lo más distantes entre ellos, mediante un hiperplano en la cual diferencie todas las clases, este hiperplano se denomina, vector soporte.

Para calcular el mejor hiperplano es importante identificar, entre todos, el vector soporte correcto, que separe bien las clases maximizando las distancias entre los puntos más cercano de cualquier clase y el hiperplano correcto, esta distancia se denomina, margen. El modelo SVM clasifica a los clientes seleccionado el hiperplano con precisión antes de maximizar el margen.

Arboles de Decisión

Son algoritmos de aprendizaje supervisado de machine learning que tiene como objetivo clasificar grupos formados por reglas binarias, por ejemplo; bueno/malo, alto/bajo, si/no, entre otros. Cuando en el problema se tiene más de dos clases, se denomina, multi-class. Este modelo es bastante preciso para la toma de decisiones por su estructura jerárquica en forma de árbol y se adapta muy bien cuando se trabaja con grandes dimensiones, aunque es un poco débil cuando se trata de ruidos en los datos.

El algoritmo de árbol sugiere la división binaria recursiva, es decir, evalúa la mejor variable para la ramificación únicamente del proceso de división actual. Ver Ilustración 4.

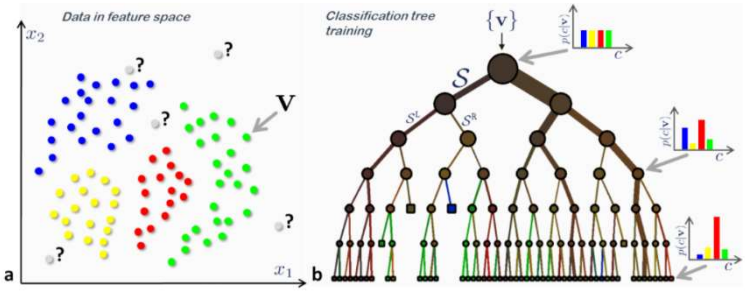


Ilustración 4 Clasificación de tipo jerárquica, árbol de decisión

Uno de los parámetros para el modelo de árboles de decisiones, es el criterio de la entropía, que significa el grado de congestión de un determinado conjunto de datos, es decir, si los tipos de registros dentro del conjunto de datos son diversos, la entropía es alta, y cuando los registros tienen un montón de un mismo tipo, la entropía es baja. Ver Ilustración 5.

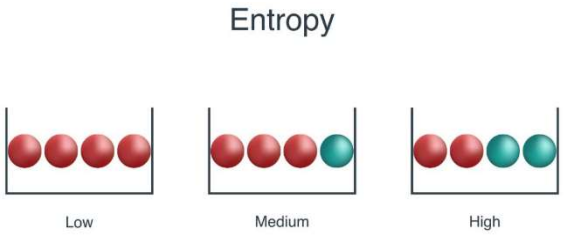


Ilustración 5 Grafica de Entropía

Redes Neuronales

Los métodos para implementar un clasificador de multi-class han mostrado ser efectivos, como lo son las redes neuronales, MLP, que consiste en una red neuronal constituida por k-capas de neuronas, donde cada capa representa a un conjunto de neuronas que recibirá las características de entrada por medio de conexiones o pesos y cada neurona producirá su salida usando una función de activación para procesar el patrón de entrada de una o más capas ocultas, seguida de otras capas anteriores que estarán conectadas con la última capa oculta en la cual representará la red final.

Análisis de rendimiento del modelo

Para el análisis de rendimiento de los algoritmos supervisados se define las métricas de evaluación, que son mecanismos integrales de los proyectos de ciencia de los datos. El propósito es estimar el accuracy de un modelo sobre los futuros datos.

Matriz de confusión

Es representada con características en forma de matriz con los pronósticos de las pruebas binarias que son utilizados repetidamente para resumir en el modelo de clasificación o "clasificador" su rendimiento acerca del conjunto de la data test versus los resultados existente conocidos.

Esta técnica es muy fácil de entenderla, pero la desventaja es que los términos relacionados podrían ser un poco complejos. [6] . Ver Ilustración 6.

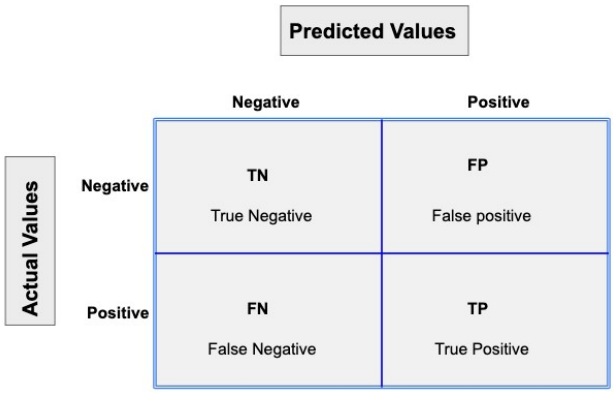


Ilustración 6 Matriz de Confusión explicativa.

Matriz de confusión con dos grupos clases.

Cada predictor será uno de los cuatros resultados, basado en cómo concuerda con el valor real:

Verdadero Positivo (TP): Predicho Verdadero y Verdadero en realidad.

Verdadero Negativo (TN): Predicho Falso y Falso en realidad.

Falso Positivo (FP): Predicción de verdadero y falso en la realidad.

Falso Negativo (FN): Predicción de falso y verdadero en la realidad. [6].

Precisión

¿Cuál es la frecuencia si predecimos lo correcto?

Precisión=TP/predicciones sí

Cuando registramos una inestabilidad en alguna clase, el desempeño se puede medir reconciliando la precisión con una métrica de poca confianza. Es decir, si obtenemos la operación de 99/1 con los grupos de A y B, considerando que el grupo B es el positivo, se podría fácilmente desarrollar un modelo en donde la mayoría sean del grupo A con un 99% de precisión. No se compensaría en incomodar en desarrollar un modelo si no tenemos como evidenciar al grupo B, por lo tanto, necesitaremos métricas distintas para determinar el comportamiento de los grupos. Y en vez de manejar a la exactitud consideramos a la precisión y sensibilidad. [6]

Se puede observar la Ilustración 7**¡Error! No se encuentra el origen de la referencia.** en donde se presenta las métricas evaluadas dentro de la matriz de confusión.

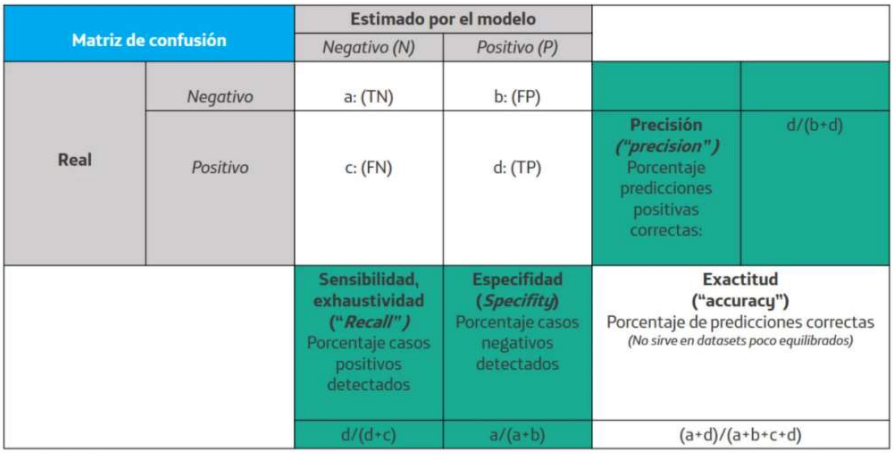


Ilustración 7 Métricas evaluadas en matriz de Confusión.

Análisis de Baias y varianza del modelo

La varianza y sesgo son los dos conceptos relevantes al momento de cuantificar el error en los modelos supervisados. Por lo tanto, es necesario entender su concepto para evaluar de manera correcta lo que nos explican. [7].

Bias

El Bias (sesgo) mide lo lejos que se encuentra el valor estimado respecto al real de la población completa. Por ejemplo, si se desea calcular la vida media de unas bombillas es necesario escoger una muestra. El tiempo de vida promedio de esta muestra es el que se le asocia a la población, pero no tiene porque es el de la población total. Este error es lo que se llama como sesgo. [7].

Varianza

Al trabajar con una muestra aleatoria de la población total es de esperar que sea diferente de otra muestra. Esta diferencia entre las muestras es lo que la varianza. Así cada vez que se realiza un nuevo muestreo se observa que los resultados suelen ser diferentes. [7]. Ver Ilustración 8.

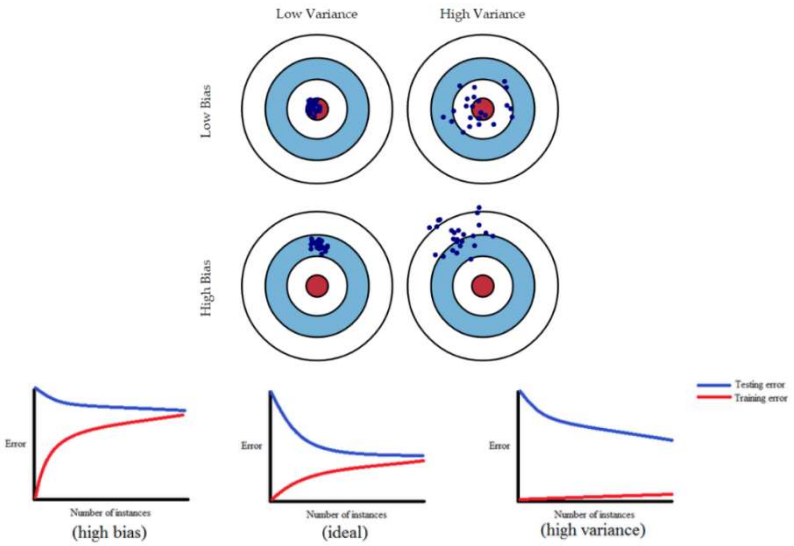


Ilustración 8 Análsis de Bias y Variance.

Reglas de Asociación.

Una vez identificado a los perfiles de los grupos de clientes, se presenta unos de los problemas a menudo en las empresas de Retail y es, la recomendación de productos complementarios a sus compradores basados en el histórico de sus transacciones, es decir, sugerirle al cliente productos que anteriormente llevaban juntos con frecuencia o tal vez recomendarle productos nuevos que reemplazarían a los de su preferencia o ya posicionados. Este sistema de recomendación se basa en el análisis de las transacciones individuales de los clientes, se denomina Reglas de Asociación, ya que identifica patrones dentro de cada transacción y genera reglas de relaciones de productos, esto se desarrolla mediante el algoritmo a priori. [8]

La forma como se describen estas asociaciones de reglas es:

$$\{Antecedente\} \Longrightarrow \{Consecuente\}$$

Lo cual se entiende que el producto que compro anteriormente tiene relación con el producto que comprará después dentro de una misma venta. Los indicadores para cuantificar esta relación entre el antecedente y consecuente son: el soporte, la confianza y la mejora de la confianza. [9].

Soporte

Dentro de la regla, el soporte se denomina a la frecuencia relativa como se muestra. Por ejemplo, un soporte de 0.80 quiere decir que el antecedente y consecuente registran una relación del 80% dentro de la misma factura. Este indicador cuantifica la participación de la regla. Los valores permitidos están en el rango de 0 y 1. [9].

Confianza

Dentro de las transacciones se denomina a la confianza como el porcentaje cuando se visualiza el antecedente y también se visualiza el consecuente, es decir, mide la precisión de las reglas. El cálculo para obtener este indicador se utiliza la siguiente expresión:

$$\text{confianza}(\{\text{Antecedente}\} \Longrightarrow \{\text{Consecuente}\}) = \text{confianza}(\{\text{Antecedente,Consecuente}\})/\text{confianza}(\{\text{Antecedente}\})$$

Cuando,

$$\text{confianza}(\{\text{Antecedente}\} \Longrightarrow \{\text{Consecuente}\})$$

se muestra el antecedente y consecuente juntos se denomina confianza de la regla. Ejemplo si se registra 100 productos en el antecedente y 200 productos con el consecuente y 80 productos más entre los dos, se tiene finalmente una confianza para la regla del 0,8.

Para esta regla distinta $\{\text{Consecuente}\} \Longrightarrow \{\text{Antecedente}\}$ se registra un soporte similar, pero con una confianza distinta del 0,4. Lo cual se evidencia que la segunda regla registra menor peso que la primera, dando a que el antecedente es lo que hace que el consecuente aparezca, pero al revés no es lo mismo. Los valores permitidos para este indicador están en el rango de 0 y 1. [9]

Mejora de la confianza

Por último, la mejora de la confianza se denomina como a la participación de la confianza entre las reglas con respecto a lo que se podría obtener si fueran independientes. El cálculo para este indicador se encuentra en la siguiente expresión:

confianza({Antecedente}⟹{Consecuente}) = confianza({Antecedente,Consecuente}) / (confianza({Antecedente}) × confianza({Consecuente }))

Cuando la regla se dificulta identificarla el valor se aproxima a 1. Pero si incrementa el valor entonces podremos decir que se diferencia mucho mejor a la regla. [9].

Si los artículos A y B se compran juntos con más frecuencia, se pueden tomar varias medidas para aumentar las ganancias. Por ejemplo:

- ✓ Se pueden generar más ganancias si se puede identificar la relación entre los artículos comprados en diferentes transacciones.
- ✓ A y B se pueden colocar juntos para que cuando un cliente compre uno de los productos no tenga que ir muy lejos para comprar el otro producto.
- ✓ Las personas que compran uno de los productos pueden ser dirigidas a través de una campaña publicitaria para comprar el otro.
- ✓ Se pueden ofrecer descuentos colectivos en estos productos si el cliente compra ambos.

Tanto A como B se pueden empaquetar juntos. [9].

Fuentes Relacionadas a la caracterización de clientes.

Contrí, Saura, Molina, (2008), escogió una investigación por encuestas para realizar una investigación cuantitativa en la que tenía como objetivo diferenciar a grupos de consumidores estimando beneficios con respecto a sus distribuidores. El estudio consiste en observar a dos grupos que son: quienes comercializan el consumo (alimentación, calzado) y los de establecimientos quienes son los que mercadean los bienes para cada domicilio. Para trascender en el diseño de sus estrategias comerciales se obtuvo unas características de los consumidores. El desarrollo consiste en realizar una encuentra a cada persona que salía con sus compras, las características que contiene esta encuesta son demográficas como: el género del cliente, la edad de la persona, nivel de educación, antigüedad del cliente. Ver Ilustración 9 de las variables analizadas. Luego del levantamiento de información se realizó el análisis exploratorio con su respectiva limpieza de los datos como son: detección de outliers, valores faltantes, y posteriormente el análisis de componentes

principales (ACP), análisis de la varianza (ANOVA) y análisis de conglomerados en el software estadístico SPSS versión. [10].

VARIABLES ANALIZADAS Y PROCEDENCIA DE LAS ESCALAS DE MEDIDA UTILIZADAS

Variable	Adaptado a partir de
Beneficios de la relación	Gwinner *et al.* (1998)
Costes de cambio	Patterson y Smith (2001)
Valor percibido	Sweeny y Soutar (2001)
Calidad de servicio logístico	Mentzer *et al.* (2001) y elaboración propia
Intensidad percibida de uso de TIC por el minorista	Observatorio (2006) y elaboración propia
Calidad de la relación con la tienda	Wong (2004)
Confianza	Ball *et al.* (2006)
Compromiso	Hennig-Thurau (2004)
Actitud general y relativa	Bove y Johnson (2000)
Lealtad global	Srinivasan *et al.* (2002); Anderson y Srinivasan (2003)
Lealtad conductual	East y Sinclair (2000); Antón y Rodríguez (2004); East *et al.* (2005)

FUENTE: Elaboración propia.

Ilustración 9 Variables analizadas del estudio de Contrí, Saura, Molina, (2008)

Solarte, G & Ocampo, C. (2009, agosto), Presentan una investigación para el diagnóstico médico de enfermedades complejas, en este estudio se exploran los modelos clasificatorios de datos a través de los comportamientos de las personas como son: los clústeres, random forest y el PCA, análisis de componentes principales, usan metodologías de clasificación para ayudar a la toma decisiones de una manera más eficaz y anticipada. Para este estudio se utilizaron fuentes de datos de un estudio entre 100 registros de los cuales se les tomo 16 muestras distintas de centro del Saludcoop EPS de municipio de Santa Rosa - Chinchiná; donde obtuvieron características como: Colesterol, LDL, Triglicéridos, Peso, Altura y HDL, Peso Ideal, Grasa Piel, Presión diastólica y sistólica,HDL, ver datos de los componentes principales de la Ilustración 10. [11].

Matriz de componentes(a)						
	Componente					
	1	2	3	4	5	6
Edad	.423	-2.796E-02	.204	-.213	2.879E-02	.346
Peso	8.764E-02	.809	.301	-.141	.413	.160
Colesterol	.917	-.107	-.292	-1.025E-02	8.630E-02	9.594E-02
Triglicéridos	.602	.327	-4.917E-02	-.142	-.346	-.416
HDL	.248	-.565	3.514E-02	.170	.127	.669
LDL	.898	4.705E-02	-.324	-5.888E-02	6.440E-02	-8.839E-02
Peso Ideal	.310	-6.763E-02	.724	.171	.454	-.258
Altura	-.144	.814	-.260	-.254	6.812E-02	.390
Grasa piel	.216	-.141	.630	-.392	-7.896E-02	-4.308E-04
SystolicBP	6.533E-02	.374	.279	.598	-.318	.112
DiastolicBp	.285	.296	9.445E-02	.647	-.201	.107
Eje_R/min	-6.391E-02	9.795E-04	-.369	.324	.703	-.228
Método de extracción: Análisis de componentes principales. a 6 componentes extraídos.						

Ilustración 10 Matriz de componentes en estudio de Solarte, G & Ocampo, C. (2009)

Huerta Bustos, M. (2011), trabajó con datos de una tienda especialista enfocada en las categorías de perfumería, cuidado personal y cosmética, trabajó con respecto al marketing, es decir, se realizan promociones, actividades y descuentos a los clientes y no se prometen propuestas especiales de acuerdo con las distintas necesidades y tipos de consumidores. La empresa replantea sus estrategias y opta por el marketing focalizado, por esto decide realizar caracterización perfiles de clientes, por medio de una segmentación a priori por ciclo de vida y a posteriori una segmentación transaccional. La metodología desarrollada consigna de la realización de dos segmentaciones semejantes ver Ilustración 11; una segmentación transaccional que incluye dependencia con las promociones, presencia de canasta de compra, la media de monto y frecuencia de compra y otra segmentación por ciclo de vida que incluya aspecto como hijos, ocupación, edad y estado civil. Ambas segmentaciones dan origen a los perfiles de los clientes. La investigación condescendió encontrar y caracterizar 6 perfiles de clientes. [12].

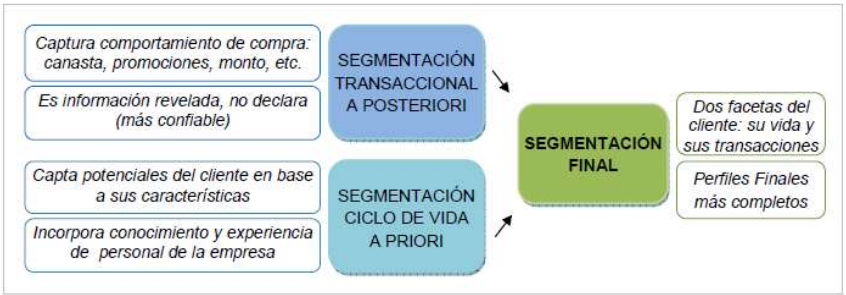


Ilustración 11 Metodología desarrollada en estudio Huerta Bustos, M. (2011)

Torres Villanueva, S. (2011), aborda el estudio de los clientes de menor escala de una compañía distribuidora de maquinaria Caterpillar, consistía en identificar gustos y preferencias de sus clientes concentrando información transaccional de estos para su análisis con el objetivo de obtener una propuesta de valor mejor y de facilitar la toma de decisiones de la compañía. La metodología que se desarrolló en este estudio consiste la realización de encuestas y luego la segmentación de los clientes. Se identifica las mejores características y el servicio optimo entregado, para obtener un conocimiento de satisfacción de sus clientes, encontrar oportunidades en el mercado y consecuentemente incrementar los objetivos de la compañía. Ver Ilustración 12. Dentro del desarrollo del estudio se evidenció segmentos con preferencias altas dando como resultado 4 grupos: Clientes Frecuentes (12%), consumidores que se caracterizan por su mayor presencia de compras; Clientes con Promesa (22%), los consumidores con una frecuencia no tan alta pero con promesa de regresar; Clientes Mercenarios (9%), aquellos consumidores que se fijan en los precios y que no se encuentran contentos con las compras y finalmente los Cliente Relacional (57%), quienes valoran la marca de la compañía y prometen una compra a un plazo largo. Se construye la solución de acuerdo con los resultados obtenidos: Repuesto disponible, Pagos, Tipos de Repuestos, Solución Esperada y Supervisión al consumidor. [13].

N°	Variables de Segmentación
1	Antigüedad del cliente
2	Tamaño de la empresa del Cliente
3	Industria/Rubro del cliente
4	N° Empleados
5	Flota de Equipos (en general)
6	Flota de Equipos (Caterpillar)
7	Importancia de la Flota en el Negocio del Cliente
8	Frecuencia de Compra de Repuestos
9	Frecuencia de Renovación de Equipos
10	Horas de Uso Mensual Promedio por Equipo
11	Disposición a pagar por sobre el precio de mercado en un repuesto Caterpillar
12	Disposición a pagar por sobre el precio de mercado en un Equipo Caterpillar
13	Porcentaje del Equipo que está dispuesto a pagar por la Mantención de la Máquina
14	Satisfacción Servicio Entregado
15	Satisfacción Servicio Técnico
16	Satisfacción Cumplimiento de Plazos
17	Satisfacción Opciones de Financiamiento
18	Satisfacción Disponibilidad de Repuestos
19	Nombre de la Empresa del cliente

Ilustración 12 Selección de características en estudio de Torres Villanueva, S. (2011)

Torrico, Gutiérrez, Cabezudo (2012), en el ambiente de compra online, la investigación sugiere que los consumidores que prefieren facturar online registran estimulaciones distintas a las personas que eligen comprar de una forma distinta como el tradicional, destacándose entre ellos principalmente obtener más variedad, comodidad y luego realizar compras recreativas (Rohm y Swaminathan, 2004). Ademá, a partir de la confianza, personalidad y características preferidas de la web también se muestran distintas entre cada uno de los clientes online (Prodanova y San Martín, 2012). Esto además sucede en el argumento móvil, donde las características únicas de las compras móviles, o m-commerce, (por ejemplo, pantalla chica, la capacidad de procesamiento de datos finitos, la ubicuidad, varios tipos de aplicaciones móviles y plataformas distintas) hacen que los deseos de los clientes móviles sean distintos respecto a los demas canales de distribución (por ejemplo, de las tiendas físicas, por catálogo y compras online) (Yang et al., 2012). [14]. Ver Ilustración 13.

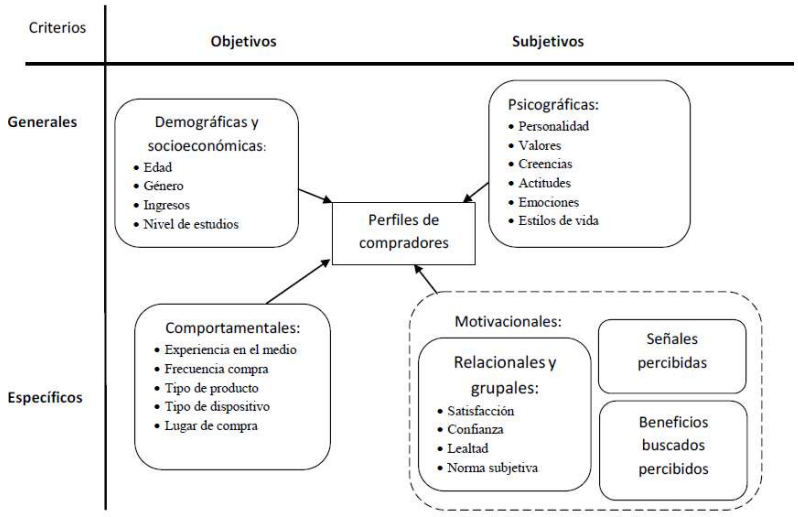
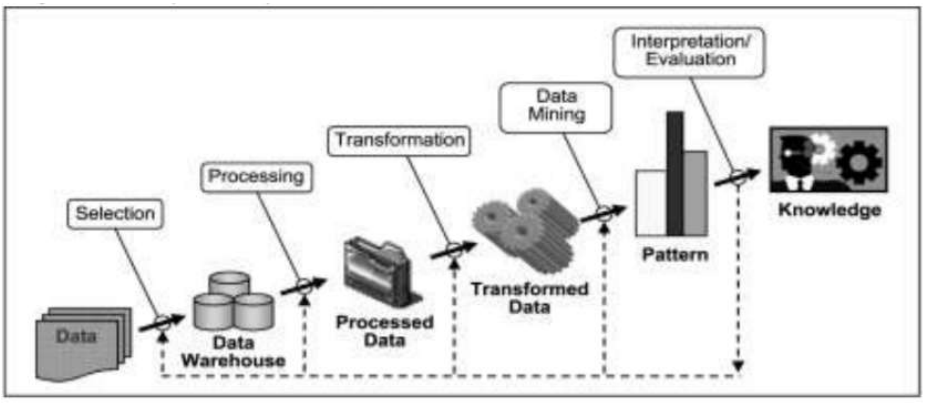


Ilustración 13 Variables relevantes de segmentación de compradores online y móvil, Torrico, Gutiérrez, Cabezudo (2012)

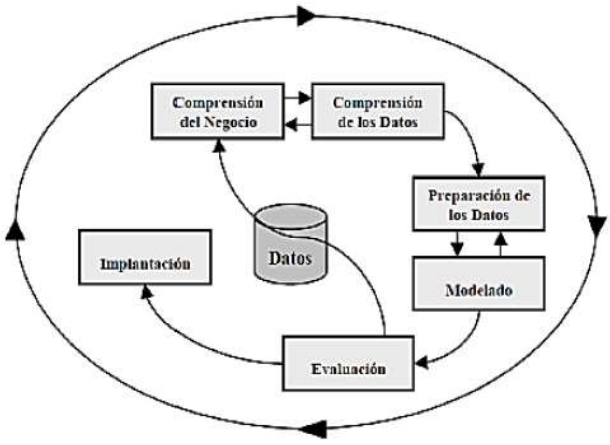
C. M. Cubides Proaños. (2013) En la empresa DPC Studio S.A.S ubicada en Bogota, Colombia, se desarrolló una investigación de data mining para determinar la clasificación de sus clientes. La compañía proporcionó información como: el id, nombre, descripción y tipo del producto que se llevan en la factura. Cumpliendo el objetivo de la investigación se analizaron los datos en donde se construyó un modelo no supervisado usando el algoritmo no supervisado de K-Means con lo cual se clasificaron para los clientes del dataset. Ver la Ilustración 14 en donde se visualiza el proceso de la investigación. La conclusión da como resultados 3 clúster que considera al portafolio de los productos y a los sectores identificados. [15].



Fuente: www.emeraldinsight.com

Ilustración 14 Etapas del proceso KDD en estudio de C. M. Cubides Proaños. (2013)

Morelo Tapias, K. A. (2014), propuso un sistema para la caracterización de perfiles de clientes de la compañía Zona T de donde se obtuvieron datos sociodemográficos para determinar patrones en los consumidores, donde finalmente resultaron 5 características cuantitativas y 6 cualitativas, el dataset registra once características y 180 consumidores en un periodo comprendido entre el año 2012 y 2013. Para el desarrollo del estudio de utilizó el algoritmo no supervisado de K-Means en donde se le realizo una adaptación construyendo un algoritmo nuevo denominado como K-SSE ver Ilustración 15 y con este algoritmo se derivó a un software diseñada en la web que fue desarrollado considerando las propiedades de los conceptos de la data mining. En la Ilustración 16 se visualiza el bosquejo de los casos. Se concluye en los resultados 3 segmentos de consumidores bien identificados y también se identifica a los productos con mayores compras. [16].



Fuente: (Gallardo, 2011)

Ilustración 15 Fases para la creación del modelo, Morelo Tapias, K. A. (2014)

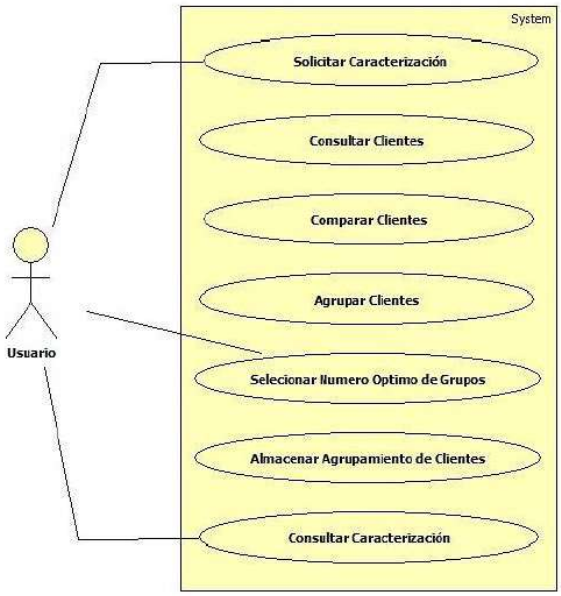


Ilustración 16 Casos de uso de la herramienta web, Morelo Tapias, K. A. (2014)

Abadía, & Patiño. (2020), Una compañía en Popayán dedicada a la venta del por mayor de productos de lácteos, realizó un estudio que trata de identificar la relación de sus ingresos de empresa con respecto a la venta que realiza. Adicional la compañía desea fidelizar a sus clientes de acuerdo con distribución que se obtenga. De acuerdo al problema se decide construir un modelo de caracterización de clientes, en donde se considera un algoritmo no supervisado y el análisis de marketing como es el RFM. El dataset a trabajar contiene 2837 registros que son clientes únicos y 112 características, el periodo de los datos son un año. Primero realizan el cálculo del score RFM mediante Excel para luego crear un modelo de clustering basado en algoritmo K-means, el algoritmo con mayor uso para determinar grupos de forma natural, la implementación fue realizada en lenguaje Python obteniendo finalmente 4 clusters dentro de sus datos como se muestra en la Ilustración 17. [17].

SEGMENTACION DE CLIENTES CON K-MEANS

Clúster	Clientes	Porcentaje	Monto T	Monto T%	Prom(R)	Prom(F)	Prom(M)	Clasificacion
0	187	7%	46.900.369,00 COP	1%	276	5	250.804,00 COP	Clientes Poco Aporte
1	1717	61%	6.354.262.864,00 COP	77%	42	42	3.700.793,00 COP	Clientes Buenos
2	12	0%	1.054.201.070,00 COP	13%	50	39	87.850.089,00 COP	Clientes VIP
3	921	32%	756.220.060,00 COP	9%	59	15	821.066,00 COP	Clientes Regulares
Total	**2837**	**100%**	**8.211.584.363,00 COP**	**100%**				

Ilustración 17 Segmentación por RFM y K-Means, Abadía, & Patiño. (2020)

2.3 Librerías y software utilizado

En el desarrollo de este proyecto aplicará el lenguaje de programación Python, en donde se procesará la limpieza del dataset, el análisis exploratorio de los datos, la selección de las características más relevantes, la reducción dimensional del dataset, el modelo de agrupamiento y los modelos de clasificación para entrenar los datos.

Python

El lenguaje de Python es un software distribuido y desarrollado libremente de código abierto, open-source. También se concreta como un lenguaje de programación con mayor horizonte, flexible, interpretado y con una sintaxis concisa y clara. Debido a que Python es un lenguaje orientado a objetos y con una programación dinámica se denomina como un lenguaje multiparadigma. [4]

NumPy. Está relacionado con funciones matriciales y numéricas. [4]

Matplotlib. Está relacionado para el desarrollo visual, es de gran aporte para la realización de los gráficos a gran dimensión y con diferentes formas y temas. [4]

Pandas. Se trata de la integración del análisis de la información y datos modelados como: creación, agrupación y manipulación de arreglos y gran volumen de datos, es flexible para las estructuras y dimensiones de los datos que están basados en los diccionarios de Pandas. [4]

Scikit-Learn. Está relacionado para los modelos supervisados y no supervisados de machine learning. Esta librería incluye la facilidad del desarrollo de algoritmos como: el análisis de regresión, también el análisis de clústeres, bosques aleatorios, algoritmo SVM, DBSCAN, y el algoritmo no supervisado de K-means. [4]

Los algoritmos que se aplicarán de la librería para el proyecto son: para el clustering K-means y de clasificación SVM, nearest neighbors, random forest. [4]

Júpiter.

El cuaderno de Python es una plataforma interactiva que se encuentra en la web, en donde permite la creación de archivos de distintos formatos y se puede compartir a distintas fuentes, es un cuaderno de código abierto en donde soporta cálculos numéricos y científicos, así como gráficos y caracteres de textos. El desarrollo y descripción de las líneas de los códigos se los realiza en la interfaz del cuaderno. Trabaja con librerías ya establecidas, funciones que ayudan al desarrollo de análisis de big data, ciencia de datos, análisis de negocio e inteligencia artificial, entre otros. [4]

CAPÍTULO 3

3. Diseño e implementación

El diseño de la solución considera la integración con el datawarehouse de la empresa, acoplada dentro de la infraestructura implementada para el procesamiento batch del datawarehouse como se muestra en la Ilustración 18, en este servidor se almacena la información la compañía de Retail, como las tablas de ventas, clientes con sus datos sociodemográficos y todos los productos que dispone dentro de su catálogo mediante un sistema de ERP, en la cual permite unificar u organizar los recursos de todas las áreas de forma modular para así, optimizar, automatizar y planificar los procesos dentro de la empresa. Esta fuente de datos que la utilizaremos como data de origen se encuentra estructurada mediante los procesamientos del modelado de datos.

La extracción de los datos se lo realizará en SQL Server; la transformación y carga de la información será desarrollada mediante el analysis services y será guardada en formato .csv. Una vez obtenida la data con la información requerida se procederá a realizar con Jupyter Notebook el análisis exploratorio de los datos para: detectar outliers, valores faltantes y tipo de datos correctos para realizar la limpieza del dataset, y observar el comportamiento de ventas de los clientes evidenciando características relevantes para el desarrollo del modelo de caracterización.

La construcción del modelo de caracterización consiste en el desarrollo de dos algoritmos de machine learning: el no supervisado, clustering, que, a partir de las características seleccionadas relevantes, identifica grupos naturales de clientes que están formados de acuerdo a su comportamiento de ventas, y el supervisado, modelo elegido entre; naive bayes, KNN, SVM, arboles de decisión y redes neuronales. El entrenamiento se realizará a partir de la partición (train y test) del dataset. Luego se entrenará y clasificará los grupos de clientes encontrados en el clustering. Finalmente se segmentarán y perfilarán todos los clientes que han realizado compras en la compañía.

Posteriormente los resultados de los grupos se presentarán mediante visualizaciones en una herramienta de BI en la nube (Power BI), con conexión on premise data gateway hacia el servidor de base de datos del datawarehouse. Esta transferencia de datos se resguarda a través de Azure Service Bus. Para facilitar la protección de la información que se encontrará en la nube se limitaran las credenciales que se les dará a los encargados del enlace y solo podrán ser comprendidos en el servidor de la on premise data gateway. Ver ilustración 18.

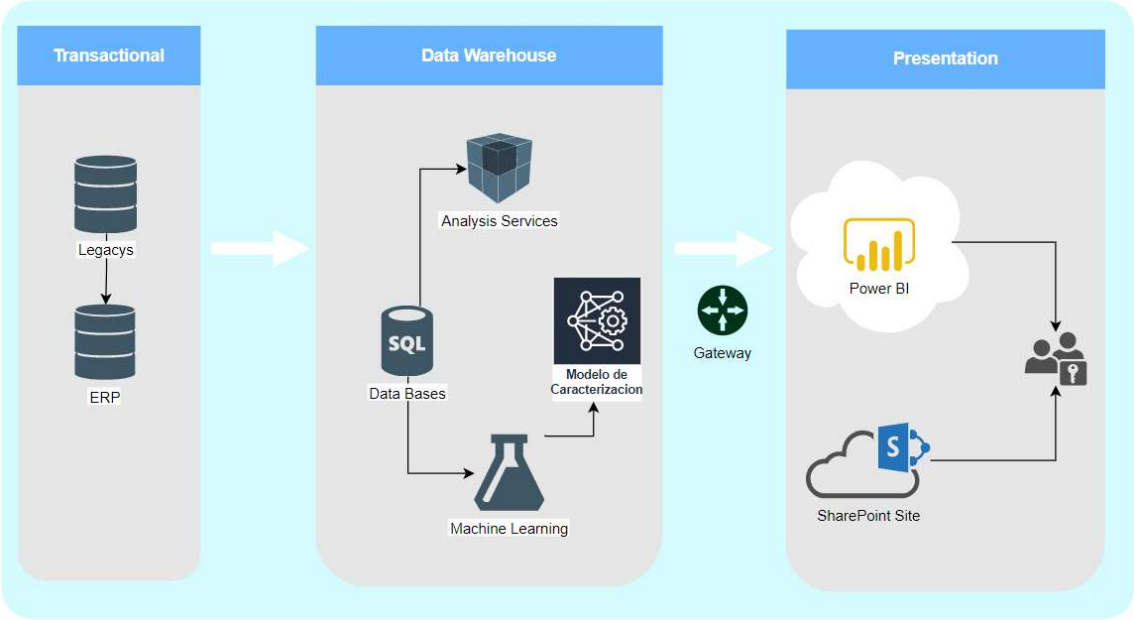


Ilustración 18 Diseño de Arquitectura de la solución

3.1 Fase de Recolección

Los datos que se utilizará para el desarrollo de este proyecto corresponden a la información de ventas proporcionada por la empresa de Retail, entre el periodo de enero 2015 hasta diciembre 2020. Los datos son estructurados y se guardan en el servidor del datawarehouse de la compañía.

El proceso de extracción de los datos, transformación y carga (ETL), se lo realizará con la herramienta de SQL Server ya que nos permite trabajar en modo cliente-servidor (ERP), soporta procedimientos almacenados, y además nuestros datos originales se encuentran estructurados, ya que la información para nuestro estudio será un modelo relacional.

Las tablas de información seleccionadas son: facturación, clientes, productos, forma de pago, y tiempo. Se preseleccionarán todas las características dentro de cada tabla seleccionada para realizar a priori el análisis de negocio y análisis estadístico descriptivo, para determinar las características que conformarán el dataset final. Los datos serán exportados en formato .csv para su respectivo análisis.

El dataset finalmente contiene las ventas por: fecha, provincia y ciudad del cliente, edad, género y estado civil, forma y medio de pago, información de productos como; línea, grupo, marca, código y descripción del producto, monto de venta y unidades vendidas, la granularidad es nivel de facturas. Contiene 1,761.308 Registros y 23 características, de las cuales 17 son categóricas, 5 son numéricas y una de tiempo, es decir, presentamos una data mixta o combinada.

El detalle de las características del dataset se muestra en la Tabla 1:

Tabla 1 Detalle de características y su descripción dataset

Características	***Descripción***
FECHA	*Fecha de la venta*
CANAL	*Canal de venta (comercio electrónico, tiendas físicas y call center)*
FACTURA	*Numero de factura*
Id_Cliente	*Código de cliente*
PROVINCIA_CLIENTE	*Provincia del cliente*
CIUDAD_CLIENTE	*Ciudad del cliente*
EDAD	*Edad del cliente*
GENERO	*Descripción del sexo del cliente (Masculino /Femenino)*
ESTADO_CVIL	*Descripción del Estado civil del cliente (Soltero /Casado/Viudo/Divorciado/Unión Libre)*
FORMA_PAGO	*Forma de pago de la venta*
MEDIO_PAGO	*Medio de pago de la factura*
LINEA	*Línea del producto (categoría)*
GRUPO	*Subgrupo del producto (subcategoría)*
MARCA	*Marca del producto*
COD_PRODUCTO	*Código del producto (categoría)*
PRODUCTO	*Descripción del producto (categoría)*
combo	*Si el producto es combo (0=si;1=no)*
promocion	*Si el producto está en promoción (0=si;1=no)*
regalo	*Si el producto es un regalo (0=si;1=no)*
UNIDADES_VENDIDAS	*Número de unidades vendidas del producto*
VENTA_TOTAL	*valor en dólares de la venta*
DESCUENTO	*Valor descontado en la factura a nivel de ítem de factura*

[1] Total, características: 23 (17 categóricas, 5 características numéricas y 1 datetime).
[2] Total, registros: 1,761.308 (transacciones).
[3] Tiempo comprendido del análisis: año de venta, 2015 – 2020
[4] Tamaño del dataset: 322.5+ mb.
[5] Granularidad: a nivel de facturas.

El tamaño del dataset es de 266 MB, tamaño considerable para guardarlo en formato .csv, por lo que se distribuye toda la data por cada año de venta (2015 al 2020), es decir se tendrá 6 archivos .csv, (un archivo por año) ver anexo 2.

Luego de la recolección de los datos, se realiza la lectura de los 6 archivos .csv en la hoja de trabajo de jupyter notebook, para después concatenarlos en un solo dataset y posteriormente se eliminará los archivos anteriores para evitar el consumo en memoria en el pc. Las librerías que se van a utilizar dentro de este proyecto se la pueden visualizar en el anexo 1.

Una vez concatenada los archivos, se procede a realizar el análisis exploratorio de los datos para empezar con la limpieza de esta.

La limpieza de la información consiste en identificar, corregir y/o depurar registros incompletos, duplicados, incorrectos, imprecisos, outliers y mal estructurados. Adicional, se comprobarán que todas las características del dataset tengan el tipo y formato que corresponden, con el correcto análisis de limpieza se obtiene mejores resultados.

Para el proceso de limpieza se empezará a realizar las visualizaciones de las características numéricas para identificar la presencia de outliers, para lo cual se graficará los boxplot de aquellas características.

Se tomaron las características de; VENTA_TOTAL, DESCUENTO, UNIDADES_VENDIDAS, y EDAD y se graficaron los boxplot de cada uno, con el objetivo de evidenciar la presencia de algunos valores aberrantes o outliers. En el boxplot de la VENTA_TOTAL se observa una venta máxima cercana a $60 mil y valores que superan el tercer cuartil ($787), estos datos podríamos analizarlos como outliers. En la característica de DESCUENTO se observan valores que superan los $25mil y valores altos muy superiores al rango intercuartil, todos estos valores también podrían ser analizados como outliers. En UNIDADES_VENDIDAS se observa un valor de 140 unidades esto quiere decir que se registran ventas con unidades altas, lo cual podría alterar los resultados y para ello se podrían excluir estos valores del dataset. En la característica de la EDAD, se evidencia la presencia de edades negativas, esto debido al error del ingreso de la información, y se observa edades de 120 años, por lo que también se considerará estos valores como outliers. Ver Ilustración 19.

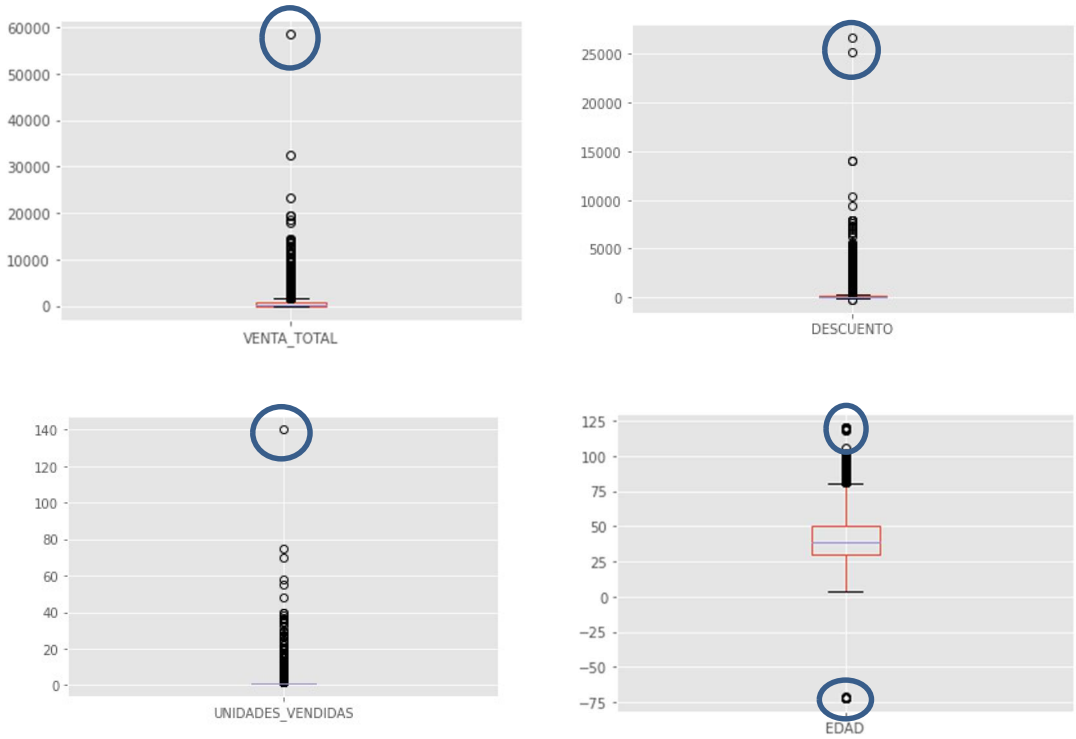


Ilustración 19 Presencia de Outliers en las características numéricas del dataset

Otra forma para evidenciar estos valores outliers es graficando la distribución de densidad de estas características numéricas. En la característica de la EDAD, se puede observar que las edades están concentradas entre 16 y 100 años, y se evidencia los valores atípicos que están en los extremos de la distribución. En la característica de VENTA_TOTAL se observa que las ventas están concentradas hasta menos de $10mil, a partir de este número consideraremos valores atípicos que distorsionan la distribución. En la característica de DESCUENTO la distribución se encuentra sesgada hacia la derecha del observador con valores concentrados hasta los $10mil, superior a este valor será considerado como outliers. Ver en la Ilustración 20.

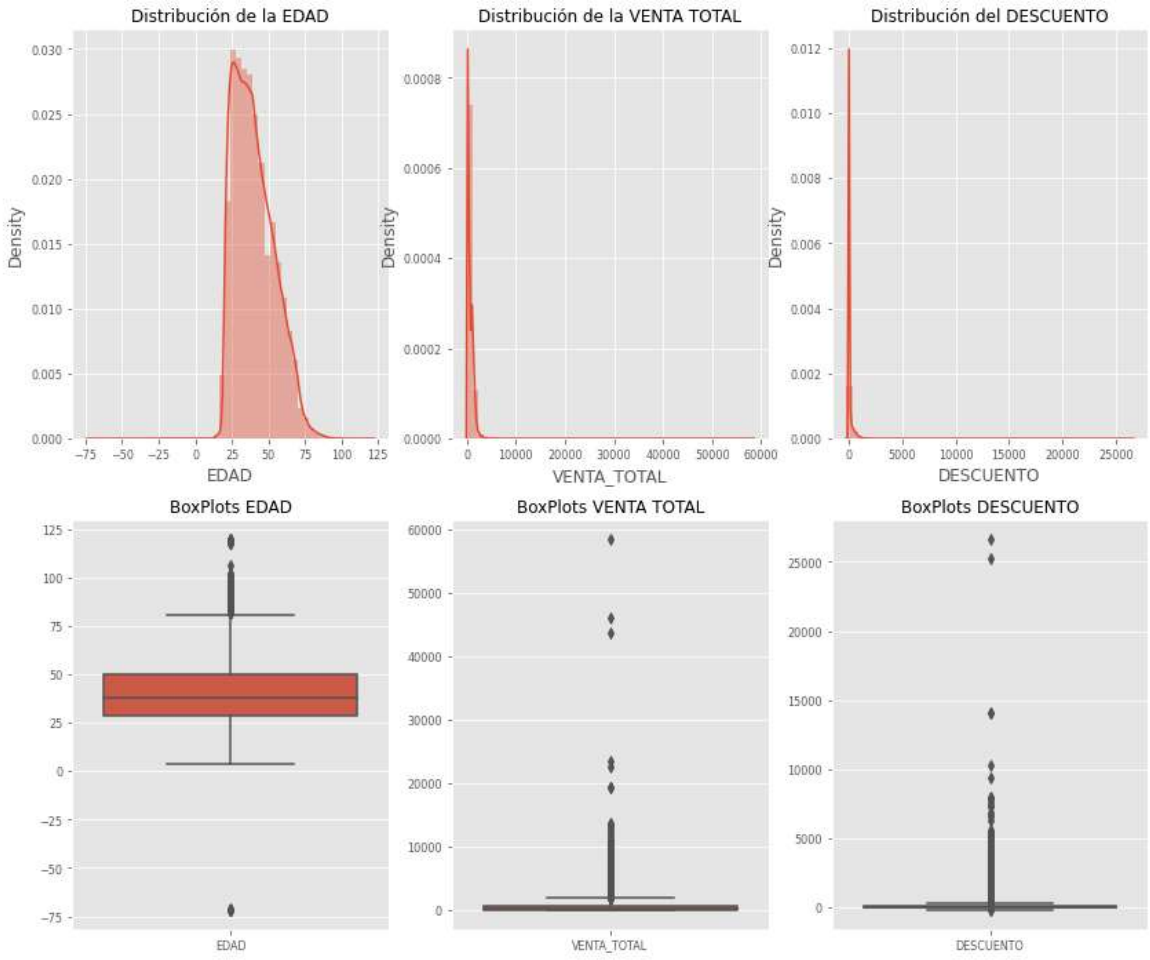


Ilustración 20 Graficas de densidad y BoxPlots de las características numéricas

Luego de detectar gráficamente los outliers, se procede a cuantificar estos valores utilizando el rango intercuartil. Ver Tabla 2:

Tabla 2 Tabla de rango intercuartil para característica numéricas

Rango Intercuartil	***Características***			
	EDAD	***VENTA_TOTAL***	***UNIDADES_VENDIDAS***	***DESCUENTO***
Q1 Quantile	*29*	*40,81*	*1,00*	*3,38*
Q2 Quantile	*38*	*172,24*	*1,00*	*27,93*
Q3 Quantile	*50*	*787,12*	*1,00*	*113,61*
Q4 Quantile	*120*	*58.485,00*	*185*	*26659,36*

Se puede observar en la característica de la EDAD que el tercer cuartil es 50 años y el cuarto cuartil es de 120 años, este último valor es considerado como un outliers, por lo que se filtrará la data hasta 100 años. En la característica de VENTA_TOTAL a pesar de que el tercer cuartil es $787,12 se excluirán los valores

superiores a $15mil, esto de acuerdo con el análisis de la gráfica del boxplot. En UNIDADES_VENDIDAS el valor máximo es de 140 unidades, a pesar de que el primer, segundo y tercer cuartil es 1 unidad, se excluirán las unidades mayores a 40, este valor de acuerdo con el análisis de su boxplot. En DESCUENTO, se considerará los outliers a los valores superiores a $10mil, valor analizado de acuerdo con la gráfica del boxplot. Ver en la Ilustración 20.

Una vez identificado estos valores atípicos dentro del dataset, se procederá a detectar valores faltantes dentro de cada característica, para lo cual se utilizará la función de *isna()* que proporciona la librería de *pandas*. Ver Ilustración 21.

```
FECHA                   0
UNIDADES_VENDIDAS       0
regalo                  0
promocion               0
combo                   0
PRODUCTO                0
COD_PRODUCTO            0
MARCA                   0
GRUPO                   0
LINEA                   0
VENTA_TOTAL             0
plazo_pago              0
FORMA_PAGO              0
des_sexo                0
des_estado_civil        0
CIUDAD_CLIENTE          0
PROVINCIA_CLIENTE       0
Id_Cliente              0
FACTURA                 0
CANAL                   0
MEDIO_PAGO              0
DESCUENTO               0
EDAD               114175
dtype: int64
```

Ilustración 21 Tabla de características con valores faltantes

Se observa que para cada característica no existe la presencia de valores ausentes (N/A), es decir, cada dimensión tiene al menos un registro.

Se realizó un análisis de conteo de los valores de las características del dataset para determinar valores inconsistentes que no pertenecen a la característica. En la Ilustración 22. Observamos que en la característica FECHA se registra 2182 valores únicos, la fecha de venta que más se repite es 29 de noviembre del 2019 con 6231 veces. En la característica CANAL, se registra 3 valores únicos, el canal tiendas tiene mayor frecuencia. En la característica des_sexo, se observa 3 valores únicos, masculino es el que más se repite. MEDIO_PAGO registra 6 valores únicos, credicard,

que es el crédito directo de la compañía, tiene mayor frecuencia dentro de la característica y GRUPO, es el grupo de producto, registra 190 grupos de productos únicos, televisores que es el más frecuente en las ventas.

	count	unique	top	freq
FECHA	1387539	2182	29/11/2019	6231
CANAL	1387539	3	TIENDAS	1310845
FACTURA	1387539	987693	0000017PW42090	195
PROVINCIA_CLIENTE	1387539	28	GUAYAS	802401
CIUDAD_CLIENTE	1387539	325	GUAYAQUIL	579559
des_estado_civil	1387539	6	SOLTERO	900480
des_sexo	1387539	3	MASCULINO	759313
FORMA_PAGO	1387539	2	CREDITO	829309
MEDIO_PAGO	1387539	6	CREDICARD	828571
LINEA	1387539	25	BLANCA	326484
GRUPO	1387539	190	TELEVISORES	194358
MARCA	1387539	245	SMC	188290
COD_PRODUCTO	1387539	50878	4655	22082
PRODUCTO	1387539	9182	LICUADORA 1,25LT 3VEL OST CROM	22733

Ilustración 22 Tabla de valores únicos para características categóricas

Se observa que las características des_sexo y des_estado_civil tienen en sus registros la opción NO REGISTRA que corresponden a valores que no fueron registrados en el ingreso de la información y se etiquetaron bajo esa descripción. En el barplot de la característica des_sexo, se observa que masculino tiene mayor participación (50%, 986mil registros), seguido de femenino (37%, 654mil registros) y con muy pocos datos no registra (7%, 120mil registros), este último no es parte del análisis por lo que se excluirá dentro de la limpieza. En el barplot de la característica des_estado_civil se observa que soltero tiene mayor participación (68%, 1,193mil registros), seguido de casado (20%, 358mil registros), no registra no será incluido para el análisis y se lo quitará en el proceso de limpieza. Ver

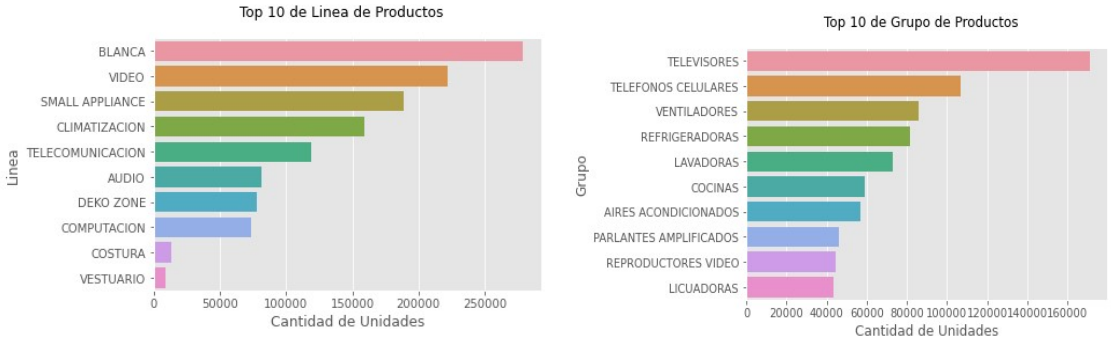


Ilustración 31, Tabla 3 y Tabla 4.

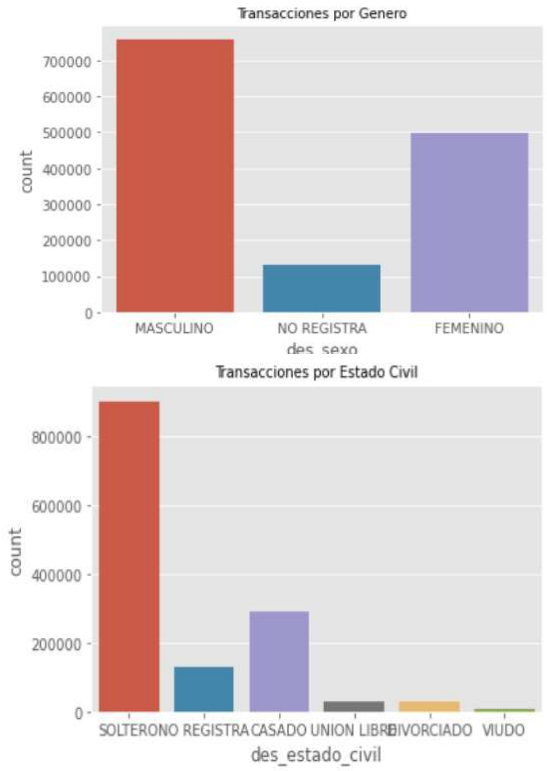


Tabla 3 Tabla de frecuencia del Género

	des_sexo	
	Cantidad	***Porcentaje***
FEMENINO	*654.378*	*37%*
MASCULINO	*986.006*	*56%*
NO REGISTRA	*120.924*	*7%*
TOTAL	***1.761.308***	***100%***

Tabla 4 Tabla de frecuencia del Estado Civil

	des_estado_civil	
	Cantidad	***Porcentaje***
SOLTERO	*1.193.892*	*68%*
CASADO	*358.597*	*20%*
UNION LIBRE	*40.302*	*2%*
DIVORCIADO	*38.346*	*2%*
VIUDO	*8.987*	*1%*
NO REGISTRA	*121.184*	*7%*
TOTAL	***1.761.308***	***100%***

Ilustración 23 Distribución de Transacciones por Genero y Estado Civil

Dentro de la limpieza de los datos se comprueba el tipo de características que registra el dataset. Este paso es importante para los análisis posteriores ya que no podremos calcular métricas de estadísticas a características categóricas, para lo cual se utilizará la función *info()* de la librería *pandas,* que describe la información del dataframe como: total de registros, total de columnas, tamaño en memoria y los tipos para cada características. Se observa en la Ilustración 24 los tipos para cada característica (Dtype), como, por ejemplo: CANAL tipo *object* (categórico), EDAD tipo *float64* (numérico), UNIDADES_VENDIDAS tipo *int64* (entero). Se evidencia que la

característica FECHA registra el tipo *object* y no tiene tipo *datetime*, para lo cual se corregirá dentro de la limpieza.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1761308 entries, 0 to 240415
Data columns (total 23 columns):
 #   Column             Dtype
---  ------             -----
 0   FECHA              object
 1   CANAL              object
 2   FACTURA            object
 3   Id_Cliente         int64
 4   PROVINCIA_CLIENTE  object
 5   CIUDAD_CLIENTE     object
 6   des_estado_civil   object
 7   des_sexo           object
 8   EDAD               float64
 9   FORMA_PAGO         object
 10  MEDIO_PAGO         object
 11  plazo_pago         int64
 12  LINEA              object
 13  GRUPO              object
 14  MARCA              object
 15  COD_PRODUCTO       object
 16  PRODUCTO           object
 17  combo              int64
 18  promocion          int64
 19  regalo             int64
 20  UNIDADES_VENDIDAS  int64
 21  VENTA_TOTAL        float64
 22  DESCUENTO          float64
dtypes: float64(3), int64(6), object(14)
memory usage: 322.5+ MB
```

Ilustración 24 Tabla de tipos de datos de las características

Después del proceso de limpieza de los datos, finalmente tenemos el dataset con un total de registros de 1,627.347, 92% del dataset inicial. Debido al orden de los filtros, los outliers más excluidos corresponden a la característica de la EDAD 115.181 registros, 8% del dataset inicial. Ver

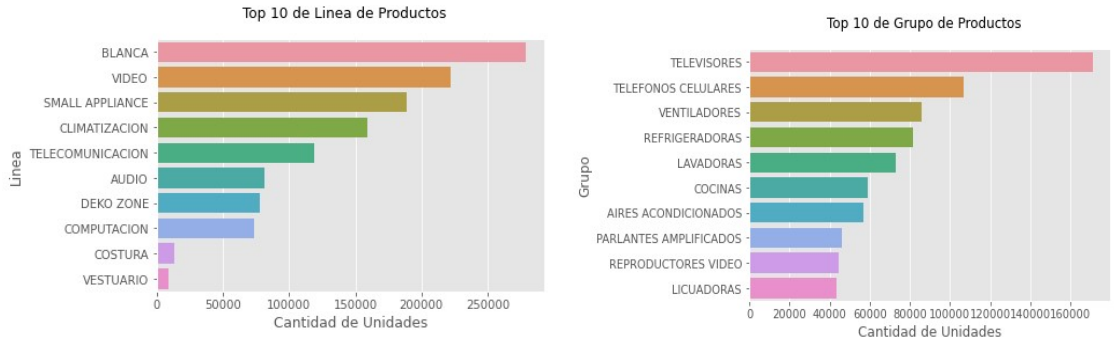
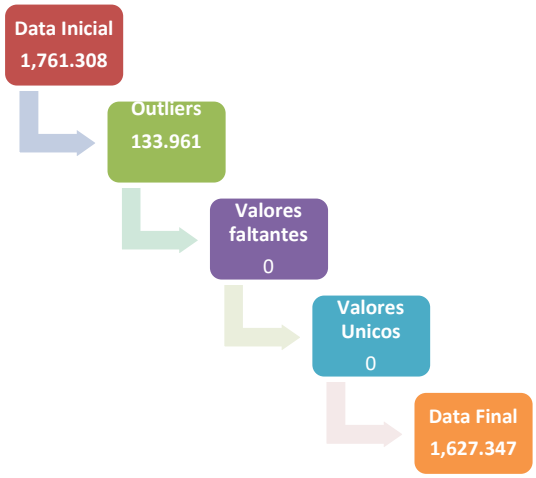


Ilustración 315 y Tabla 5.

Tabla 5 Tabla de distribución de resultados de limpieza dataset

	Cantidad	***Porcentaje***
DATA INICIAL	***1.761.308***	***100%***
CLEANING	*133.961*	*8%*
EDAD	*115.181*	*7%*



VENTA_TOTAL	*5.160*	*0%*
DESCUENTO	*6.544*	*0%*
UNIDADES_VENDIDAS	*1*	*0%*
des_sexo	*6.717*	*0%*
des_estado_civil	*358*	*0%*
DATA FINAL	***1.627.347***	***92%***

Ilustración 25 Resultados de la limpieza del dataset

3.2 Fase de Análisis

Luego de tener nuestro dataset limpio, en esta fase se realiza el análisis exploratorio de los datos para evidenciar el comportamiento de venta de los clientes por sus formas de pago, canal de compra, líneas de productos y montos de venta a nivel de las características demográficas del cliente.

Para profundizar el exploratorio y posterior enriquecer el conjunto de características de entrada para el modelo de agrupación se crearon características agrupadas provenientes de las características numéricas, por ejemplo, la característica EDAD, VENTA_TOTAL, DESCUENTO y PLAZO se la agruparon de acuerdo con sus valores aproximados a los rangos de cuartiles. Ver Tabla 6.

Tabla 6 Tabla de rango intercuartil para el dataset limpio

Rango Intercuartil	*Características*				
	EDAD	*VENTA_TOTAL*	*% DSCTO*	*PLAZO*	*MONTH*
Q1 Quantile	*29*	*37,57*	*5%*	*0*	*3*
Q2 Quantile	*38*	*159,37*	*30%*	*12*	*6*
Q3 Quantile	*50*	*753,09*	*50%*	*18*	*9*
Q4 Quantile	*100*	*11.498,76*	*90%*	*60*	*12*

Las características %DSCTO y MONTH fueron calculadas previamente, %DSCTO es el porcentaje de descuento que ha tenido el cliente de acuerdo con la venta que realizó, MONTH es el mes en donde realizó la venta. Una vez calculados los rangos intercuartil se agrupan los valores de cada característica. Ver Tabla 6.

Tabla 7 Tabla de grupos de rangos

Tabla Rango Venta	
Rango	***Descripción***
< 100	*< $100*
100 - 400	*$100 - $400*
400 - 1100	*$400 - $1100*
> 1100	*> $1100*

Tabla Rango Temporalidad	
Rango	***Descripción***
1-3	*1trim*
4-6	*2trim*
7-9	*3trim*
10-12	*4trim*

Tabla Rango Descuento	
Rango	***Descripción***
0	*0*
1-30	*1-30*
30-50	*30-50*
> 50	*> 50*

Tabla Rango Edad	
Rango	***Descripción***
< 30	*Jóvenes*
30-50	*Adultos*
>= 50	*Adultos Mayores*

Tabla Rango Plazo	
Rango	***Descripción***
0	*0 meses*
1-6	*1m - 6m*
6-12	*6m - 12m*
12-18	*12m - 18m*
>=18	*>= 18 m*

Se observa que la EDAD se agruparon en: “Jóvenes” menor a 30 años, “Adultos” entre 30 y 49 años y “Adultos Mayores” mayor o igual a 50 años. El PLAZO se agrupo en; “0m” 0 meses, “1m-6m” entre 1 y 6 meses, “6m-12m” entre 6 y 12 meses, “12m-18m” entre 12 y 18 meses, y “>=18m” mayor o igual a 18 meses. La VENTA TOTAL se agruparon en: “<100” menor a $100, “100-400” entre $100 y $400, “400-1100” entre $400 y $1100, y “>1100” mayor a $1100.

La estadística descriptiva de algunas de estas características agrupadas se puede observar en la

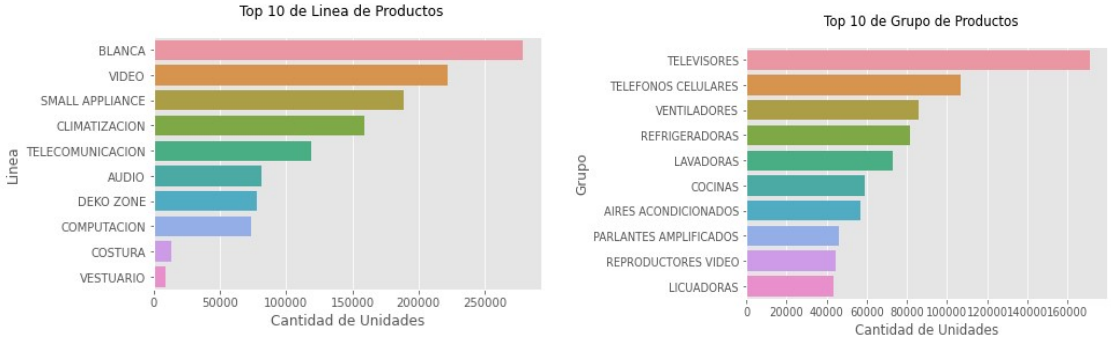


Ilustración 316.

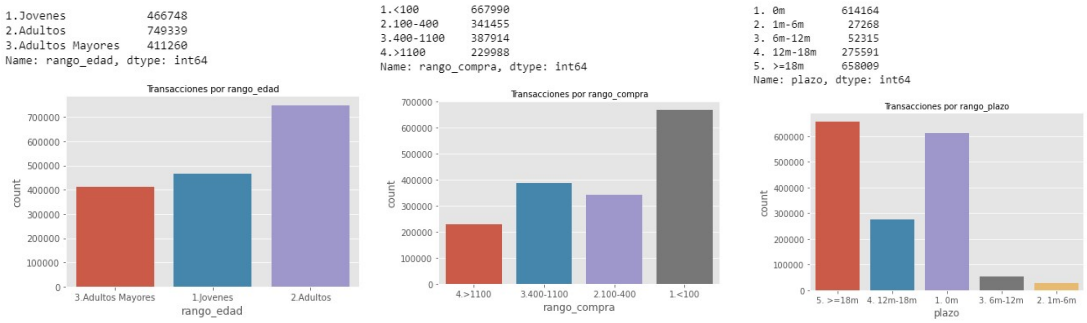


Ilustración 26 Características numéricas agrupadas por sus cuartiles

En el rango de edad se observa que los “Adultos” registran mayor transacción de venta (46%), seguido de los “Jóvenes” (29%) y “Adultos Mayores” (25%). En el rango de compra, se visualiza que las ventas menores a $100 tienen una mayor participación del 41%, seguido de las ventas comprendidas entre “$400 - $1100” con 24%, con el 21% las ventas comprendidas entre “$100 - $400” y por ultimo las ventas mayores a $1100 con una participación del 14%. En el rango del plazo se muestra que los diferidos mayor o igual a 18 meses registran mayores transacciones (40%), seguido de los plazos a 0 meses (38%) y “12m – 18m” (17%).

Para el análisis de las características categóricas se graficaron los barplot con el objetivo de comparar los valores de los grupos. En el barplot de la característica DES_SEXO se observa mayor presencia de hombres vs mujeres (60,1% vs 39,9%). Para la característica ESTADO_CIVIL se muestra una mayor participación en los clientes solteros (72,8%), seguido de casado (21,9%) y unión libre (2,5%), es decir, los que más compran son los solteros. En la característica CANAL, se observa una diferenciada participación del canal de tiendas físicas (95,1%) vs tiendas y web, es

decir, el lugar por donde se realiza la mayor parte de la venta es por las distintas tiendas físicas de la compañía. En el barplot de la FORMA_PAGO se visualiza que crédito registra mayor participación que contado (59,7% vs 40,3%), es decir, crédito es la forma más utilizada al momento de la venta. Podremos decir que, la mayoría de los clientes que más compran son: masculinos solteros que compran en la tienda física en forma de pago a crédito. Ver Ilustración 27.

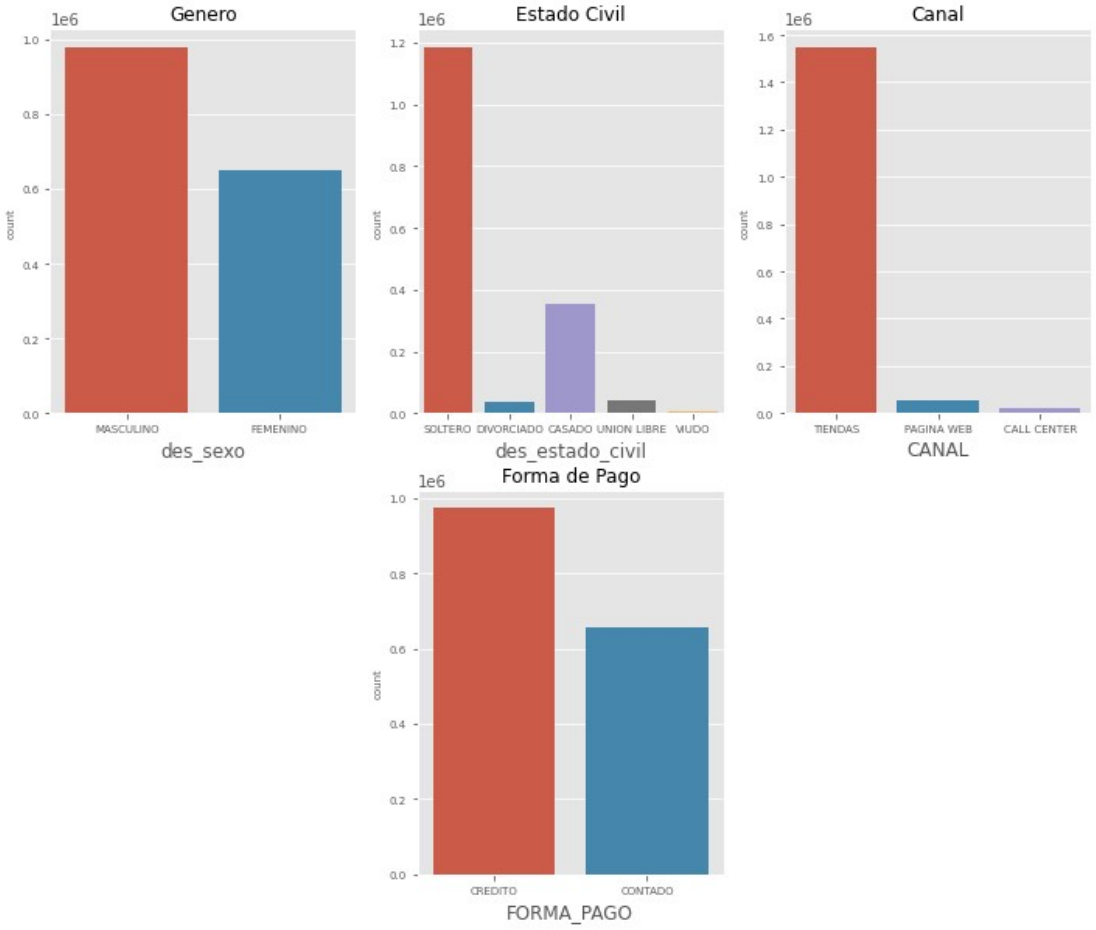


Ilustración 27 Distribución de características categóricas

Para las características numéricas se graficaron la distribución y sus respectivos boxplot con el objetivo de visualizar simetría, sesgos de valores y cuantificar sus datos a través de sus cuantiles. En la distribución de la característica de EDAD, se observa que los valores se encuentran asimétrico hacia la izquierda del observador, es decir, se evidencia compras de clientes jóvenes, y en el boxplot se muestra el valor de la media es 39,3 años y el tercer cuartil alrededor de los 50 años. En la característica de la VENTA_TOTAL, se observa que los valores siguen una distribución asimétrica positiva,

es decir, el valor de la mediana ($159,3) es menor que el valor de la media ($451,3), se evidencia compras de menor monto, en el boxplot se cuantifica que el valor del primer cuartil es de $37,5 y el tercer cuartil es $753,0. Con la característica DESCUENTO, la distribución muestra una asimetría hacia la izquierda, es decir, no se evidencia muchos descuentos dentro la venta, en el boxplot se registra el valor de la media $118,1 y el valor máximo de $88,0. Ver Ilustración 28.

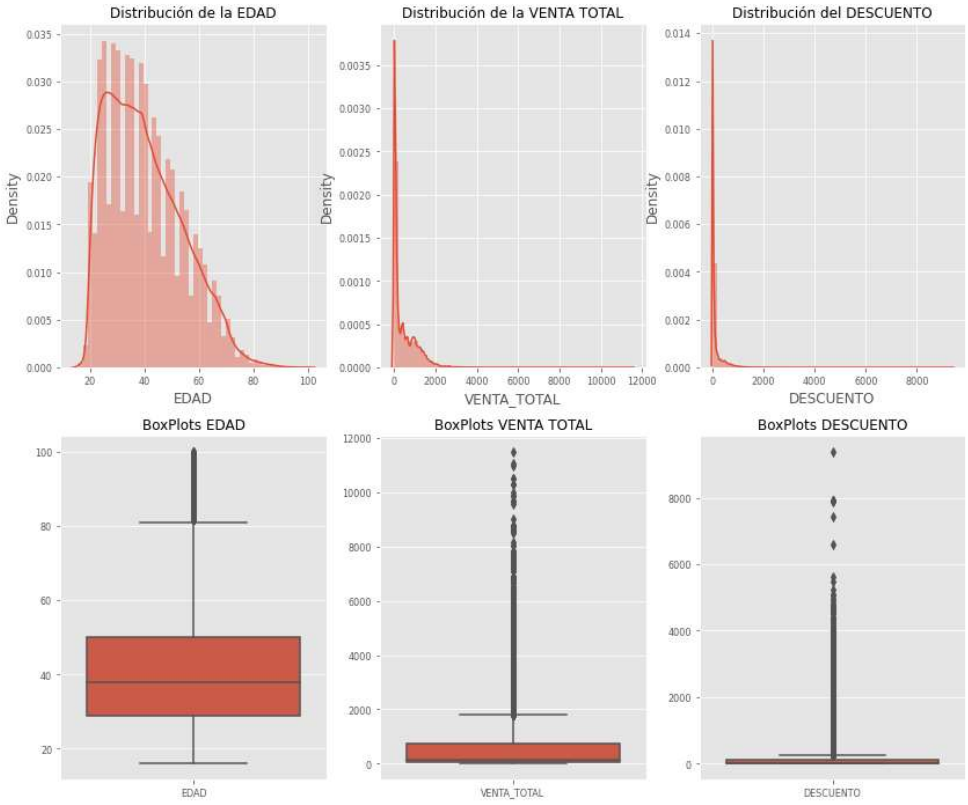
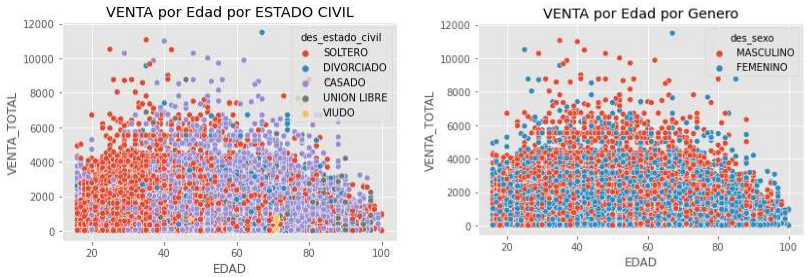


Ilustración 28 Distribución de densidad y boxplot para Edad, Venta y Descuento

Con el objetivo de visualizar la dispersión entre dos características, ventas a nivel demográfico, se grafica los scatterplot. Ver la Ilustración 29



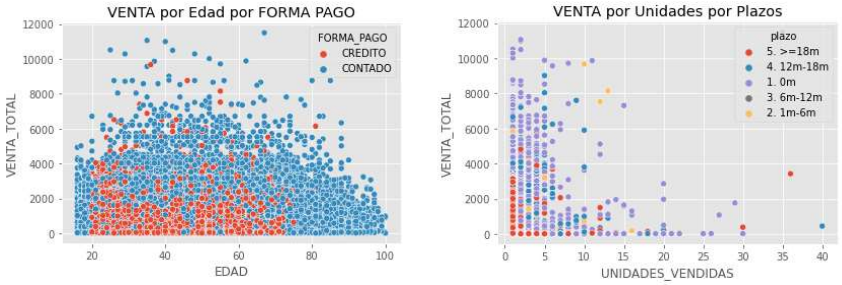


Ilustración 29 Distribución de Venta a nivel demográfico.

Se observa en el scatterplot por estado civil que las ventas se encuentran de cierta forma agrupadas, la mayoría de los clientes solteros registran edades menores a 40 años y ventas menores a $6000, y en mayor dispersión los clientes casados mayor a 40 años con ventas concentradas entre $2000 y $4000. La venta promedio de los solteros es $435.5 vs los casados $501,9. A nivel de género, las ventas de masculino vs femenino se encuentran dispersas a lo largo de la edad. Por su forma de pago, se visualiza que los clientes que compran a crédito están concentrados entre las edades de 18 y 70 años, esto debido por la política de crédito de la compañía, la venta promedio a crédito es $503,5 vs a contado $373,8. En el scatterplot de los plazos se observa que las transacciones están dispersas a menores unidades, a menor plazo menor unidad mayor venta.

Se realiza un heatmap o mapa de calor para distinguir mediante los colores más fuertes donde se genera el mayor volumen de ventas a nivel demográfico. En el primer heatmap se observa que la mayor cantidad de compras lo realiza los clientes masculinos con un rango de compra menor a $100, mientras que la menor cantidad de compras lo realiza las clientes femeninas con un rango de compra mayor a $1100. En el segundo heatmap se visualiza que la mayor venta promedio lo realizan los clientes masculinos casado y divorciados, mientras que la menor venta promedio lo realizan los clientes viudos. Ver Ilustración 30.

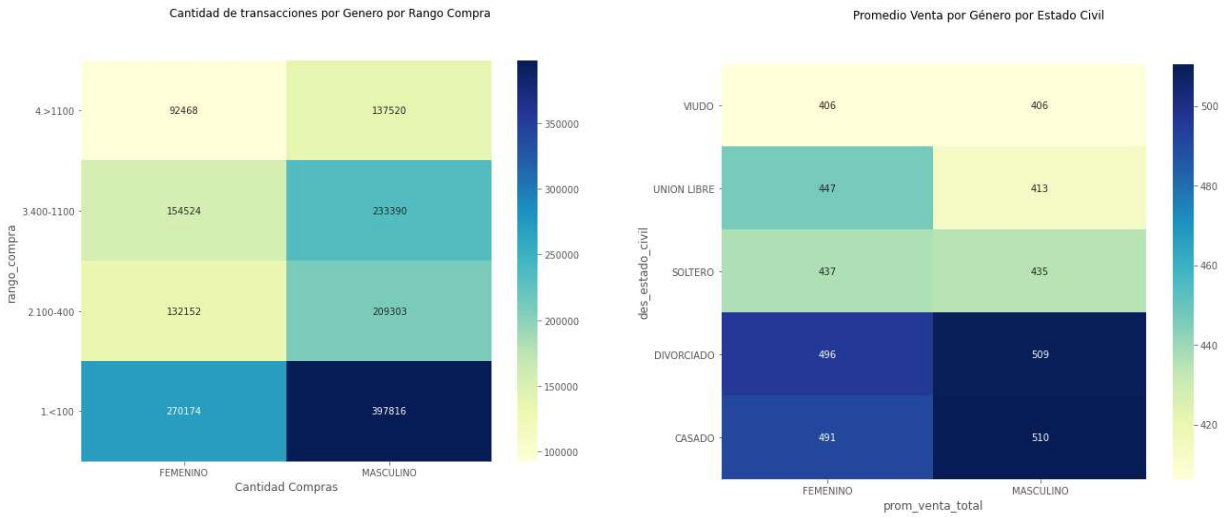
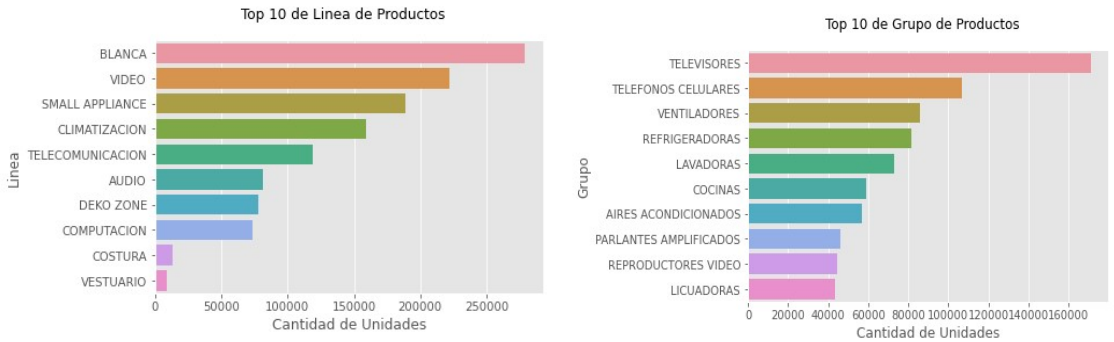


Ilustración 30 Comportamiento de compras a nivel demográfico

El catálogo de productos de la compañía es variado y amplio, se registran artículos como: televisores, aires acondicionados, neveras, lavadoras, computadoras, play stations, parlantes, productos pequeños como: licuadoras, tostadoras, plancha de ropa, artículos de hogar, artículos para bebes, entre otros, por lo que estos productos están agrupados por familias o niveles: 25 líneas de productos (blanca, audio tecnología, entre otros), 190 grupos, 245 marcas y 50,878 sku.

Para el análisis exploratorio por productos se graficaron los barchart o diagrama de barras ya que el objetivo es comparar las cantidades vendidas por líneas de productos. En el primer barchart se observa el top 10 de las líneas de productos en donde; la línea Blanca, Video y Small Appliance son los más vendidos. Para el segundo barchart se visualiza el top 10 de los grupos de productos, en donde los Televisores, Teléfonos Celulares y Ventiladores lideran como los grupos de productos que más se venden. Ver Ilustración 31.



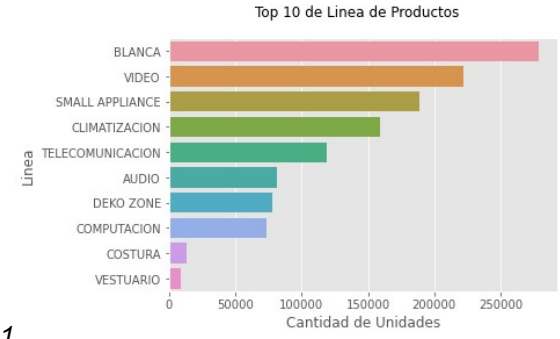


Ilustración 31

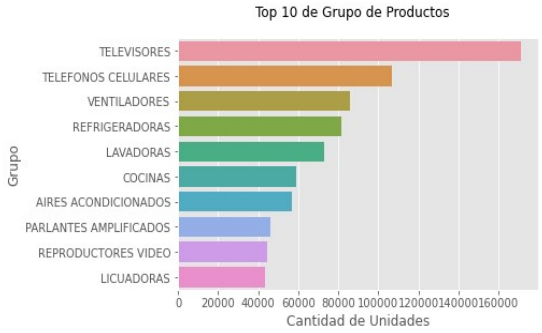


Ilustración 31 Comportamiento de las líneas y grupos de productos

Para el desarrollo de este análisis se reagruparán las líneas de productos de acuerdo con su similitud, para que no sea extensa la dimensionalidad de las características que tiene el dataset al momento de la modelización. La agrupación de las líneas de productos es la siguiente. Ver tabla 8 e Ilustración 32.

Tabla 8 Tabla de grupos de rangos

Tabla Rango Línea de Producto	
Línea de producto	***Grupo de Líneas***
VIDEO	*TECNO*
AUDIO	
COMPUTACIÓN	
TECNOLOGÍA	
TELECOMUNICACIÓN	
BLANCA	*BLANCA*
CLIMATIZACION	*CLIMA*
SMALL APPLIANCE	*SMALL APPLIANCE*
DEKO ZONE	*OTROS*

COSTURA
VESTUARIO
DEPORTE
BELLEZA
CONSUMO MASIVO
HERRAMIENTAS Y FERRETERIA
CUIDADO PERSONAL
ESCOLAR
HOME APPLIANCE
JUGUETES
INSTRUMENTOS MUSICALES
MUEBLES
BEBE
RELOJERIA
SALUD
ALIMENTOS

```
BLANCA             278591
CLIMA              159605
OTROS              503597
SMALL APPLIANCE    189055
TECNO              496499
Name: rango_linea, dtype: int64
```

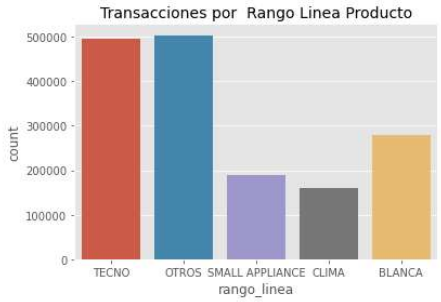


Ilustración 32 Agrupación de Línea de Productos

Para el análisis temporal se realizó un gráfico de líneas, ya que el objetivo es observar el volumen de ventas a lo largo del tiempo. Se visualiza las ventas por meses para cada año (2015 al 2020) en donde se observa estacionalidad en los últimos 6 meses, la venta se incrementa en los meses, mayo y noviembre, esto debido a las campañas de Madres y del BlackFriday respectivamente. Los meses más bajos en venta son enero, febrero y julio. Ver la Ilustración 33.

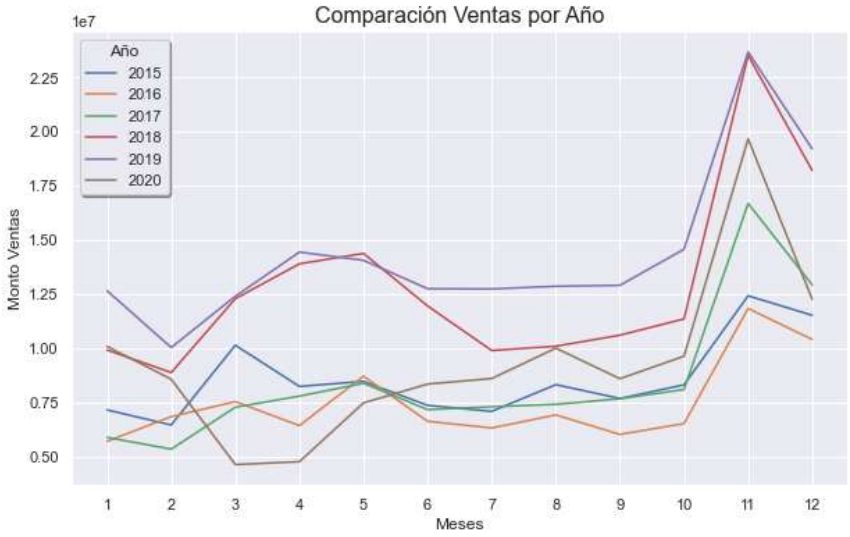


Ilustración 33 Comportamiento de la temporalidad de las transacciones

Para robustecer nuestro análisis exploratorio de los datos se realizó el análisis de RFM, que mide tres dimensiones: Recencia (tiempo que compro por ultima ves), Frecuencia (número de veces que ha comprado) y Monto (cantidad monetaria de venta). Para ello se calcularon las nuevas características de frecuencia y recencia (monto ya lo tenemos) Ver Ilustración 34. Luego se ha agrupado por cliente para obtener el respectivo score RFM, como resultado se obtuvieron 3 grupos: clientes de valor alto, clientes de valor medio y clientes de valor bajo. Ver Ilustración 345.

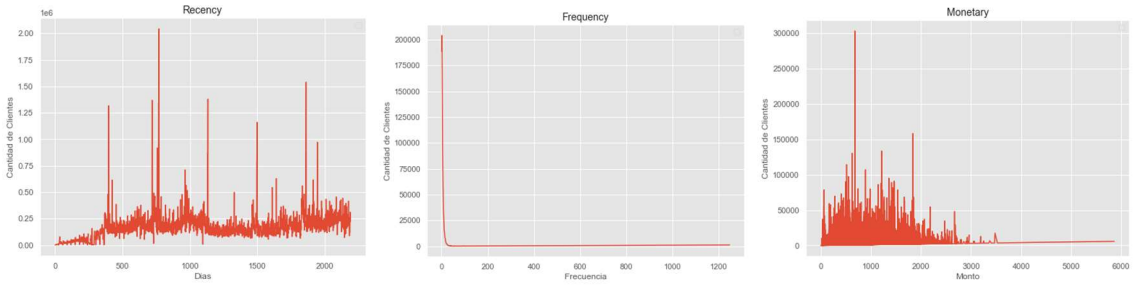


Ilustración 34 Gráficas de comportamiento RFM.

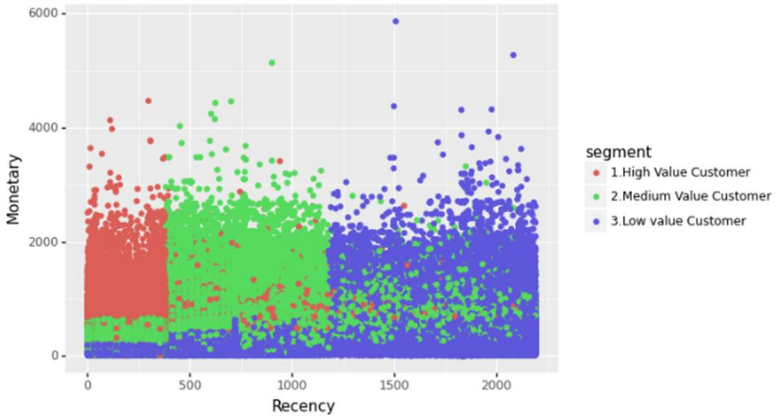


Ilustración 35 Gráfica de dispersión de los segmentos de RFM.

En el gráfico de dispersión se observa que los segmentos de RFM dependen de la recencia. Los clientes “altos” registran menos recencia, menor a 500 días; mientras los clientes “medios” tienen entre 500 y un poco más de 1000 días y los clientes “bajos” son los más antiguos y que compran de menor valor. Para cuantificar estos valores se calculó las medidas descriptivas de los segmentos. Ver Ilustración 36.

	Recency			Frequency			Monetary		
	mean	median	count	mean	median	count	mean	median	count
segment									
1.High Value Customer	415.220301	376	114997	4.699914	4	114997	619.137887	575.436667	114997
2.Medium Value Customer	766.744256	720	152891	2.442950	2	152891	582.225108	503.960000	152891
3.Low value Customer	1155.705870	1082	173403	1.377652	1	173403	348.808754	252.740000	173403

Ilustración 36 Estadística descriptiva de los segmentos del modelo RFM.

Los clientes del grupo “alto” son los más recientes, 415,2 días en promedio, son los que más veces compran, 4 veces en promedio y son los que tienen mayor venta, $619,1 en promedio. Mientras que los clientes del grupo “bajo” registran mayor recencia, 1155,7 días en promedio, menor frecuencia, 1.3 veces y tiene la menor venta, $348.8 en promedio.

La etiqueta creada en el análisis de RFM, segmentación, se utilizará como una característica de entrada más para el modelo de caracterización de los clientes.

Una vez analizado el comportamiento de ventas de los clientes a nivel demográfico, por productos, por el tiempo, y por su segmentación, se tienen algunas conclusiones como, por ejemplo: los clientes masculinos son los que más compran, se registran más solteros y jóvenes que casados y adultos, la mayoría de los clientes (65%) llevan un solo productos, la línea de producto que más se vende es “blanca” y los mejores meses de venta son mayo y noviembre.

Luego del exploratorio se seleccionan las características que formarán parte del dataset final que será entrenado en la fase de modelización. Estas características del dataset han crecido, ahora se registran 41 vs inicialmente 23, se evidencia un problema de dimensionalidad.

Selección de las características

El objetivo de esta sección es seleccionar las características más relevantes para el modelo de clasificación reduciendo la dimensionalidad del problema del dataset para lo cual se seleccionarán un subconjunto de las características originales, se analizarán la correlación para las características numéricas y la variabilidad para las características categóricas, y así obtener una mejor eficacia de entrenamiento en el modelo.

El método que se usará para la selección de las características es el análisis de correlación con coeficiente de Pearson en la cual se identificarán características altamente correlacionadas. Ver Ilustración 37.

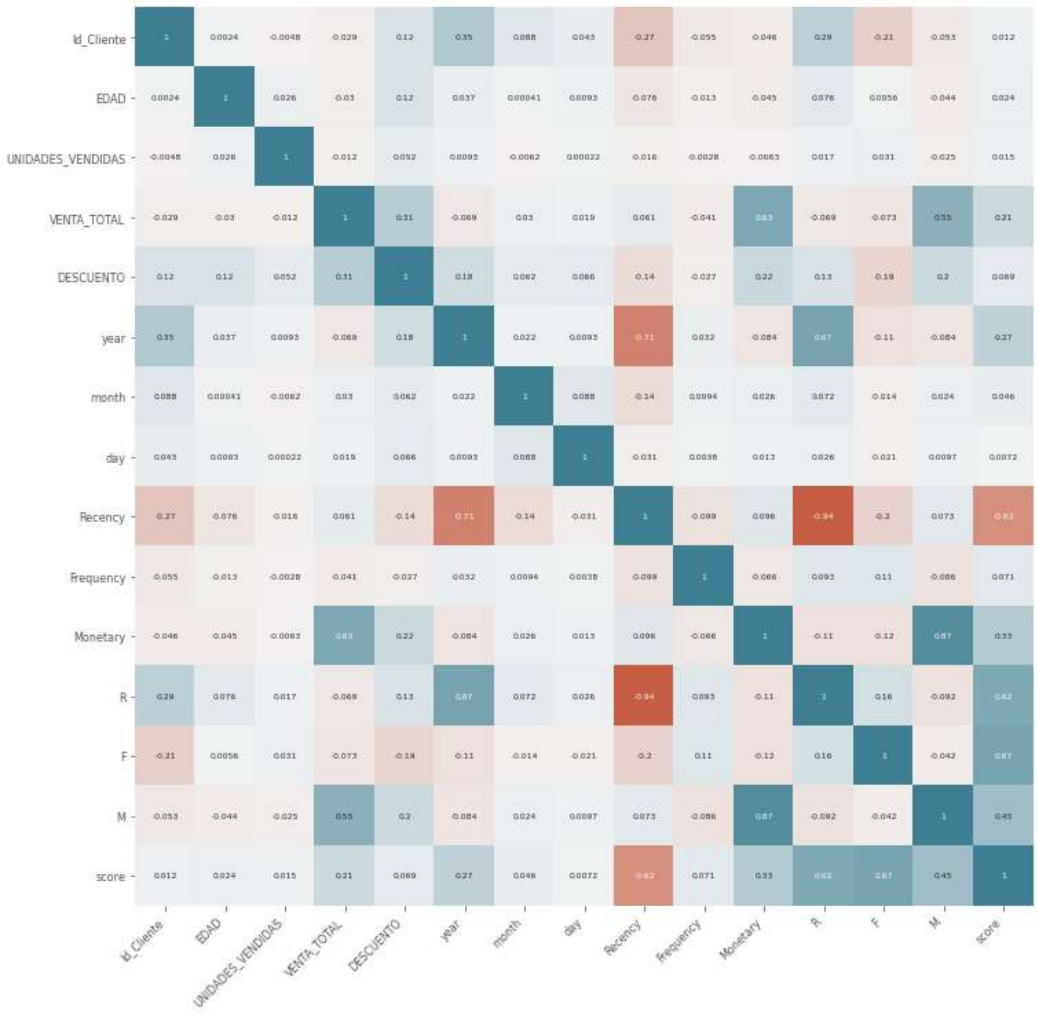


Ilustración 37 Correlación del dataset.

Para la matriz de correlación de los datos se graficó el heatmap para identificar por medio de los colores fuertes las correlaciones positivas y negativas entre las características numéricas. Se observa en la diagonal principal de la matriz, las correlaciones altamente positivas (1) que corresponden a la correlación entre las mismas características. Se visualizan altas correlaciones entre características redundantes que fueron creadas para el análisis exploratorio, por ejemplo: Monetary y M correlación 0,87; Venta total y Monetary correlación 0,63; year y Recency correlación 0,71; R y Recency correlación 0,94; Score y Recency correlación 0,62. Todas estas altas correlaciones identificadas y las características redundantes que se visualizan (las de RFM y tiempo, year y month) se excluirán para el dataset del modelo y así evitar el ruido en la interpretación de los resultados. Adicional las características categóricas que no registran alta variabilidad no serán parte del dataset del modelo, por ejemplo:

género (la mayoría son masculinos), canal (el 95% son por tiendas), estado civil (solteros registra mayor participación).

Finalmente, las características seleccionadas para el dataset del modelo son: forma de pago, rango de plazo, regalo, rango de compra, rango de línea de producto y rango % de descuento, como se muestra en la Ilustración 38.

```
datos
```

	FORMA_PAGO	plazo	rango_compra	regalo	linea_BLANCA	linea_CLIMA	linea_OTROS	linea_SMALL APPLIANCE	linea_TECNO	%_descuento
0	CREDITO	3.6m-18m	3.400-1100	0	0	0	0	0	1	2.1-30
1	CREDITO	3.6m-18m	3.400-1100	0	1	0	0	0	0	2.1-30
2	CREDITO	3.6m-18m	2.100-400	0	1	0	0	0	0	2.1-30
3	CONTADO	1.0m	3.400-1100	0	1	0	0	0	0	3.30-50
4	CREDITO	3.6m-18m	3.400-1100	0	1	0	0	0	0	2.1-30
...	...	...	...	...	...	...	...	...	...	...
1066677	CREDITO	4.>=18m	3.400-1100	0	1	0	0	0	0	2.1-30
1066678	CREDITO	3.6m-18m	3.400-1100	0	1	0	0	0	0	3.30-50
1066679	CREDITO	4.>=18m	3.400-1100	0	0	0	0	0	1	2.1-30
1066680	CREDITO	4.>=18m	2.100-400	0	1	0	0	0	0	2.1-30
1066681	CREDITO	3.6m-18m	3.400-1100	0	1	0	0	0	0	3.30-50

Ilustración 38 Dataset final para la modelización

3.3 Fase de creación del modelo

Esta fase es la creación del modelo de caracterización en donde, se diseñarán, entrenarán, validarán y calibrarán algoritmos no supervisados como, análisis de clústers, para encontrar grupos naturales con características similares de compras para tener una idea más clara al momento de clasificar a los clientes. Posteriormente, se entrenarán una serie de algoritmos supervisados de machine learning para clasificar a los clientes de acuerdo con los grupos identificados en el clúster, como; naive bayes, k-neighbors, SVM con kernel linear, SVM con kernel gaussiano, arboles de decisión y redes neuronales. Estos algoritmos se entrenarán, validaran y luego se determinará el mejor modelo para los datos. Este proceso será desarrollado con la herramienta de Python con la librería *Scikit-learn*, la cual facilita las etapas del preprocesado, entrenamiento, optimización y validación de los modelos de clasificación con sus principales algoritmos y funciones.

El dataset final registra características categóricas, por lo tanto, no podemos usar directamente el algoritmo de aprendizaje no supervisado, K-Means. El algoritmo

mide distancias numéricas y para esto se debe realizar una transformación de los datos categóricos utilizando el método One-Hot-Encoding, el cual consiste en convertir las etiquetas de las características categóricas en nuevas características numéricas de tipo dummies (o y 1). La ventaja de utilizar esta metodología es que las características numéricas proporcionan mejores grupos en los clústeres, la desventaja es que se pierde la relación de datos categóricos. Para el desarrollo de este método se utilizó la función *pd.get_dummies() de la librería OneHotEncoder* desde *sklearn.preprocessing.* Ver Ilustración 379.

```
# One-hot-encoding del dataset
# ------------------------------------------------------------------------------
from sklearn.preprocessing import OneHotEncoder
datos= pd.get_dummies(datos,columns =["FORMA_PAGO","plazo","rango_compra","regalo",'%_descuento'])
datos.head()
```

	linea_BLANCA	linea_CLIMA	linea_OTROS	linea_SMALL APPLIANCE	linea_TECNO	FORMA_PAGO_CONTADO	FORMA_PAGO_CREDITO	plazo_1. 0m	plazo_2. 1m-6m	plazo_3. 6m-12m
0	0	0	0	0	1	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0
2	1	0	0	0	0	0	1	0	0	0
3	1	0	0	0	0	1	0	1	0	0
4	1	0	0	0	0	0	1	0	0	0

5 rows × 22 columns

Ilustración 39 Método OneHotEncoder al dataset.

Se observa que ahora todas las características del dataset son de tipo dummies, es decir, 1 si cumple con la característica y 0 si no. El dataset final ahora tiene 22 características dummies.

Algoritmos no supervisados

Se procede a realizar el algoritmo de clustering, con el objetivo de descubrir clústeres naturales en los patrones de los datos y que nos permita visualizar grupos con “altas” características similares de compra.

Análisis de Clustering

Uno de los problemas que se tiene a la hora de realizar el algoritmo no supervisado, es determinar el número de clúster, elegir este número a criterio propio puede causar agrupaciones heterogéneas (pocos clústeres), o grupos muy similares entres clústeres diferentes (muchos clústeres). Para definir este criterio de selección se utilizaron los métodos dendrogramas y el elbow (método del codo).

Método de Dendrograma

Este método tiene una forma jerárquica en donde se visualiza por medio de los colores, el número de clústers óptimo para el algoritmo no supervisado. Se puede observar en la Ilustración 40 que se visualizan 4 conglomerados, lo cual ocurre a un nivel de similitud de aproximadamente 600 de distancia. El primer conglomerado está conformado por las características de línea blanca, climatización y small appliance, 0% de descuentos, rangos de compra entre $100 y $400, y plazos entre 6 a 18meses. El segundo conglomerado lo conforman formas de pago crédito y % descuentos 30% y mayor a 50%. El tercer conglomerado lo forman: formas de pago contado, plazos 0 meses, descuentos entre 30% y 50% y compras menores a $100 y el ultimo conglomerado lo conforman plazos mayores a 18 meses y otras líneas de producto. Con estos indicios tenemos un análisis a priori de cómo serán los clústeres creados.

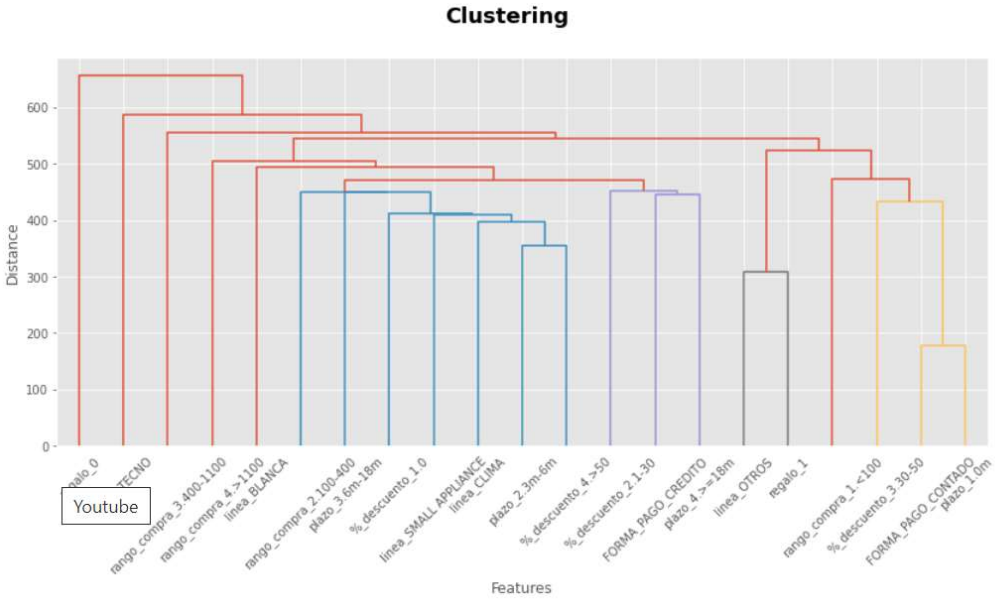


Ilustración 40 Método del dendograma.

Método de Elbow

Este método también busca el valor de k-grupos óptimo de acuerdo con la mayor distancia media entre los clústeres, es decir, maximiza las distancias entre cada grupo. En la hoja de trabajo de jupyter notebook se crea un arreglo con valores del 1 al 10 para lo cual se calculará el score de k-Means para cada valor y se determinará el

punto en donde se rompe la inercia. De acuerdo con la gráfica, el número de grupos sugerido es 4 clústers. Ver Ilustración 41.

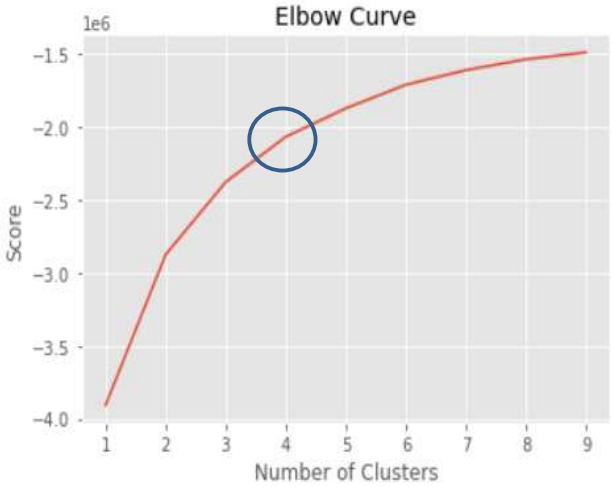


Ilustración 41 Método del Codo.

Para el desarrollo de los algoritmos no supervisados se realizarán dos métodos: Gaussian, y kmeans.

Método Gaussian

En Python usamos la función *GaussianMixture* de *Scikit-Learn* con parámetros k=4, sugerido por el método del codo, tipo de matriz de covarianza *full,* es decir, cada componente tiene su propia matriz de covarianza general, y una semilla para garantizar la reproducibilidad de los resultados. Definimos las predicciones con la función *clusterer.predict* y los valores de los centers con la función *clusterer.means_.* Posteriormente evaluamos estadísticamente (mean) todos los segmentos creados.

	Segment 0	Segment 1	Segment 2	Segment 3
linea_BLANCA	2.132800	20.752010	27.754029	46.364977
linea_CLIMA	0.006940	12.975120	27.524071	11.502970
linea_OTROS	99.981348	9.119983	8.126593	75.976023
linea_SMALL APPLIANCE	5.740386	12.119897	15.081891	31.828523
linea_TECNO	2.878000	59.124267	34.429739	47.511027
FORMA_PAGO_CONTADO	99.973107	0.000000	100.000000	0.181402
FORMA_PAGO_CREDITO	0.026893	100.000000	0.000000	99.818598
plazo_1. 0m	100.000000	0.000000	89.719419	0.000000
plazo_2. 1m-6m	0.000000	3.956028	2.022301	1.100548
plazo_3. 6m-12m	0.000000	8.269182	0.181650	3.799135
plazo_4. 12m-18m	0.000000	29.969567	8.076630	22.261523
plazo_5. >=18m	0.000000	57.805223	0.000000	72.838795
rango_compra_1.<100	89.889478	5.502008	20.896999	0.859083
rango_compra_2.100-400	7.965143	23.753583	25.868709	8.409423
rango_compra_3.400-1100	0.010410	43.819812	26.974820	38.769740
rango_compra_4.>1100	2.134968	26.924597	26.259472	51.961754
regalo_0	4.969160	100.000000	92.624491	0.138327
regalo_1	95.030840	0.000000	7.375509	99.861673
%_descuento_1.0	0.014748	33.028910	4.549846	8.846852
%_descuento_2.1-30	1.014566	56.546160	1.240115	74.947672
%_descuento_3.30-50	96.843092	9.820138	60.129041	14.868923
%_descuento_4.>50	2.127594	0.604792	34.080998	1.336553

Ilustración 42 Evaluación de las características para cada segmento con el método Gaussian

En la Ilustración 42 se pueden observar las características más relevantes (mayor valor) para cada segmento, por ejemplo: en el segmento 0 se tienen los clientes que compran otras líneas de productos al contado con plazos 0 meses y monto promedio menor a $100, la mayoría compran con regalos y descuentos promedios entre el 30% y 50%. En el segmento 2 se encuentran los clientes que compran más en tecnología en contado a plazos 0 meses, no incluyen regalos y la mayoría tienen descuentos promedios entre el 30% y 50%. A continuación, se visualiza la agrupación de los clústeres con el método gaussiano vs la data original en dos dimensiones. Ver Ilustración 42

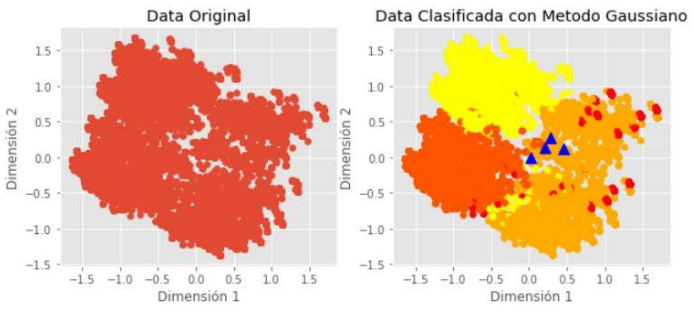


Ilustración 43 Data clasificada método Gaussiano

Método K-Means

Para el desarrollo del método k-Means, se crea el objeto usando la función *Kmeans() con parámetro k=4, y luego los valores predictores con la función kmeans.predict()* y los valores de centroides con la función *kmeans.cluster_centers_.* Posteriormente se grafica el heatmap de los segmentos encontrados para evaluar estadísticamente (métrica mean) las características más relevantes de cada segmento.

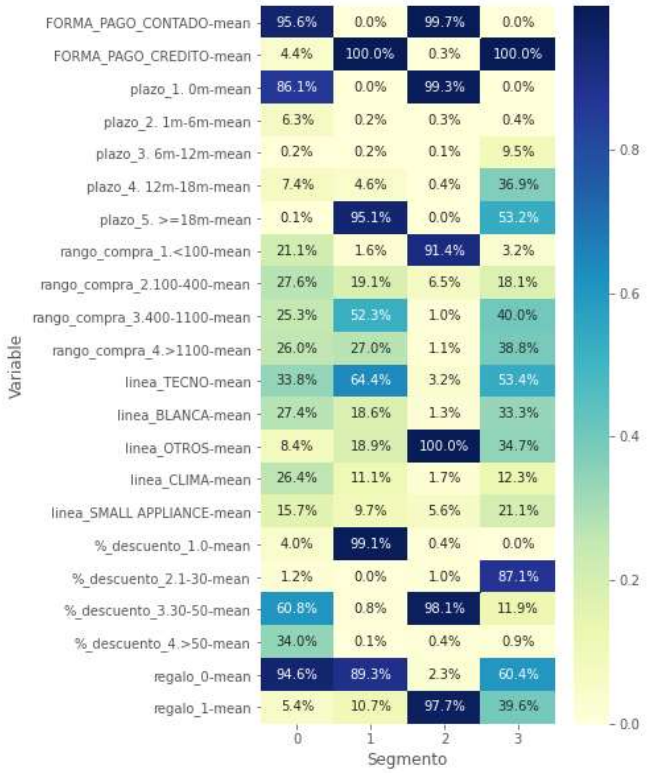


Ilustración 44 Evaluación de características para cada segmento con el método KMEANS

En la Ilustración 44 se observan las características más relevantes que tienen cada segmento creado, por ejemplo: el segmento 1 están conformados por los clientes que compran al crédito con plazos promedios mayor a 18 meses, montos de compra entre $400 y $1100 sin descuentos ni regalos/combos, los productos más frecuentes son de línea de tecnología. El segmento 3 son los clientes que compran en forma de crédito con plazos mayor a 18 meses y con descuentos promedios superior a 30%, por lo general llevan productos de tecnología sin regalos. A continuación, se muestra la agrupación de los clústeres en dos dimensiones, el método kmeans vs la data original. Ver Ilustración 44.

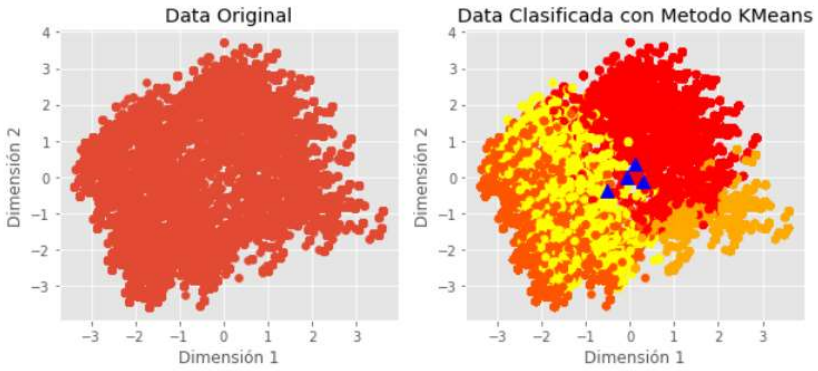


Ilustración 45 Data clasificada método K-means

Finalmente, se cuantifican los 4 segmentos creados por el método del clúster kmean como se muestra en la Ilustración 46, se observa que los segmentos 0 y 1 tienen mayor participación que los segmentos 2 y 3. El grafico muestra un desbalance no significativo en los segmentos.

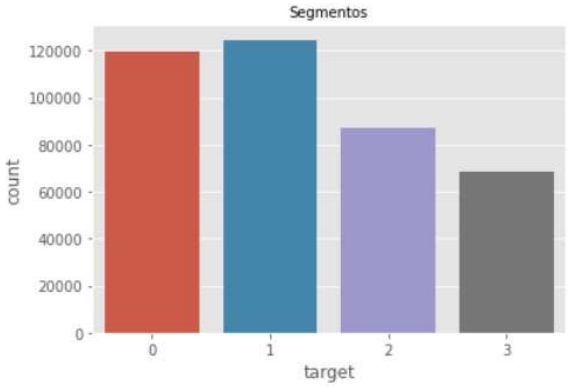


Ilustración 46 Distribución de los segmentos

Data Training y test

Para la evaluación adecuada del rendimiento de cada algoritmo supervisado, se creará un dataset de entrenamiento y otro dataset de prueba, en la cual permitirá entrenar y probar de forma efectiva los modelos, es decir, se seleccionará un subconjunto de los datos originales, llamado data de prueba, en la cual validará y comprobará si el modelo que hemos realizado a partir de los datos de entrenamiento funciona para los datos nuevos. Debido al tamaño de registros del dataset final

(1,059.376 observaciones), se generará una muestra aleatoria para la data entrenamiento y prueba.

Para el cálculo de las muestras se utiliza la función *resample* de la librería *sklearn.utils.* El tamaño de la data de muestra es de 20.000 registros sin la opción de reemplazo. El porcentaje de la muestra con respecto al tamaño de la data original es del 4%. Usamos la función *train_test_split* de *sklearn.model_selection* para dividir nuestro conjunto de datos, se particionó la muestra del dataset con un; 70% a la data de entrenamiento y un 30% a la data de prueba.

```
X = datosmodelo.drop(columns = "target").values

y = datosmodelo['target'].values

x_train, x_eval, y_train, y_eval =
train_test_split(X,y,test_size=0.30,train_size=0.70,random_state=123)
```

Para la evaluación de los resultados y certificar la independencia de la data training y test de cada modelo clasificador, se utilizará la validación cruzada o también llamado cross-validation, que consiste en obtener k-submuestras del dataset, y el resto de submuestras se utilizará como entrenamiento, es decir k-1.

Naive Bayes Gaussiano

Para el desarrollo del primero modelo clasificador se realiza en la hoja de trabajo de Júpiter notebook la función *GaussianNB()* de la librería *sklearn.naive_bayes,* luego se calcula los scores del accuracy para el modelo y los valores para cada segmento.

El resultado del accuracy del modelo usando cross-validation con n=5 es 0.96, para el segmento 0 es 0.97, para el segmento 1 es 0.98, para el segmento 2 es 0,96 y para el segmento 3 es 0,95. Para la data de prueba el accuracy es de 0,96, lo cual no se evidencia sobreajuste en el modelo clasificador.

```
Metrica del modelo 0.9685714285714285
Metricas cross_validation [0.97035714 0.98928571 0.96428571 0.98642857 0.95535714]
Media de cross_validation 0.9731428571428573
Metrica en Test 0.9696666666666667
```

Para realizar el reporte clasificador se importa la función *classification_report* de la librería *sklearn.metrics* en donde se muestra los resultados del accuracy para el modelo general y para cada segmento que se tiene. Sin cross-validation, se observa que el valor de la precisión para el modelo es de 0,97, valor bastante aceptable para la clasificación, ver resultado:

	precision	recall	f1-score	support
0	0.93	0.96	0.95	1676
1	0.99	1.00	1.00	1074
2	0.95	0.92	0.93	1360
3	1.00	1.00	1.00	1890
accuracy			0.97	6000
macro avg	0.97	0.97	0.97	6000
weighted avg	0.97	0.97	0.97	6000

K-Nearest Neighbords

El algoritmo supervisado KNN, es el segundo modelo clasificador para lo cual se desarrolla en Python y se utiliza la función *KNeighborsClassifier* de la librería *sklearn.neighbors* con los parámetros *n_neighbors = 4* y *p=2*, lo cual quiere decir que la similitud (*p=2*) entre los puntos se basará con la distancia euclideana para los 4 segmentos.

Los resultados del accuracy del modelo y de cada segmento usando cross-validation con n=5 son: 0.998, para el segmento 0 es 0.998, para el segmento 1 es 0.998, para el segmento 2 es 0,997 y para el segmento 3 es 0,998. Para la data de prueba el accuracy es de 0,998, lo cual no se evidencia sobreajuste en el modelo clasificador.

```
Metrica del modelo 0.9989285714285714
Metricas cross_validation [0.99892857 0.99892857 0.9975     0.99821429 0.99928571]
Media de cross_validation 0.9985714285714286
Metrica en Test 0.9985
```

En el reporte clasificador sin cross-validation se muestra el accuracy del modelo y para cada segmento, en donde se muestra que para todos los casos el valor es 1.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1676
1	1.00	1.00	1.00	1074
2	1.00	1.00	1.00	1360
3	1.00	1.00	1.00	1890
accuracy			1.00	6000
macro avg	1.00	1.00	1.00	6000
weighted avg	1.00	1.00	1.00	6000

Support Vector Machine (kernel Linear)

El tercer modelo clasificador es el algoritmo SVM. Debido a la representación dimensional en el espacio de las 22 características y que limita las condiciones computacionales de las maquinas, se usará las funciones de Kernel que solucionará la capacidad de las máquinas de aprendizaje y encontrará un hiperplano para problemas no lineales.

En Python se utiliza la función *svm()* de la librería *sklearn* con el parámetro *kernel='linear'*. El resultado del accuracy del modelo usando cross-validation con n=5 es 0.999, para el segmento 0 es 0.997, para el segmento 1 es 0.999, para el segmento 2 es 0,9987 y para el segmento 3 es 0,999. Para la data de prueba el accuracy es 0,999, lo cual no se evidencia sobreajuste en el modelo clasificador.

Este modelo arroja como métricas lo siguiente:

```
Metrica del modelo 0.9993571428571428
Metricas cross_validation [0.99785714 0.99928571 0.99857143 0.99964286 0.99928571]
Media de cross_validation 0.9989285714285716
Metrica en Test 0.9993333333333333
```

En el reporte clasificador sin cross-validation se muestra el accuracy del modelo y los valores para cada segmento, se observa el valor de 1 para todos.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1676
1	1.00	1.00	1.00	1074
2	1.00	1.00	1.00	1360
3	1.00	1.00	1.00	1890
accuracy			1.00	6000
macro avg	1.00	1.00	1.00	6000
weighted avg	1.00	1.00	1.00	6000

Support Vector Machine (kernel RBF)

Este modelo de support vector machine es el cuarto modelo clasificador en donde se utilizará la misma función SVM que el modelo anterior con la diferencia del parámetro en el *kernel RBF*, que está basado en la función base radial,

En python utilizaremos la función *svm* de la librería *sklearn* con el parámetro *kernel='rbf' y gamma=0.1,* grado del kernel. El resultado del accuracy del modelo usando cross-validation con n=5 es 0.998, para el segmento 0 es 0.996, para el segmento 1 es 0.997, para el segmento 2 es 0,999 y para el segmento 3 es 0,999. Para la data de prueba el accuracy es 0,998, lo cual no se evidencia sobreajuste en el modelo clasificador.

```
Metrica del modelo 0.9985
Metricas cross_validation [0.99678571 0.9975     0.99714286 0.99964286 0.9975    ]
Media de cross_validation 0.9977142857142857
Metrica en Test 0.9981666666666666
```

El resultado del reporte clasificador sin usar el método de cross-validation se muestra el accuracy del modelo y los valores para cada segmento, se observa el valor de 1 para todos los casos.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1676
1	0.99	1.00	1.00	1074
2	1.00	1.00	1.00	1360
3	1.00	1.00	1.00	1890
accuracy			1.00	6000
macro avg	1.00	1.00	1.00	6000
weighted avg	1.00	1.00	1.00	6000

Arboles de Decisión

El quinto modelo clasificador es el de árboles de decisiones para lo cual en la hoja de trabajo de jupyter se utiliza la función *DecisionTreeClassifier* de la librería *sklearn.tree*, con el parámetro *criterion='entropy'*.

Dentro del desarrollo de este modelo de árboles, es necesario definir la profundidad del árbol, que significa el tamaño del árbol y ayuda a evitar el sobreajuste en la modelización, es decir, si no lo definimos el árbol podría tener el 100% de precisión en la data de entrenamiento (todas las hojas posibles), como también podría tener una hoja por cada observación. En python para obtener el valor máximo del tamaño del árbol se utiliza la función *max_depth.* Se crea una función para determinar cuál es el valor óptimo de la profundidad del árbol.

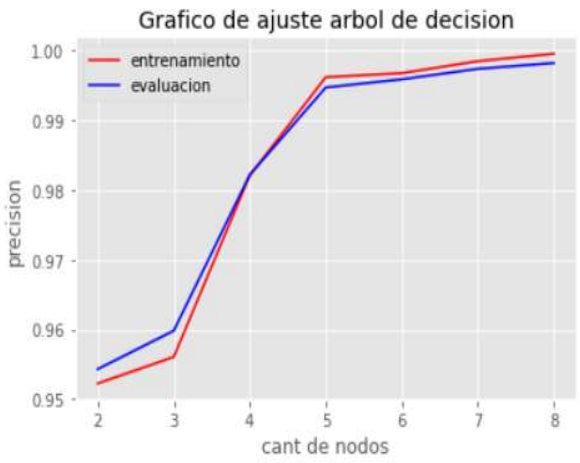


Ilustración 47 Grafica de Ajuste del modelo

Como se puede observar en la Ilustración 47, el numero donde se obtendrá la mayor precisión de clasificación es 5, es decir, la profundidad de nuestro árbol de decisión será max_depth = 5. Con esta definición del parámetro se procede a realizar el modelo de clasificación, dando como resultados los accuracy con cross-validation para el modelo de prueba de 0,998 y para el segmento 0 de 0,998, en el primer segmento de 0,998, en el segundo segmento de 0,999 y en el tercer segmento de 0,998, donde no se evidencia sobreajuste en el modelo.

```
Metricas cross_validation [0.99821429 0.99892857 0.99892857 0.99964286 0.99892857]
Media de cross_validation 0.9989285714285714
Metrica en Test 0.9988333333333334
```

Se grafica el árbol utilizando la función *plot_tree()* con una profundidad de 5 y 23 nodos terminales. Ver Ilustración 48.

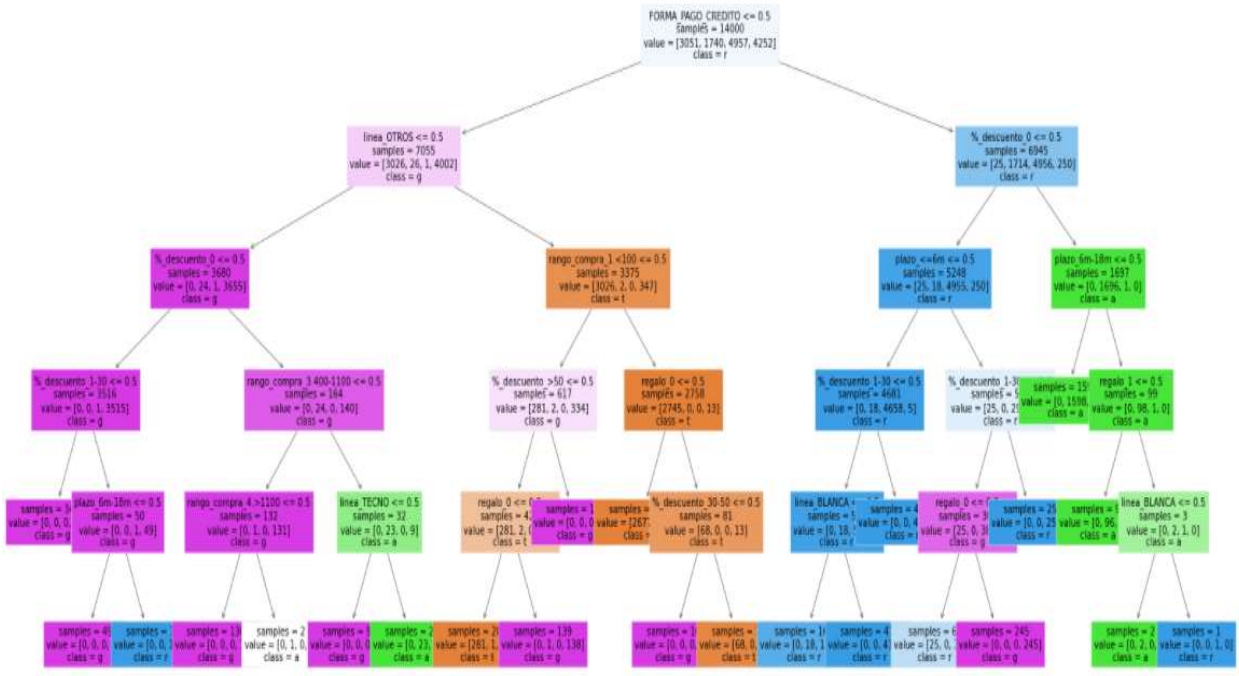


Ilustración 48 Grafica del árbol de decisiones

Dentro del árbol podemos visualizar por los colores las clases clasificadas, color purpura clase g, color verde clase a, color naranja clase t, y color celeste clase r. Los nombres de las clases (g, a, t y r) son definimos por el algoritmo del árbol. También se observa que el árbol clasifica como nodo principal a la característica forma de pago. Estas ramificaciones ayudan a entender que características conforman cada clase, para así conocer el perfil de cada cliente que pertenezca a cierta clase, por ejemplo:

- ✓ clase 0, color naranja. - los clientes están caracterizados por su forma de pago contado, línea de productos “otros”, rango de compras menor a $100, con de descuentos entre el 30% y 50%, con regalos/combos, es decir, **clientes promocionales**.
- ✓ clase 1, color purpura. - los clientes se caracterizan por la forma de pago crédito, sin descuentos ni regalos/combos, línea de productos tecnologías, rango de

compras mayor a $400 en plazos superior a 18 meses, es decir, **clientes tecnológicos**.

- ✓ clase 2, color celeste. – clientes clasificados por forma de pago crédito, plazo mayor a 6 meses, con descuentos del 30%, línea de productos Blanca y tecnologías y rangos de compras mayor $400, es decir, **clientes crediticios**
- ✓ clase 3, color verde. – clientes caracterizados por clientes a contado, con descuentos superior al 30% sin regalos/combos, todas las líneas de productos, es decir, **clientes con descuentos**

Se observa en el reporte clasificador sin cross-validation que los valores del accuracy para este modelo es del 100% tanto para el train y test como para cada segmento:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1676
1	1.00	1.00	1.00	1074
2	1.00	1.00	1.00	1360
3	1.00	1.00	1.00	1890
accuracy			1.00	6000
macro avg	1.00	1.00	1.00	6000
weighted avg	1.00	1.00	1.00	6000

La función *cost_complexity_pruning_path* de *sklearn* nos da los valores Alpha efectivos y las impurezas de los subárboles durante la poda, estos valores son usados para podar nuestro árbol y evitar un sobreajuste en la modelización. El nivel de impureza y efectividad de la data de entrenamiento se la puede visualizar en la Ilustración 49.

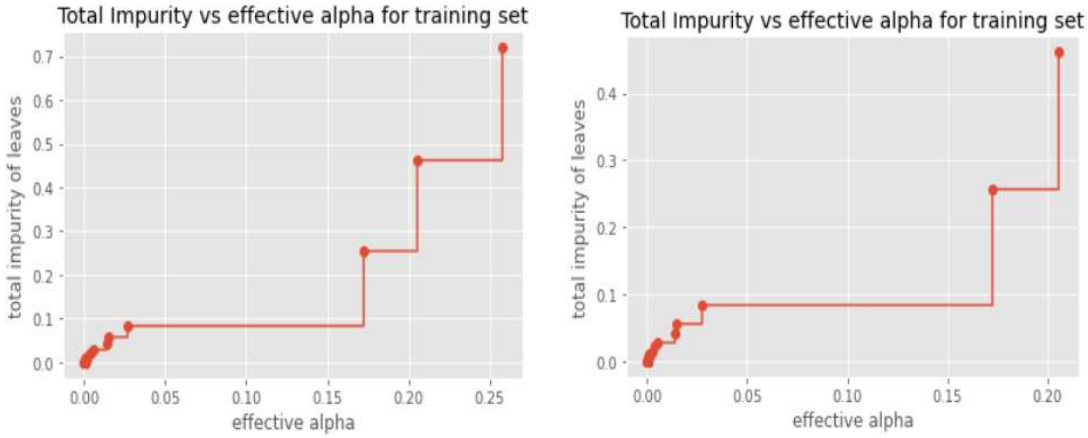


Ilustración 49 Grafica de Cost-Complexity Pruning

Redes Neuronales

El último modelo clasificador es para las redes neuronales. En python importaremos la función *MLPClassifier* de la librería *sklearn.neural_network* para realizar el modelo de clasificación de redes. Será necesario optimizar los hiper-parametros de la red, como son: el número de neuronas optimo, el learning rate, y el número de capas. Ver ilustración 50.

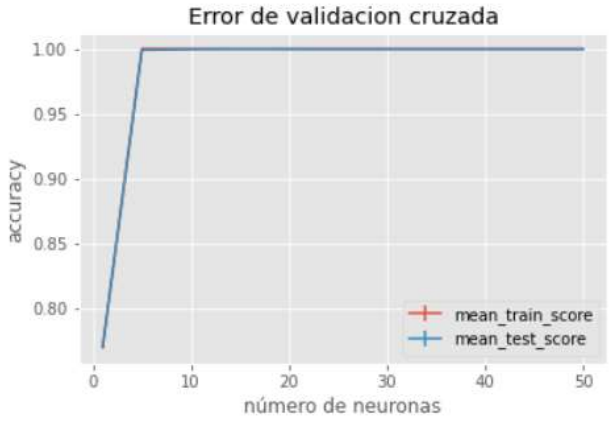


Ilustración 50 Numero optimo de Neuronas

Para determinar el número óptimo de neuronas se utilizará la función *GridSearchCV()* y se graficará un arreglo con valores entre 0 y 50 y se observará en que neurona alcanza el accuracy más alto. Se puede observar que a partir de cinco neuronas el accuracy toma su valor máximo y se estabiliza para las demás neuronas. Se muestra cómo afectan al aprendizaje algunos de los hiper-parametros más influyentes. Como los 5 predictores tienen la misma escala, no es estrictamente necesarios aplicarles una normalización previo entrenamiento.

Para determinar el hiper-parametros learning rate también utilizaremos la función de *GridSearchCV()* en donde visualizaremos en qué punto el accuracy es el más bajo para lo cual calcularemos el logaritmo del learning rate. Se puede observar que en el valor 10 el learning rate obtiene el accuracy menor. Ver Ilustración 4951.

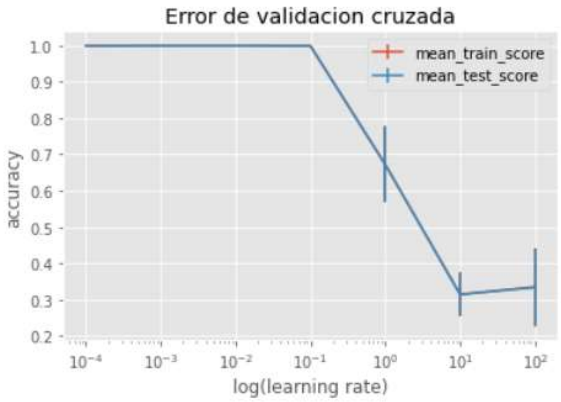


Ilustración 51 Learning rate de redes neuronales

Si bien los dos cálculos anteriores sirven para tener una idea intuitiva de cómo afecta cada hiper-parametros, no es posible optimizarlos de una forma individual, ya que el impacto final que tiene cada uno depende de qué valor tomen los demás. La búsqueda de hiper-parametros debe hacerse en conjunto. Dado el elevado número de hiper-parametros que tiene los modelos de redes neuronales, la combinación de posibles configuraciones es muy elevada. Esto hace que la búsqueda de hiper-parametros por *GridSearchCV()* (todas las combinaciones) sea poco práctica. En su lugar, suele emplearse *RandomizedSearchCV()*, que hace una búsqueda de combinaciones aleatorias.

La combinación de hiper-parametros con los mejores resultados son:

	param_learning_rate_init	param_hidden_layer_sizes	param_alpha	mean_test_score	std_test_score	mean_train_score	std_train_score
37	0.001	(10, 10)	0.001	0.99970	2.121426e-08	1.000000	0.000000
24	0.001	(10, 10)	0.1	0.99970	1.224959e-04	1.000000	0.000000
35	0.1	10	0.1	0.99970	1.224959e-04	1.000000	0.000000
32	0.01	(20, 20)	0.001	0.99965	1.414037e-04	1.000000	0.000000
10	0.1	(20, 20)	0.1	0.99960	1.870635e-04	1.000000	0.000000
18	0.001	(10, 10)	1	0.99960	1.414037e-04	1.000000	0.000000
0	0.001	(20, 20)	0.1	0.99960	1.871056e-04	1.000000	0.000000
28	0.1	10	1	0.99955	1.224500e-04	0.999925	0.000061
11	0.001	10	1	0.99955	2.121108e-04	0.999975	0.000035
19	0.1	(20, 20)	0.001	0.99955	2.121108e-04	1.000000	0.000000

Dando el mejor modelo de redes neuronales con los siguientes parámetros Alpha = 0,001, dos capas con 10 neuronas, número máximo de interacción de 2000:

```
modelo = grid.best_estimator_
modelo
```

```
MLPClassifier(alpha=0.001, hidden_layer_sizes=(10, 10), max_iter=2000,
              solver='lbfgs')
```

Los resultados de las métricas de este modelo con cross-validation son; accuracy del modelo 0,999, accuracy del segmento 0 es 0,999, del segmento 1 es 0,999, del segmento 2 es 1 y el accuracy del segmento 3 es 0,999.

```
Metrica del modelo 1.0
Metricas cross_validation [0.99964286 0.99964286 0.99928571 1.         0.99964286]
Media de cross_validation 0.9996428571428572
Metrica en Test 0.9996666666666667
```

En el resultado del reporte clasificador sin cross-validation para el modelo de redes se observa que el accuracy es del 100% tanto como para el modelo como también para cada uno de los segmentos.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1676
1	1.00	1.00	1.00	1074
2	1.00	1.00	1.00	1360
3	1.00	1.00	1.00	1890
accuracy			1.00	6000
macro avg	1.00	1.00	1.00	6000
weighted avg	1.00	1.00	1.00	6000

3.4 Visualización de Resultados

Es necesario mostrar visualmente los resultados obtenidos en la fase de modelización a los usuarios finales, quienes serán que tomarán la decisión de sus estrategias comerciales para distintas campañas de ventas de acuerdo con los perfiles encontrados en cada grupo de cliente. Para lo cual se utilizará la herramienta visual interactiva y dinámica de inteligencia llamado Microsoft Power BI, que nos brindará gráficos dinámicos e interactivos de cada grupo de clientes, mostrando el perfil sociodemográficos y comportamiento de venta para facilitar y direccionar las campañas comerciales de venta por los distintos medios.

Se muestra un diseño de cómo se presentará el dashboard en la herramienta del BI. Ver Ilustración 44

Ilustración 52 Dashboard para analisis de clusters.

CAPÍTULO 4

4. Evaluación del modelo de caracterización de clientes

Para el aprendizaje automatizado es importante las etapas del análisis de datos y modelización, de la misma forma es vital considerar la evaluación de los modelos que se usan en los entrenamientos. El rendimiento de estos modelos se cuantifica generalizando para un set de datos nuevos y observar su precisión. [6].

Para mejorar la predicción general de un modelo es imperativo realizar varias evaluaciones utilizando diferentes métricas antes de que lo pongamos a producción a ejecutarse sobre los datos nuevos. De no realizar dicha evaluación de forma apropiada puede presentarse el problema de no generalizar sobre los datos no vistos y devolver predicciones erróneas. [6].

Este problema se da básicamente en casos en que los modelos no aprenden, sino que memorizan; por tal motivo generalizar bien sobre datos nuevos. [6].

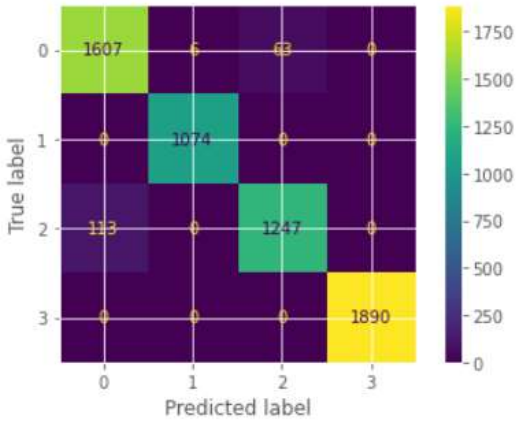
4.1 Análisis de rendimiento del modelo

Una vez construido el modelo se definen las métricas para la evaluación, este es un paso esencial de todo proyecto de ciencia de datos, con el cual se logrará estimar la precisión del modelo de aprendizaje con un set de datos nuevos (no vistos/ fuera de la muestra o futuros). [6].

4.2 Análisis de precisión del modelo

Matriz de confusión

Para la validación del modelo de Naive bayes clasificador se realiza la matriz de confusión para lo cual se importa la función *plot_confusion_matrix* de la librería *sklearn.metrics*, en donde se muestra en la diagonal principal los valores acertados por el modelo y en el resto los valores no acertados. Ver ilustración 53.

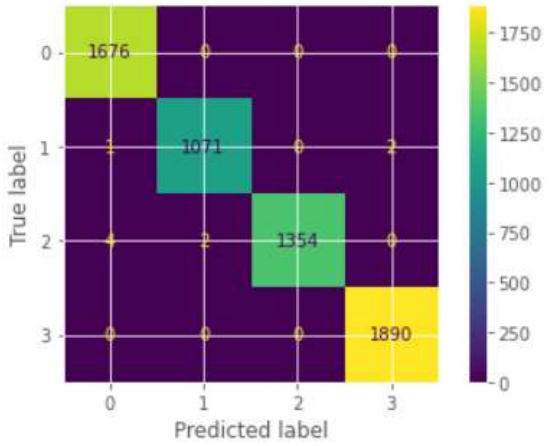


	precision	recall	f1-score	support
0	0.93	0.96	0.95	1676
1	0.99	1.00	1.00	1074
2	0.95	0.92	0.93	1360
3	1.00	1.00	1.00	1890
accuracy			0.97	6000
macro avg	0.97	0.97	0.97	6000
weighted avg	0.97	0.97	0.97	6000

Ilustración 53 Matriz de confusión método Gaussiano

Como se visualiza en la matriz de confusión del algoritmo de clasificación naive bayes tiene un porcentaje de aciertos en la data de prueba del 97% (recall). Se registran pocos valores en los cuadrantes de falso positivos y falso negativos (113 casos se predicen como segmento 0, pero la verdad es segmento 2).

En la matriz de confusión del modelo de KNN clasificador se muestra en la diagonal principal los valores acertados por el modelo y en el resto de la matriz los valores no acertados. Ver ilustración 54.

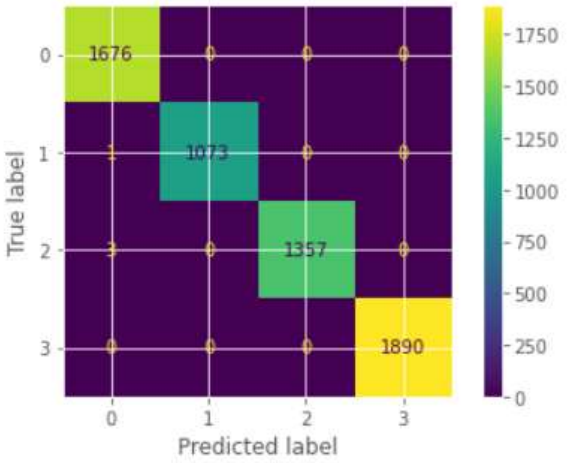


	precision	recall	f1-score	support
0	1.00	1.00	1.00	1676
1	1.00	1.00	1.00	1074
2	1.00	1.00	1.00	1360
3	1.00	1.00	1.00	1890
accuracy			1.00	6000
macro avg	1.00	1.00	1.00	6000
weighted avg	1.00	1.00	1.00	6000

Ilustración 54 Matriz de confusión método K-Nearest Neighbords

Al igual que en el modelo naive bayes, en la matriz de confusión del modelo de KNN se visualiza el recall del 100% de aciertos en la data de prueba, no se evidencia valores falsos positivos ni falsos negativos.

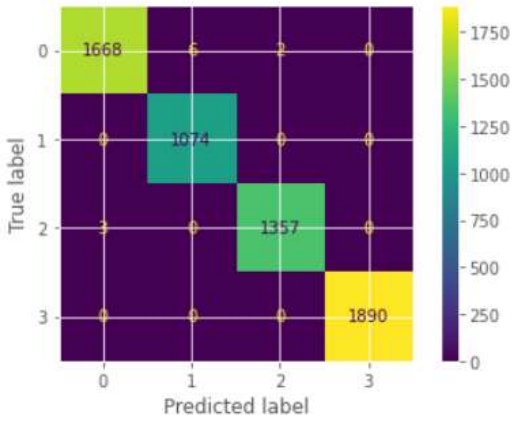
Para validar el modelo SVM Linear clasificador se realiza la matriz de confusión en donde se visualiza los valores acertados en la diagonal principal y en el resto de la matriz los valores equivocados. Ver ilustración 55. Podemos observar el recall que registra en la data de prueba, dando como resultado el 100% de aciertos:



	precision	recall	f1-score	support
0	1.00	1.00	1.00	1676
1	1.00	1.00	1.00	1074
2	1.00	1.00	1.00	1360
3	1.00	1.00	1.00	1890
accuracy			1.00	6000
macro avg	1.00	1.00	1.00	6000
weighted avg	1.00	1.00	1.00	6000

Ilustración 55 Matriz de confusión SVM kernel Linear

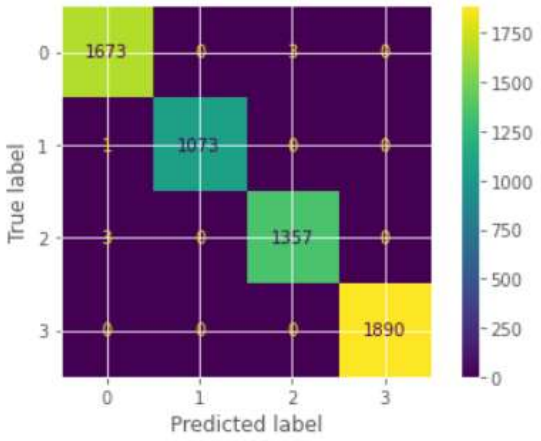
Para el modelo SVM con kernel RBF se realiza la matriz de confusión en donde se observa los valores acertados en la diagonal principal y en el resto de la matriz los valores que no fueron acertados. Se puede visualizar que el porcentaje de acierto, el recall, para los nuevos datos es del 100%. Ver ilustración 56.



	precision	recall	f1-score	support
0	1.00	1.00	1.00	1676
1	0.99	1.00	1.00	1074
2	1.00	1.00	1.00	1360
3	1.00	1.00	1.00	1890
accuracy			1.00	6000
macro avg	1.00	1.00	1.00	6000
weighted avg	1.00	1.00	1.00	6000

Ilustración 56 Matriz de confusión SVM kernel RBF

La validación del modelo de árboles de decisión se muestra en la matriz de confusión, en donde se observa como resultado el valor del recall del 100% de acierto con respecto a la data de prueba. Ver Ilustración 57:



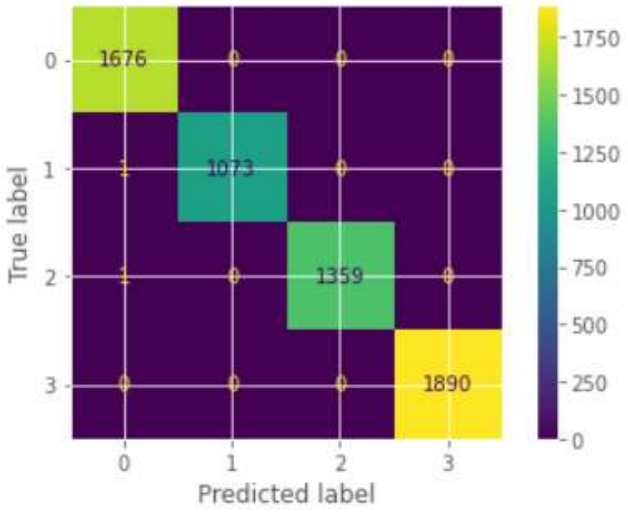
	precision	recall	f1-score	support
0	1.00	1.00	1.00	1676
1	1.00	1.00	1.00	1074
2	1.00	1.00	1.00	1360
3	1.00	1.00	1.00	1890
accuracy			1.00	6000
macro avg	1.00	1.00	1.00	6000
weighted avg	1.00	1.00	1.00	6000

Ilustración 57 Matriz de confusión del modelo de árbol de decisión

El porcentaje de aciertos en la data de prueba es cerca del 100%, los valores de la diagonal principal, segmento 0 igual a 1673, segmento 1 igual a 1073, segmento 2 igual a 1357 y segmento 3 igual a 1890 corresponden con los valores apreciados de forma bien por el modelo, tanto los positivos verdaderos, como los verdaderos negativos. La otra diagonal, por tanto, representa los casos en los que el modelo se ha

equivocado, 3 casos donde la data de entrenamiento predijo como segmento 0 y en realidad eran segmento 2.

Para la validación del modelo de Redes Neuronales clasificador se realiza la matriz de confusión, en donde se observa que da como resultado el valor del recall del 100% de acierto con los datos de prueba. Ver Ilustración 49.



	precision	recall	f1-score	support
0	1.00	1.00	1.00	1676
1	1.00	1.00	1.00	1074
2	1.00	1.00	1.00	1360
3	1.00	1.00	1.00	1890
accuracy			1.00	6000
macro avg	1.00	1.00	1.00	6000
weighted avg	1.00	1.00	1.00	6000

Ilustración 58 Matriz de confusión de Redes Neuronales

El modelo de prueba, en donde obtendremos los nuevos clientes, está clasificando de manera precisa, tal cual como clasifica el modelo de entrenamiento.

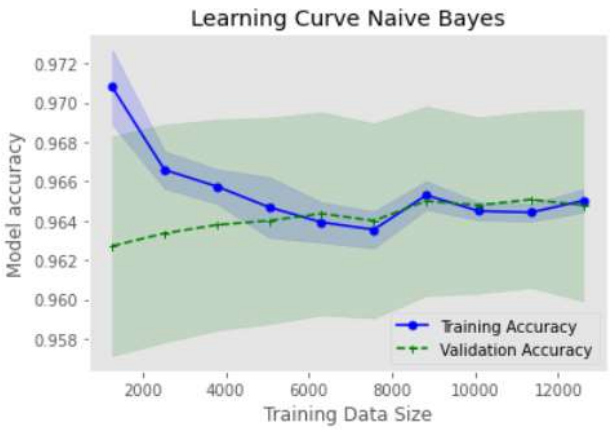
4.3 Análisis de Bias y varianza del modelo

En esta sección se mostrará el error del sobre ajuste de los modelos entrenados. Para calcular estos errores dentro del modelo se observará el bias, que es la diferencia entre la predicción esperada del modelo vs los valores verdaderos, y también se observará la varianza, que es la cantidad de estimación que cambia cuando se tiene

diferentes datos de entrenamiento. La combinación de estas dos métricas mide el rendimiento de la precisión de los modelos.

Para la visualización de esta métrica se graficará el learning curve (curva de aprendizaje) en donde muestra el score de la validación y entrenamiento para diferentes data training. Para el desarrollo se creará en python una canalización lo cual será un estimado para el método de la curva de aprendizaje, se utiliza la función *make_pipeline().*

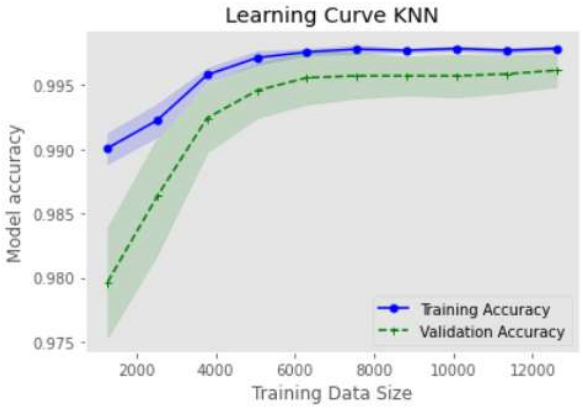
En la curva de aprendizaje de Naive bayes se muestra que a medida que crece la data de entrenamiento el accuracy training se ajusta a la data de validación, el accuracy del modelo esta alrededor de 0.966, en otras palabras, no se evidencia presencia de bias y alta variance. Ver Ilustración 44




```
Average expected loss: 0.013
Average bias: 0.014
Average variance: 0.002
```

Ilustración 59 Bias y Variance Naive Bayes

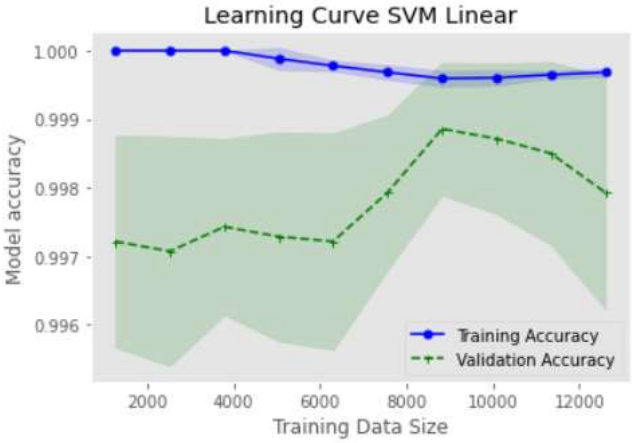
En la Ilustración 60 se observa la curva de aprendizaje en donde se diagnostica el equilibrio del modelo KNN clasificador, en la cual se muestra que a medida que crece el accuracy del training también crece el accuracy del testing, es decir, no se evidencia la presencia de sesgo ni de bias.




```
Average expected loss: 0.004
Average bias: 0.003
Average variance: 0.003
```

Ilustración 60 Bias y Variance KNN

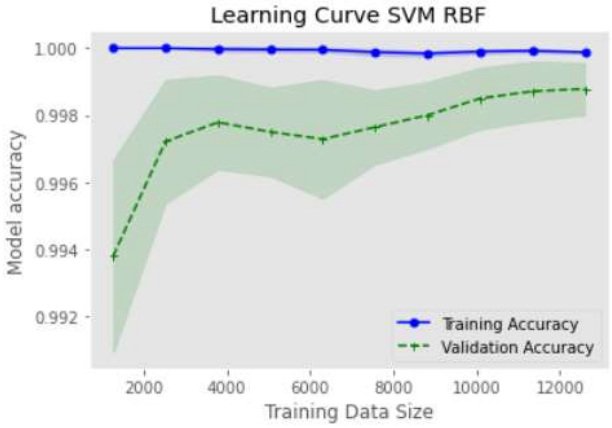
En la curva de aprendizaje del modelo SVM Linear clasificador se observa su complejidad, en donde el accuracy del training se encuentra muy distante al accuracy de la validación en cualquier tamaño de la data, claramente no se observa presencia de varianza ni bias en el modelo. Ver ilustración 61.




```
Average expected loss: 0.001
Average bias: 0.001
Average variance: 0.000
```

Ilustración 61 Bias y Variance SVM linear

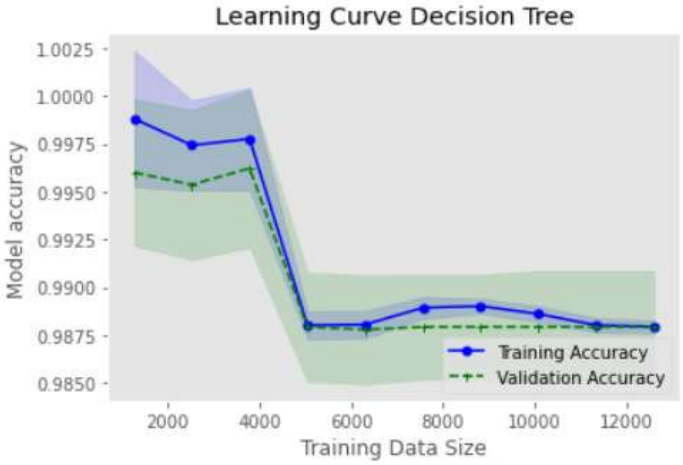
Al igual que el método svm con kernel igual linear, con la función kernel igual rbf también se observa el crecimiento del accuracy del training y del accuracy del testing. No se evidencia la presencia de varianza ni bias en la modelización. Ver ilustración 62.




```
Average expected loss: 0.002
Average bias: 0.002
Average variance: 0.001
```

Ilustración 62 Bias y Variance SVM RBF

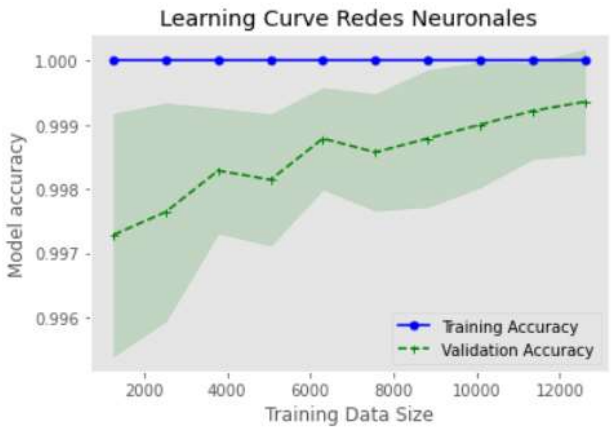
En la curva de aprendizaje del modelo de árboles de decisión podemos observar que el accuracy del training se ajusta al accuracy del testing, a medida que se aumenta el tamaño de la data el accuracy se va ajustando al valor 0.98, el grafico da un resultado más ideal que las anterior, no presenta evidencia de sesgo y varianza. Ver Ilustración 49.




```
Average expected loss: 0.001
Average bias: 0.002
Average variance: 0.001
```

Ilustración 63 Bias y Variance Arbol de decisión

En la curva de aprendizaje del modelo de redes se puede observar que el valor de la precisión del training no se ajusta al valor accuracy del testing, por más precisión que tenga el modelo, no se presenta evidencia de sesgo y varianza. Ver Ilustración 49.




```
Average expected loss: 0.001
Average bias: 0.000
Average variance: 0.000
```

Ilustración 64 Bias y Variance Redes Neuronales

Resumen de las métricas de rendimiento analizadas.

Una vez entrenado los distintos modelos de clasificación, se procede a la selección del mejor modelo, para lo cual se comparará los resultados de acuerdo con sus métricas.

Tabla 9 Tabla de resultados de métricas de los modelos aplicados

Modelos de Clasificación	***Accuracy***		***Rendimiento***		***Time***
	train	*test*	*bias*	*variance*	
Naive Bayes	*0.968*	*0.969*	*0.014*	*0.002*	*0,0160*
KNN	*0.998*	*0.998*	*0.003*	*0.003*	*2,0118*
SVM Linear	*0.999*	*0.999*	*0.001*	*0.000*	*0,2238*
SVM RBF	*0.998*	*0.998*	*0.002*	*0.001*	*1,0062*
Arboles Decisión	***0.999***	***0.998***	***0.002***	***0.001***	***0,4698***
Redes Neuronales	*1.000*	*1.000*	*0.000*	*0.000*	*0,3477*

Como se puede apreciar en la tabla 8, los modelos desarrollados en esta fase muestran una alta precisión, bajo sesgo y baja varianza, es decir, las características seleccionadas son los suficientemente discriminatorias para clasificar a sus clientes en sus correspondientes grupos sin importar el algoritmo de aprendizaje de máquina que se seleccione. Algunos modelos se diferencian por su demanda de tiempo

computacional, ya que ciertos algoritmos generan mayor tiempo que otros en procesarlos. Para nuestro proyecto se ha decidido seleccionar el modelo de árboles decisiones por facilidad de entendimiento en los perfiles de los clientes y en la toma de decisiones por los interesados.

4.4 Análisis de los perfiles de los clientes

Los perfiles de clientes le dan una información útil al área de marketing, para obtener una visión más clara por donde dirigir sus estrategias de ventas. Estos perfiles dan a conocer los comportamientos y tendencias a los que les motiva comprar, similar que sus medios sociodemográficos y redes sociales. Para la toma de decisiones en sus estrategias comerciales esto es una parte importante para considerar.

A la hora de clasificar y diseñar perfiles de clientes, es necesario al menos obtener los datos sociodemográficos de cada persona, como; edad, género, ubicación, entre otros, para obtener de mejor manera una estrategia comercial más directa al cliente, qué al final es el lenguaje que se usará para comunicarse con el cliente por los distintos medios de publicidad, SMS, email, redes sociales, entre otros.

Se obtuvieron 4 segmentos de clientes para los cuales se describe el perfil de cada grupo clasificado.

Segmento 1**. Con descuentos**

Los clientes de este grupo son homogéneos en su comportamiento de compras, sus perfiles son: clientes que compran a contado en su mayoría en efectivos, con descuentos en promedio mayor al 30%, no se diferencian por montos de ventas y por productos que más se repiten son; tecnologías, línea blanca y climatización, por lo general estos productos no vienen con combo, regalo o promoción. Es decir, son clientes en efectivos.

Segmento 2, **Tecnológicos.**

Los clientes que forman este grupo tienen un comportamiento de venta muy similar y su perfil se describe como: clientes que compran a un monto promedio superior a $700 en forma de crédito con un plazo promedio superior a 18 meses, los productos más frecuentes son de tecnología en donde se incluye línea de productos como audio, video, y computación, estos clientes compran estos productos en su

mayoría con un descuento promedio del 30% y no llevan regalos. Estos clientes son denominados tecnológicos.

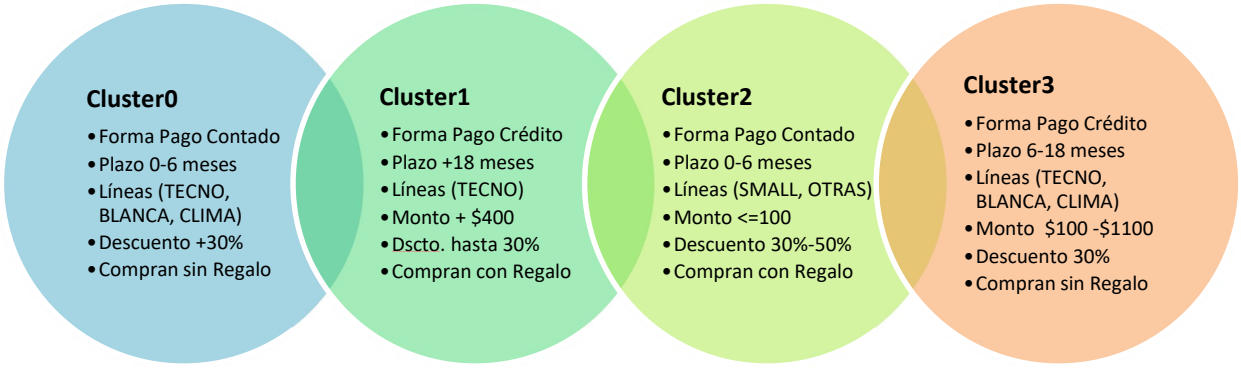
Segmento 3. **Compras Pequeñas**

En este grupo, los clientes comparten un comportamiento de venta muy parejo, los perfiles de este grupo son: clientes que compran a contado en efectivo con un monto promedio menor a $100, los productos que predominan en este grupo son: juguetes, ropa, alimentación, por lo general productos pequeños, con un descuento promedio entre el 30% y 50% adicional se incluyen regalos o combos, es decir, aprovechan las promociones de esta línea de productos para comprar.

Segmento 4. **Crediticio**

El perfil de los clientes que conforman este grupo es: clientes que compran con el crédito directo de la compañía, con un monto promedio entre $400 y $1100 a un plazo promedio entre 12 y 18 meses, descuentos alrededor del 30%, los productos que más compran estos clientes son de la línea Blanca, tecnologías. Ver Ilustración 65.

Segmentos de Clientes



CLIENTES CON DESCUENTOS	*CLIENTES TECNOLOGICOS*	*CLIENTES COMPRAS BAJAS*	*CLIENTES CREDITICIOS*
			

Ilustración 65 Segmentación y Perfiles de Clientes

Reglas de Asociación.

Para el proyecto se desarrollará el algoritmo apriori en el lenguaje de programación python usando la función *association_rules* de la librería *mlxtend.frequent_patterns.* Los resultados para cada segmento son:

Clientes con Descuentos

Los productos con mayor support, es decir, productos con mayor frecuencia dentro de cada transacción que compran los clientes con descuentos, estos son: televisores, ventiladores, aires acondicionados, refrigeradoras, lavadoras entre otros. Se puede observar los 20 productos con mayor support en la Ilustración 66.

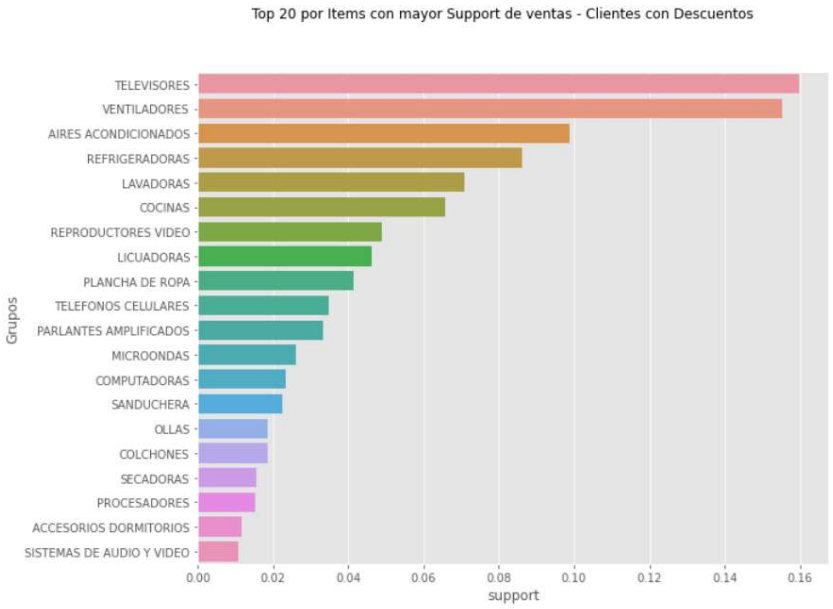


Ilustración 66 Top de líneas de productos del segmento Clientes con Descuentos

Una vez identificado los productos con mayor frecuencia, se generan las reglas de asociación de acuerdo con la confianza, las regla con mayor probabilidad de venta son: si el cliente con descuento compró cocinas, sanduchera y/o ollas es muy probable que en su próxima compra lleve una refrigeradora, si el cliente llevó refrigeradora, plancha de ropa y/o procesadores es muy probable que se lleve una sanduchera, entro otros. Las 20 reglas con mayor probabilidad de venta se visualizan en la Ilustración 67.

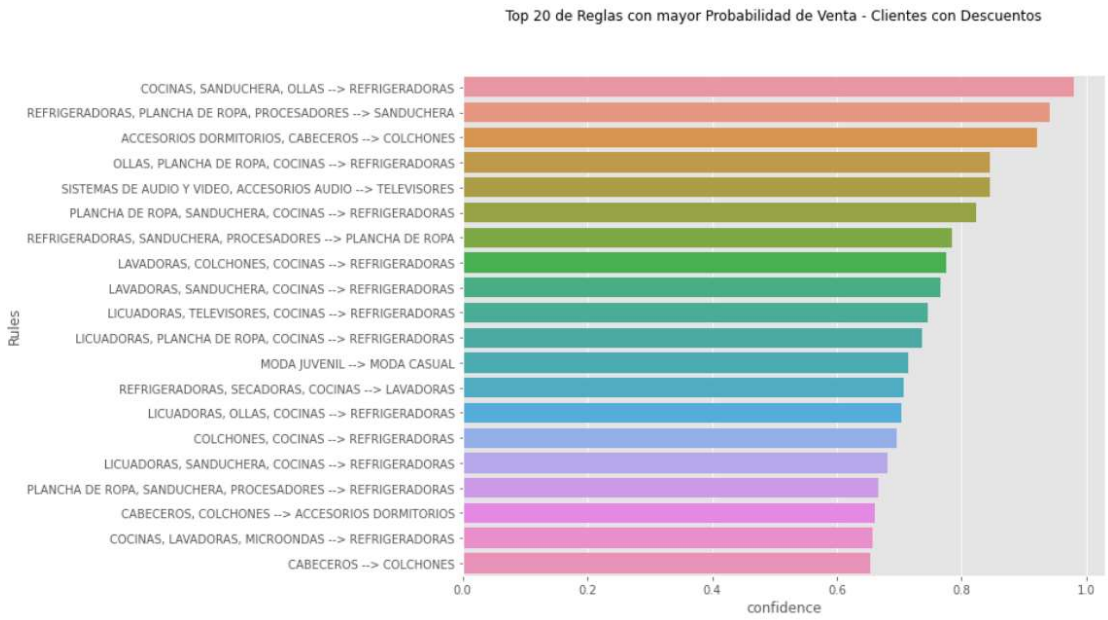


Ilustración 67 Top de reglas de asosciación del segmento Clientes con Descuentos

Estas son las reglas que tienen mayor participación en la venta, es decir las reglas con mayor support, que tienen más frecuencia o se repiten en las transacciones, indistintamente de su confianza. Estas reglas son: si el cliente compró una refrigeradora entonces es probable que comprará una cocina, si el cliente se llevó una lavadora es probable que se lleve una refrigeradora, si el cliente se compró un sistema de audio y video es probable que ahora compre un televisor. A continuación, se presentan en la Ilustración 68 las 20 reglas con mayor support en la participación de venta.

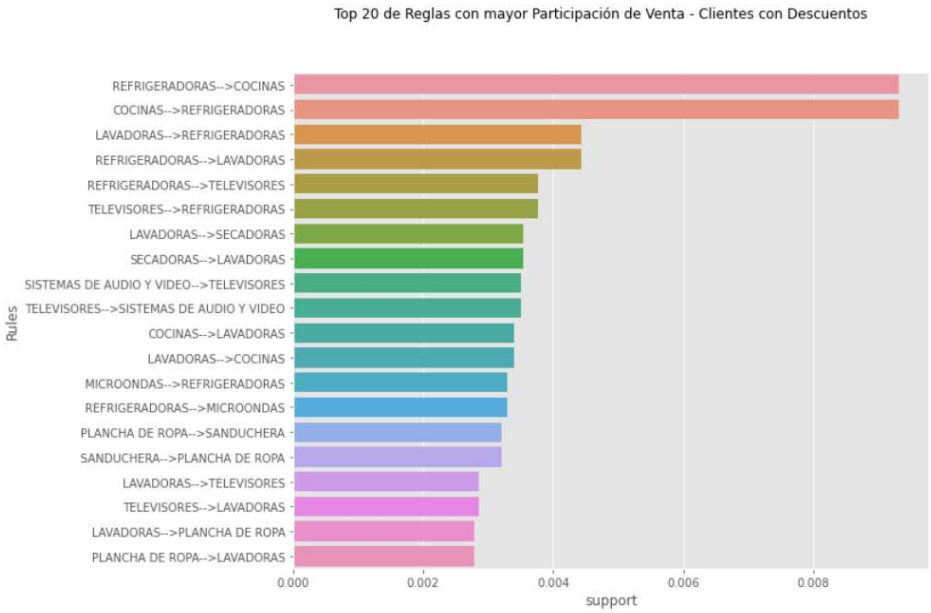


Ilustración 68 Top reglas con mayor participacion ventas del segmento Clientes con Descuentos

Clientes Tecnológicos

Este grupo de clientes comparten una similitud en sus transacciones y es que la mayoría de los clientes que conforman este clúster compran productos tecnológicos, esto lo podemos observar en los productos con mayor support de ventas, que son: televisores, teléfonos celulares, parlantes amplificados, entre otros. En la Ilustración 69 visualizamos el top 20 de los productos con mayor support de venta para este grupo de clientes.

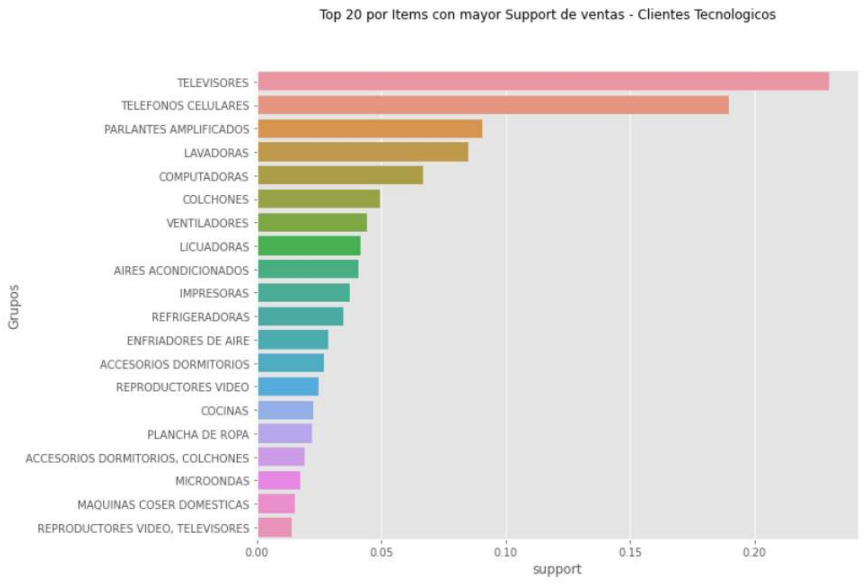


Ilustración 69 Top de líneas de productos del segmento Clientes Tecnológicos

Las reglas de asociación con mayor probabilidad de venta para los clientes tecnológicos son: los clientes que compraron accesorios dormitorios y/o cabeceros registran mayor probabilidad de comprar colchones, si el cliente compró accesorios video es probable que compre televisores, si el cliente compró impresoras y/o accesorios computo es muy probable que en su próxima compra lleve computadora. Las 20 reglas con mayor confianza de los clientes tecnológicos se presentan en la Ilustración 70.

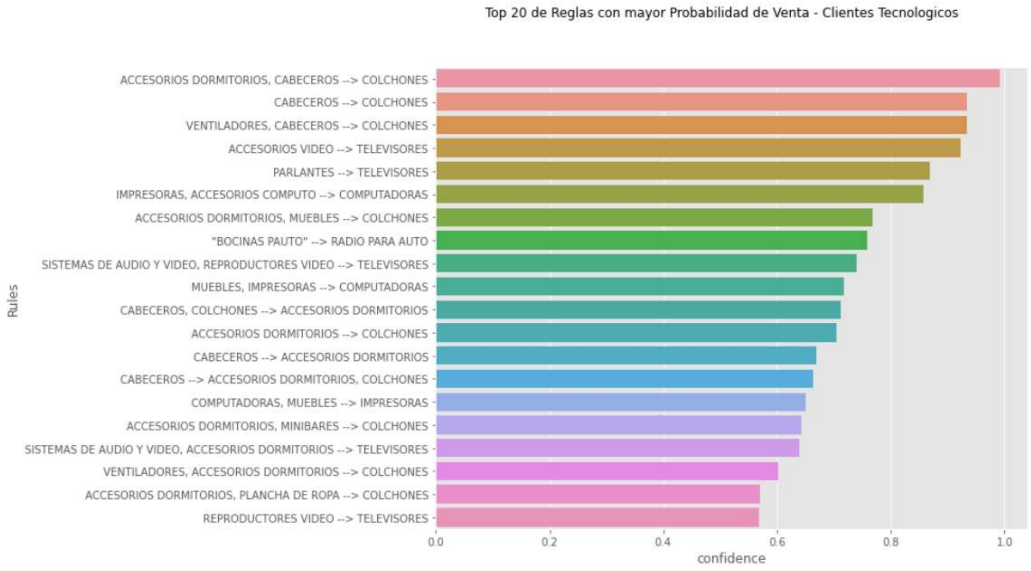


Ilustración 70 Top de reglas de asosciación del segmento Clientes Tecnológicos

A continuación, las reglas que registran mayor support en la venta, si compran accesorios dormitorios entonces llevan colchones, si compran televisores entonces comprarán reproductores video, si llevan computadores es probable que compren impresoras. En la Ilustración 71 se observan las 20 reglas con mayor support en la venta.

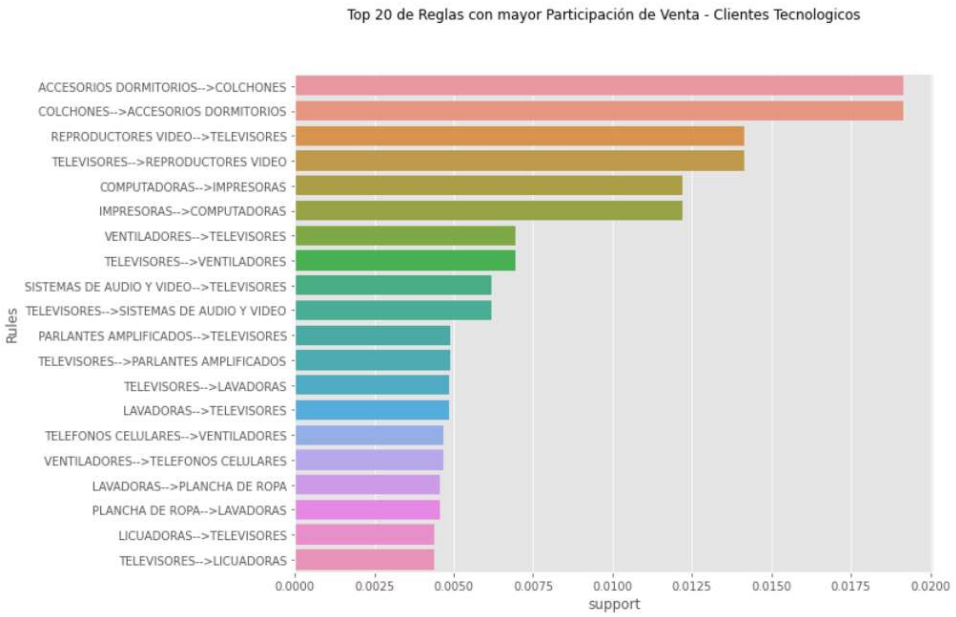


Ilustración 71 Top reglas con mayor participacion ventas del segmento Clientes Tecnológicos

Clientes con Compras Pequeñas

Los productos más frecuentes que compran la mayoría de los clientes que conforman el grupo de productos pequeños, es decir, productos que tienen un costo promedio menor a $100 son: accesorios dormitorios, plancha de ropa, reproductor video, ventiladores, sanduchera, licuadoras, ollas, entre otros. El top 20 de los productos con mayor support de ventas para este clúster se observa en la Ilustración 72.

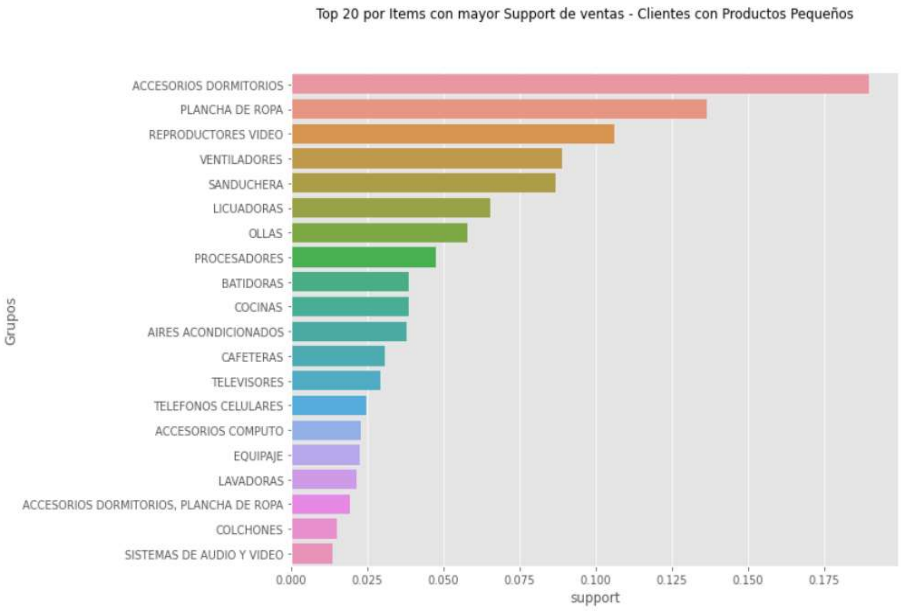


Ilustración 72 Top de líneas de productos del segmento Clientes con Compras Pequeñas

Las reglas de asociación para este grupo de clientes son: teléfonos celulares, reproductor de video y/o plancha de ropa consecuencia televisores; muebles y/o impresoras compran computadoras; teléfonos celulares, televisores y/o plancha de ropa probables de comprar reproductores de video, entre otro. Las 20 reglas con mayor confianza para los clientes que conforman este grupo se observan en la Ilustración 76.

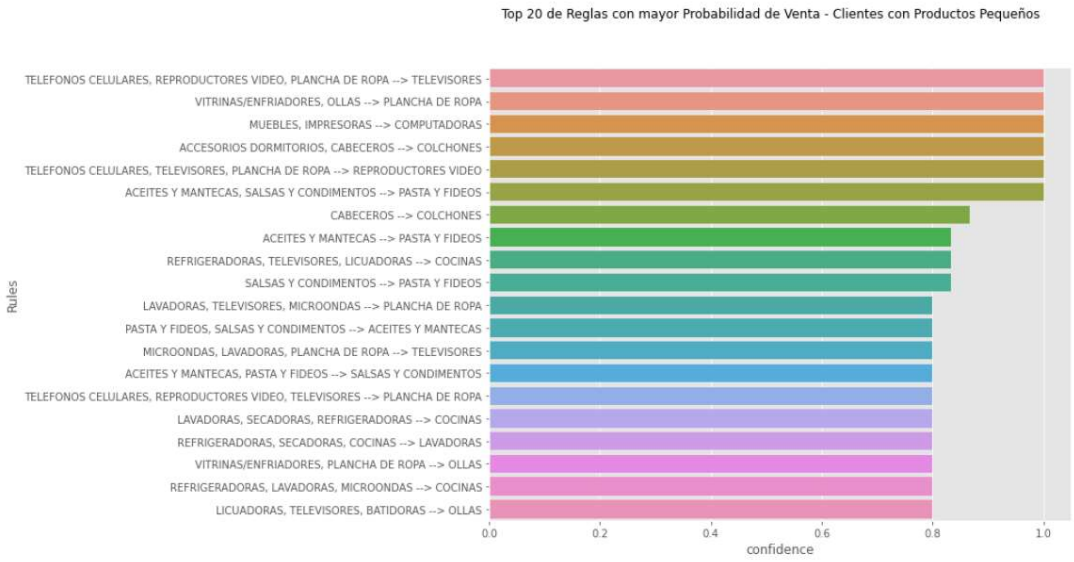


Ilustración 73 Top de reglas de asosciación del segmento Clientes con compras Pequeñas

Las reglas más frecuentes dentro de las transacciones en participación de venta son: accesorios dormitorios compran plancha de ropa, si compran sanduchera comprarán plancha de ropa, entre otros. El top 20 de las reglas con mayor support para los clientes que compran productos pequeños se muestra en la Ilustración 77.

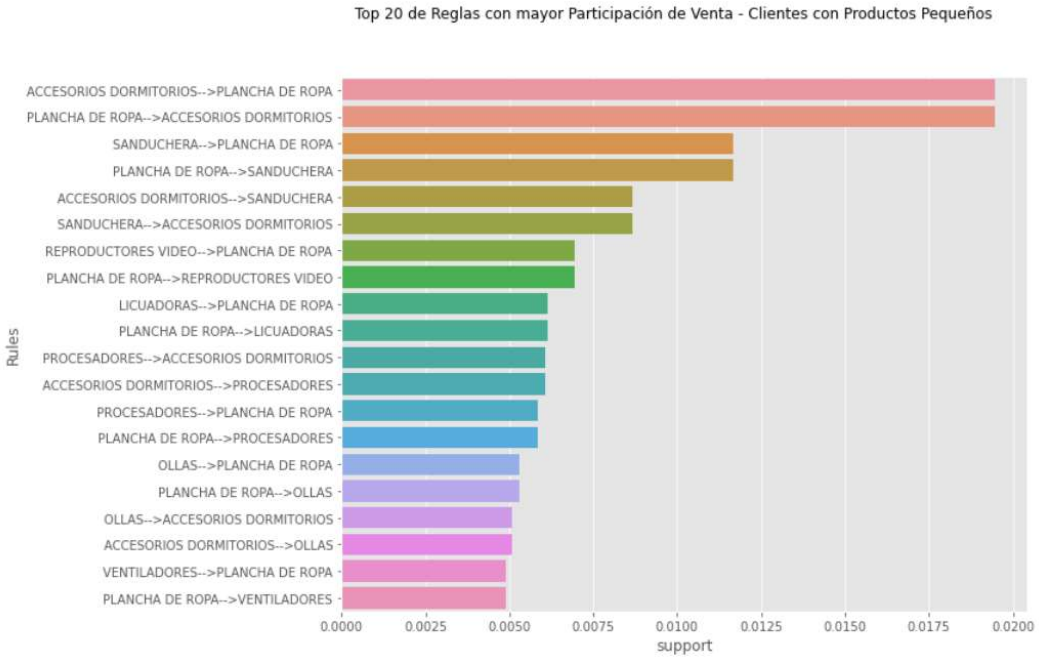


Ilustración 74 Top reglas con mayor participacion ventas del segmento Clientes con compras Pequeñas

Clientes Crediticio

Este grupo de clientes son homogéneos en su forma de compra, ya que la mayoría de los clientes que lo conforman compran a crédito, y los productos con mayor frecuencia de compra son: televisores, refrigeradoras, teléfonos celulares, lavadoras, cocinas, entre otros. El top 20 de los clientes con mayor support de ventas se muestra en la Ilustración 75.

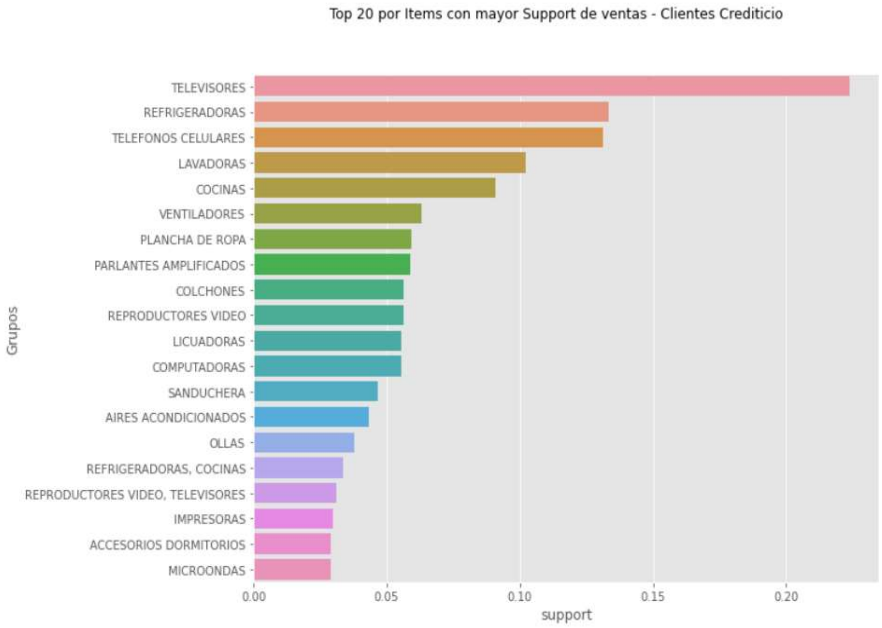


Ilustración 75 Top de líneas de productos del segmento Clientes Crediticio

Las reglas de asociación para este grupo de clientes se muestran en la Ilustración 76, en donde se observa las 20 reglas con mayor confianza.

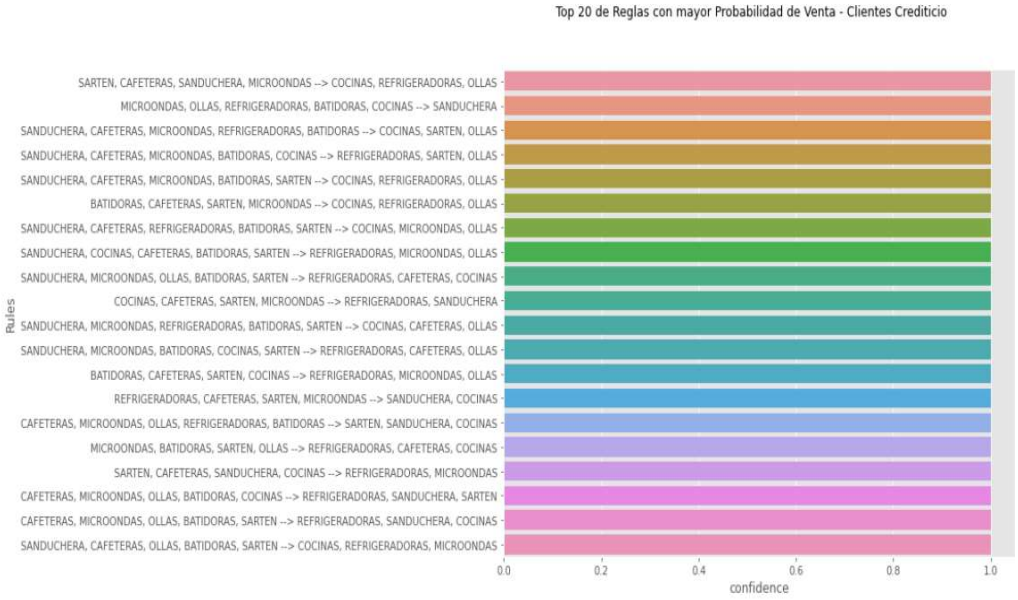


Ilustración 76 Top de reglas de asosciación del segmento Clientes Crediticio

El top 20 de las reglas con mayor support de participación para la venta se muestra en la Ilustración 77.

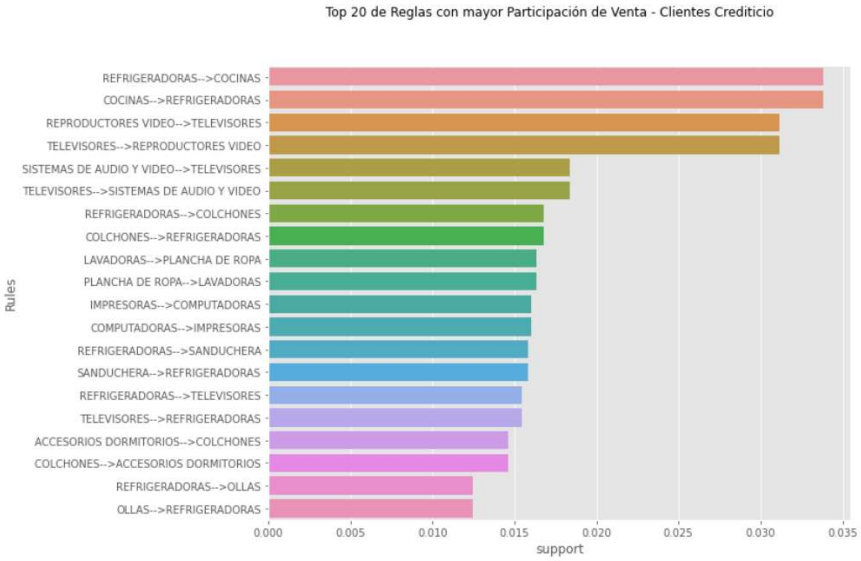


Ilustración 77 Top reglas con mayor participacion ventas del segmento Clientes Crediticio

Conclusiones y Recomendaciones

Dentro del análisis estadístico de los datos se exploraron las características de venta a nivel demográfico en donde las características monetarias discriminan “mejor” que las características sociodemográficas debido a su alta variabilidad por la tendencia del negocio.

Los patrones de venta significativos más relevante por su variabilidad son: los clientes con mayores transacciones de compras son masculinos casados entre las edades de 30 años y 50 años que prefieren comprar en las tiendas físicas en forma de crédito, la línea de producto que más rotación tiene son las líneas blancas.

El algoritmo no supervisado de clustering encontró naturalmente 4 segmentos de clientes de acuerdo con su comportamiento de venta en donde se etiquetaron los grupos de acuerdo con sus perfiles; clientes con descuentos, clientes tecnológicos, clientes compras bajas y clientes crediticio.

Los modelos clasificadores se ajustaron a los datos ya que el desarrollo de los algoritmos supervisados de machine learning registraron una alta precisión sin sobreajuste. El modelo construido que categorizará a los clientes será el de árboles de decisiones.

La herramienta de visualización implementada es la de Power BI que nos permitirá a los usuarios visualizar de forma interactiva y dinámica el comportamiento demográfico y de compra para cada segmento del cliente, adicionalmente permite el análisis a profundidad en su versión en la nube y de escritorio.

Lista de Referencias

[1] Banco Mundial, «www.bancomundial.org,» 8 Junio 2021. [En línea]. Available: https://www.bancomundial.org/es/news/feature/2021/06/08/the-global-economy-on-track-for-strong-but-uneven-growth-as-covid-19-still-weighs.

[2] O. Dev, «www.oecd.org,» 2020. [En línea]. Available: https://www.oecd.org/dev/Impacto-financiero-COVID-19-Ecuador.pdf.

[3] F. Parra, *Estadística y Machine Learning con R. RPubs Blog.,* 2017.

[4] L. MORENO GONZÁLES-PÁRAMO, *Herramientas avanzadas de análisis de datos de aplicación en ingeniería civil: Machine Learning using Python.,* 2017.

[5] Wikipedia®, «es.wikipedia.org,» 5 Marzo 2020. [En línea]. Available: https://es.wikipedia.org/wiki/Validaci%C3%B3n_cruzada#cite_note-kfcv-4.

[6] N. S. Chauhan, «datasource.ai,» 2 Septiembre 2020. [En línea]. Available: https://www.datasource.ai/es/data-science-articles/metricas-de-evaluacion-de-modelos-en-el-aprendizaje-automatico.

[7] D. Rodríguez, «analyticslane.com,» 24 Mayo 2019. [En línea]. Available: https://www.analyticslane.com/2019/05/24/los-conceptos-de-sesgo-y-varianza-en-aprendizaje-automaticos/.

[8] programmerclick, «programmerclick.com,» 2020. [En línea]. Available: https://programmerclick.com/article/8103986113/.

[9] D. Rodríguez, «www.analyticslane.com,» 31 Agosto 2018. [En línea]. Available: https://www.analyticslane.com/2018/08/31/reglas-de-asociacion-market-basket-analysis/.

[10] G. B. S. I. G. &. M. M. E. R. Contrí, «Segmentación de clientes del comercio minorista en función de los beneficios de la relación minorista-consumidor,» *Información Comercial Española, ICE: Revista de economía,* pp. (844), 219-236, 2008.

[11] G. &. O. C. Solarte, *Técnicas de clasificación y análisis de representación del conocimiento para problemas de diagnóstico, Análisis de Clúster y árboles de clasificación. Universidad tecnológica de Pereira,* Pereira, Colombia, 2009.

[12] M. Huerta Bustos, «Segmentación de Clientes de Tienda Retail Especialista,» 2011. [En línea]. Available: http://repositorio.uchile.cl/handle/2250/104296.

[13] S. Torres Villanueva, *Segmentación de Clientes de Menor Escala para una Empresa Distribuidora de Maquinaria,* 2011.

[14] P. R. G. S. S. M. &. C. R. S. J. Torrico, «Múltiples formas para segmentar el mercado de compradores online y móvil,» *Cuadernos de Estudios Empresariales,* pp. 22, 99, 2012.

[15] C. M. Cubides Proaños, *Aplicación de la minería de datos para la segmentación de clientes de la empresa DPC Studio SAS (Bachelor's thesis, Universidad Piloto de Colombia),* 2013.

[16] K. A. Morelo Tapias, *Sistema para la caracterización de perfiles de clientes de la empresa zona T (Doctoral dissertation, Universidad de Cartagena).,* 2014.

[17] F. A. P. &. P. N. A. P. Abadía, *Segmentación de clientes de una empresa comercializadora de productos de consumo masivo en la ciudad de popayán soportado en machine learning y análisis RFM (Recency, Frecuency y Money).,* 2020.

Apéndice

Anexo 1. Librerías

Librerias

In [1]:

```
# Tratamiento de datos
# ==============================================================================
import pandas as pd
import numpy as np
import statsmodels.api as sm
import seaborn as sns
import json
from scipy import stats
from fitter import Fitter, get_common_distributions
from datetime import datetime
import datetime as dt

# Gráficos
# ==============================================================================
import matplotlib.pyplot as plt
import matplotlib as mpl
from matplotlib.patches import Ellipse
from matplotlib import style
from matplotlib.colors import ListedColormap
import matplotlib.patches as mpatches
style.use('ggplot') or plt.style.use('ggplot')
%matplotlib inline
import matplotlib.font_manager
#from shapely.geometry import shape, Point
from plotnine import *

# Preprocesado y modelado
# ==============================================================================
from sklearn.decomposition import PCA
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import scale
from sklearn.preprocessing import OneHotEncoder
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn.metrics import silhouette_score
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.tree import export_graphviz
from sklearn.tree import export_text
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import validation_curve
from sklearn.model_selection import learning_curve
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.compose import ColumnTransformer
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import pairwise_distances_argmin_min
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from mlxtend.frequent_patterns import apriori # librerias para el modelo apriori(reglas de asociacion por items)
from mlxtend.frequent_patterns import association_rules
from imblearn.ensemble import BalancedBaggingClassifier

# Configuración warnings
# ==============================================================================
import warnings
warnings.filterwarnings('ignore')
warnings.simplefilter('ignore')
# instalar en la console terminal
# pip install shapely
# conda install -y -c conda-forge plotnine
# pip install mlxtend
```

Anexo 2. Transformación

Dataset

In [2]:
```
# importamos las dataset a nivel de facturas con la descripcion de los productos
# ========================================================================
data2015 = pd.read_csv ('Datos2_2015.csv', delimiter=';')
data2016 = pd.read_csv ('Datos2_2016.csv', delimiter=';')
data2017 = pd.read_csv ('Datos2_2017.csv', delimiter=';')
data2018 = pd.read_csv ('Datos2_2018.csv', delimiter=';')
data2019 = pd.read_csv ('Datos2_2019.csv', delimiter=';')
data2020 = pd.read_csv ('Datos2_2020.csv', delimiter=';')
ventas = pd.concat([data2015,data2016,data2017,data2018,data2019,data2020])
del data2015
del data2016
del data2017
del data2018
del data2019
del data2020
ventas.head()
```

In [3]:
```
# Estadistica descriptiva de la caracteristicas numericas del dataset
# ========================================================================
ventas.describe()
```

In [4]:
```
# Tipos de las caracteristicas del dataset
# ========================================================================
ventas.dtypes
```

In [5]:
```
# Informacion del dataset
# ========================================================================
ventas.info()
```

Anexo 2. Limpieza

Limpieza del Dataset

Identificación de valores faltantes N/A

In [6]:
```
# Número de datos ausentes por variable
# ========================================================================
ventas.isna().sum().sort_values()
```

Identificación de Outliers

In [7]:
```
# Boxplot de la caracteristica EDAD
# ========================================================================
boxplot = ventas.boxplot(column=['EDAD'])
```

In [8]:
```
# Estadistica descriptiva de la caracteristica EDAD
# ========================================================================
ventas["EDAD"].describe()
```

In [9]:
```
# Distribución de la caracteristica EDAD
# ========================================================================
plt.title('Distribucion de la EDAD', fontsize = 'medium')
sns.distplot(ventas.EDAD)
```

In [10]:
```
# Boxplot de la caracteristica VENTA_TOTAL
# ========================================================================
boxplot = ventas.boxplot(column=['VENTA_TOTAL'])
```

In [11]:
```
# Estadistica descriptiva de la caracteristica VENTA TOTAL
# ========================================================================
ventas["VENTA_TOTAL"].describe()
```

In [12]:
```
# Medidas de Posicion para la VENTA TOTAL
# ================================================================================
print("Q1 quantile : ", np.quantile(ventas['VENTA_TOTAL'], .25))
print("Q2 quantile : ", np.quantile(ventas['VENTA_TOTAL'], .50))
print("Q3 quantile : ", np.quantile(ventas['VENTA_TOTAL'], .75))
print("100th quantile : ", np.quantile(ventas['VENTA_TOTAL'], 1))
```

In [13]:
```
# Distribución de la caracteristica VENTA TOTAL
# ================================================================================
plt.title('Distribucion de la VENTA_TOTAL', fontsize = 'medium')
sns.distplot(ventas.VENTA_TOTAL)
```

In [14]:
```
# Boxplot de la caracteristica DESCUENTO
# ================================================================================
boxplot = ventas.boxplot(column=['DESCUENTO'])
```

In [15]:
```
# Estadistica descriptiva de la caracteristica DESCUENTO
# ================================================================================
ventas["DESCUENTO"].describe()
```

In [16]:
```
# Boxplot de la caracteristica Plazo PAGO
# ================================================================================
boxplot = ventas.boxplot(column=['plazo_pago'])
```

In [17]:
```
# Estadistica descriptiva de la caracteristica Plazo Pago
# ================================================================================
ventas["plazo_pago"].describe()
```

In [18]:
```
# Medidas de Posicion para la DESCUENTO.
# ================================================================================
print("Q1 quantile : ", np.quantile(ventas['DESCUENTO'], .25))
print("Q2 quantile : ", np.quantile(ventas['DESCUENTO'], .50))
print("Q3 quantile : ", np.quantile(ventas['DESCUENTO'], .75))
print("100th quantile : ", np.quantile(ventas['DESCUENTO'], 1))
```

In [19]:
```
# Distribución de la caracteristica DESCUENTO
# ================================================================================
plt.title('Distribucion de la DESCUENTO', fontsize = 'medium')
sns.distplot(ventas.DESCUENTO)
```

In [20]:
```
# Boxplot de la caracteristica UNIDADES_VENDIDAS
# ================================================================================
boxplot = ventas.boxplot(column=['UNIDADES_VENDIDAS'])
```

In [21]:
```
# Estadistica descriptiva de la caracteristica UNIDADES_VENDIDAS
# ================================================================================
ventas["UNIDADES_VENDIDAS"].describe()
```

In [22]:
```
# Medidas de Posicion para la UNIDADES_VENDIDAS.
# ================================================================================
print("Q1 quantile : ", np.quantile(ventas['UNIDADES_VENDIDAS'], .25))
print("Q2 quantile : ", np.quantile(ventas['UNIDADES_VENDIDAS'], .50))
print("Q3 quantile : ", np.quantile(ventas['UNIDADES_VENDIDAS'], .75))
print("100th quantile : ", np.quantile(ventas['UNIDADES_VENDIDAS'], 1))
```

In [23]:
```
# Distribución de la caracteristica UNIDADES_VENDIDAS
# ================================================================================
plt.title('Distribucion de la UNIDADES_VENDIDAS', fontsize = 'medium')
sns.distplot(ventas.UNIDADES_VENDIDAS)
```

In [24]:
```
# Distribución de las caracteristicas numéricas
fig = plt.figure(figsize=(14,12))
# subplot #1
plt.subplot(231).tick_params(labelsize = 8)
plt.title('Distribución de la EDAD', size = 12)
sns.distplot(ventas.EDAD)

#  subplot #2
plt.subplot(2,3,2).tick_params(labelsize = 8)
plt.title('Distribución de la VENTA TOTAL', size = 12)
sns.distplot(ventas.VENTA_TOTAL)

#  subplot #3
plt.subplot(233).tick_params(labelsize = 8)
plt.title('Distribución del DESCUENTO', size = 12)
sns.distplot(ventas.DESCUENTO)

#  subplot #4
plt.subplot(2,3,4).tick_params(labelsize = 8)
plt.title('BoxPlots EDAD', size = 12)
sns.boxplot(data = ventas[['EDAD']])

#  subplot #5
plt.subplot(235).tick_params(labelsize = 8)
plt.title('BoxPlots VENTA TOTAL', size = 12)
sns.boxplot(data = ventas[['VENTA_TOTAL']])

#  subplot #6
plt.subplot(236).tick_params(labelsize = 8)
plt.title('BoxPlots DESCUENTO', size = 12)
sns.boxplot(data = ventas[['DESCUENTO']])

plt.show()
```

In [25]:
```
# Tabla de la caracteristica des_sexo
# ================================================================================
ventas.des_sexo.value_counts().sort_index()
```

Out[25]:
```
FEMENINO       654378
MASCULINO      986006
NO REGISTRA    120924
Name: des_sexo, dtype: int64
```

In [26]:
```
# Tabla de la caracteristica des_estado_civil
# ================================================================================
ventas.des_estado_civil.value_counts().sort_index()
```

Out[26]:
```
CASADO          358597
DIVORCIADO       38346
NO REGISTRA     121184
SOLTERO        1193892
UNION LIBRE      40302
VIUDO             8987
Name: des_estado_civil, dtype: int64
```

In [27]:
```
# tamaño de la data inicial
ventas.shape[0]
```

Out[27]: 1761308

In [28]:
```
# Filtro excluyendo los ouliers - EDAD
ventas = ventas[(ventas['EDAD']>= 16) & (ventas['EDAD']<= 100)]
#n=ventas.shape[0]-ventas1.shape[0]
#print('Tamaño de muestra: %s' % n)
ventas.shape
```

Out[28]: (1646127, 23)

In [29]:
```
# Filtro excluyendo los ouliers - VENTA_TOTAL
ventas = ventas[(ventas['VENTA_TOTAL']>= 1) & (ventas['VENTA_TOTAL']<= 15000)]
ventas.shape
```

Out[29]: (1640967, 23)

In [30]:
```
# Filtro excluyendo los ouliers - DESCUENTO
ventas = ventas[(ventas['DESCUENTO']>= 0) & (ventas['DESCUENTO']<= 10000)]
ventas.shape
```

Out[30]: (1634423, 23)

In [31]:
```
# Filtro excluyendo los ouliers - UNIDADES_VENDIDAS
ventas = ventas[(ventas['UNIDADES_VENDIDAS']>= 1) & (ventas['UNIDADES_VENDIDAS']<= 40)]
ventas.shape
```

Out[31]: (1634422, 23)

In [32]:
```
# Filtro excluyendo la descripcion NO_REGISTRA
ventas =  ventas[(ventas['des_sexo'] != 'NO REGISTRA')]
ventas.shape
```

Out[32]: (1627705, 23)

In [33]:
```
# Filtro excluyendo la descripcion NO_REGISTRA
ventas =  ventas[(ventas['des_estado_civil'] != 'NO REGISTRA')]
ventas.shape
```

Out[33]: (1627347, 23)

Proceso de Limpieza

In [25]:
```
# Filtro excluyendo los ouliers - EDAD
ventas = ventas[(ventas['EDAD']>= 16) & (ventas['EDAD']<= 100)]

# Filtro excluyendo los ouliers - VENTA_TOTAL
ventas = ventas[(ventas['VENTA_TOTAL']>= 1) & (ventas['VENTA_TOTAL']<= 15000)]

# Filtro excluyendo los ouliers - DESCUENTO
ventas = ventas[(ventas['DESCUENTO']>= 0) & (ventas['DESCUENTO']<= 10000)]

# Filtro excluyendo los ouliers - UNIDADES_VENDIDAS
ventas = ventas[(ventas['UNIDADES_VENDIDAS']>= 1) & (ventas['UNIDADES_VENDIDAS']<= 40)]

# Filtro excluyendo la etiqueta NO_REGISTRA
ventas =  ventas[(ventas['des_sexo'] != 'NO REGISTRA')]
ventas =  ventas[(ventas['des_estado_civil'] != 'NO REGISTRA')]

# convertimos la variable FECHA en objeto fecha y creamos las variables de los tiempos
ventas['FECHA'] = pd.to_datetime(ventas['FECHA'])
ventas['month'] = ventas['FECHA'].dt.month
```

In [35]:
```
# tamaño de la data final
ventas.shape
```

Out[35]: (1627347, 23)

Anexo 3. Exploratorio

Analisis Exploratorio de los Datos

In [36]:
```
# para el analisis exploratorio, se crearon caracteristicas agrupadas en funcion de las anteriores
```

In [37]:
```
# Medidas de Posicion para la EDAD
print("Q1 quantile of arr : ", np.quantile(ventas['EDAD'], .25))
print("Q2 quantile of arr : ", np.quantile(ventas['EDAD'], .50))
print("Q3 quantile of arr : ", np.quantile(ventas['EDAD'], .75))
print("100th quantile of arr : ", np.quantile(ventas['EDAD'], 1))
```

```
Q1 quantile of arr :  29.0
Q2 quantile of arr :  38.0
Q3 quantile of arr :  50.0
100th quantile of arr :  100.0
```

In [38]:
```
# funcion de Generacion de Edad
def rango_edad(age):
    if (age <= 30):
        return "1.Jovenes"
    elif (age > 30) & (age < 50):
        return "2.Adultos"
    elif (age >= 50):
        return "3.Adultos Mayores"
```

In [39]:
```
# Medidas de Posicion para el Plazo.
print("Q1 quantile of arr : ", np.quantile(ventas['plazo_pago'], .25))
print("Q2 quantile of arr : ", np.quantile(ventas['plazo_pago'], .50))
print("Q3 quantile of arr : ", np.quantile(ventas['plazo_pago'], .75))
print("100th quantile of arr : ", np.quantile(ventas['plazo_pago'], 1))
```

```
Q1 quantile of arr :  0.0
Q2 quantile of arr :  12.0
Q3 quantile of arr :  18.0
100th quantile of arr :  60
```

In [40]:
```
# funcion del PLAZO
def rango_plazo(plazo):
    if (plazo == 0):
        return "1. 0m"
    elif (plazo >= 1) & (plazo < 6):
        return "2. 1m-6m"
    elif (plazo >= 6) & (plazo < 12):
        return "3. 6m-12m"
    elif (plazo >= 12) & (plazo < 18):
        return "4. 12m-18m"
    elif (plazo >= 18):
        return "5. >=18m"
```

In [41]:
```
# Medidas de Posicion para la Venta.
print("Q1 quantile of arr : ", np.quantile(ventas['VENTA_TOTAL'], .25))
print("Q2 quantile of arr : ", np.quantile(ventas['VENTA_TOTAL'], .50))
print("Q3 quantile of arr : ", np.quantile(ventas['VENTA_TOTAL'], .75))
print("100th quantile of arr : ", np.quantile(ventas['VENTA_TOTAL'], 1))
```

```
Q1 quantile of arr :  37.57
Q2 quantile of arr :  159.37
Q3 quantile of arr :  753.09
100th quantile of arr :  11498.76
```

In [42]:
```
# funcion Rango de Venta
def rango_venta(VENTA_TOTAL):
    if (VENTA_TOTAL < 100):
        return "1.<100"
    elif (VENTA_TOTAL >= 100) & (VENTA_TOTAL < 400):
        return "2.100-400"
    elif (VENTA_TOTAL >= 400) & (VENTA_TOTAL < 1100):
        return "3.400-1100"
    elif (VENTA_TOTAL >= 1100) :
        return "4.>1100"
```

In [43]:
```
# SE CREA LA CARACTERISTICA %DSCTO
with np.errstate(divide='ignore', invalid='ignore'):
    ventas["%DSCTO"] = np.true_divide(ventas['DESCUENTO'],ventas['VENTA_TOTAL'])
    ventas["%DSCTO"][ventas["%DSCTO"] == np.inf] = 0
    ventas["%DSCTO"] = np.nan_to_num(ventas["%DSCTO"])
```

In [44]:
```
# cuartiles del Descuento
print("Q1 quantile of arr : ", np.quantile(ventas['%DSCTO'], .25))
print("Q2 quantile of arr : ", np.quantile(ventas['%DSCTO'], .50))
print("Q3 quantile of arr : ", np.quantile(ventas['%DSCTO'], .75))
print("100th quantile of arr : ", np.quantile(ventas['%DSCTO'], 1))
```

```
Q1 quantile of arr :  0.054106014831090364
Q2 quantile of arr :  0.4254992319508449
Q3 quantile of arr :  0.4619309054318852
100th quantile of arr :  6.701583434835566
```

In [45]:
```
# funcion porct_dscto
def porct_dscto(porct_dscto):
   if (porct_dscto <0.01):
      return "1.0"
   elif (porct_dscto >= 0.01) & (porct_dscto < 0.30):
      return "2.1-30"
   elif (porct_dscto >= 0.30) & (porct_dscto < 0.50):
      return "3.30-50"
   elif (porct_dscto >= 0.50) :
      return "4.>50"
```

In [46]:
```
# funcion de Temporalidad
def rango_mes(month):
   if (month >= 1) & (month <= 3):
      return "1trim"
   elif (month >= 4) & (month <= 6):
      return "2trim"
   elif (month >= 7) & (month <= 9):
      return "3trim"
   elif (month >= 10) & (month <= 12):
      return "4trim"
```

In [49]:
```
# Creamos las caracteristicas agrupadas y agregamos al dataset
ventas['rango_compra'] = ventas['VENTA_TOTAL'].map(lambda x: rango_venta(x))
ventas['rango_edad'] = ventas['EDAD'].map(lambda x: rango_edad(x))
ventas['%_descuento'] = ventas['%DSCTO'].map(lambda x: porct_dscto(x))
ventas['plazo'] = ventas['plazo_pago'].map(lambda x: rango_plazo(x))
ventas['rango_mes'] = ventas['month'].map(lambda x: rango_mes(x))
ventas.head()
```

In [50]:
```
# Rango EDAD
plt.title('Transacciones por rango_edad', fontsize = 'medium')
sns.countplot(data = ventas, x='rango_edad')
print(ventas.rango_edad.value_counts().sort_index())
```

In [51]:
```
# Rango Compra
plt.title('Transacciones por rango_compra', fontsize = 'medium')
sns.countplot(data = ventas, x='rango_compra')
print(ventas.rango_compra.value_counts().sort_index())
```

In [52]:
```
# Rango PLAZO
plt.title('Transacciones por rango_plazo', fontsize = 'medium')
sns.countplot(data = ventas, x='plazo')
print(ventas.plazo.value_counts().sort_index())
```

In [53]:
```
# Rango MONTH
plt.title('Transacciones por rango_mes', fontsize = 'medium')
sns.countplot(data = ventas, x='rango_mes')
print(ventas.rango_mes.value_counts().sort_index())
```

In [54]:
```
# CountPlot de los features Categoricos
fig = plt.figure(figsize=(12,10))
plt.subplot(231).tick_params(labelsize = 7)
plt.title('Genero', fontsize = 12)
plt.ylabel('count', size = 7)
sns.countplot(data = ventas, x='des_sexo')

plt.subplot(2,3,2).tick_params(labelsize = 7)
plt.title('Estado Civil', fontsize = 12)
plt.ylabel('count', size = 7)
sns.countplot(data = ventas, x='des_estado_civil')

plt.subplot(233).tick_params(labelsize = 7)
plt.title('Canal', fontsize = 12)
plt.ylabel('count', size = 7)
sns.countplot(data = ventas, x='CANAL')

plt.subplot(235).tick_params(labelsize = 7)
plt.title('Forma de Pago', fontsize = 12)
plt.ylabel('count', size = 7)
sns.countplot(data = ventas, x='FORMA_PAGO')

plt.show()
```

In [55]:
```
# Tabla de la caracteristica des_sexo
# ========================================================================
ventas.des_sexo.value_counts().sort_index()
```

Out[55]:
```
FEMENINO     649318
MASCULINO    978029
Name: des_sexo, dtype: int64
```

In [56]:
```
ventas.des_estado_civil.value_counts().sort_index()
```

Out[56]:
```
CASADO          356050
DIVORCIADO       38101
SOLTERO        1184281
UNION LIBRE      39979
VIUDO             8936
Name: des_estado_civil, dtype: int64
```

In [57]:
```
ventas.CANAL.value_counts().sort_index()
```

Out[57]:
```
CALL CENTER      23946
PAGINA WEB       55311
TIENDAS        1548090
Name: CANAL, dtype: int64
```

In [58]:
```
ventas.FORMA_PAGO.value_counts().sort_index()
```

Out[58]:
```
CONTADO    655129
CREDITO    972218
Name: FORMA_PAGO, dtype: int64
```

In [59]:
```
# Descriptivo caracteristicas numéricas
# ========================================================================
ventas.select_dtypes(include=['float64', 'int']).describe()
```

In [60]:
```
# Distribución de las caracteristicas numéricas
fig = plt.figure(figsize=(14,12))
# subplot #1
plt.subplot(231).tick_params(labelsize = 8)
plt.title('Distribución de la EDAD', size = 12)
sns.distplot(ventas.EDAD)

#  subplot #2
plt.subplot(2,3,2).tick_params(labelsize = 8)
plt.title('Distribución de la VENTA TOTAL', size = 12)
sns.distplot(ventas.VENTA_TOTAL)

#  subplot #3
plt.subplot(233).tick_params(labelsize = 8)
plt.title('Distribución del DESCUENTO', size = 12)
sns.distplot(ventas.DESCUENTO)

#  subplot #4
plt.subplot(2,3,4).tick_params(labelsize = 8)
plt.title('BoxPlots EDAD', size = 12)
sns.boxplot(data = ventas[['EDAD']])

#  subplot #5
plt.subplot(235).tick_params(labelsize = 8)
plt.title('BoxPlots VENTA TOTAL', size = 12)
sns.boxplot(data = ventas[['VENTA_TOTAL']])

#  subplot #6
plt.subplot(236).tick_params(labelsize = 8)
plt.title('BoxPlots DESCUENTO', size = 12)
sns.boxplot(data = ventas[['DESCUENTO']])

plt.show()
```

In [61]:
```
# Venta Prom por GENERO
ventas_genero = ventas[(ventas['des_sexo'] != 'NO REGISTRA')].groupby(["des_sexo"])["VENTA_TOTAL"].mean().reset_index()
ventas_genero
```

In [62]:
```
# VENTA TOTAL por Edad por GENERO
plt.title('VENTA por Edad por Genero')
sns.scatterplot(data = ventas,x='EDAD',y='VENTA_TOTAL',hue='des_sexo')
```

In [63]:
```
# Venta Prom por FORMA PAGO
ventas_formapago = ventas[(ventas['des_sexo'] != 'NO REGISTRA')].groupby(["FORMA_PAGO"])["VENTA_TOTAL"].mean().reset_index()
ventas_formapago
```

In [64]:
```
# VENTA TOTAL por Edad por FORMA PAGO
plt.title('VENTA por Edad por FORMA PAGO')
sns.scatterplot(data = ventas,x='EDAD',y='VENTA_TOTAL',hue='FORMA_PAGO')
```

In [65]:
```
# Venta Prom por ESTADO CIVIL
ventas_estcivil = ventas[(ventas['des_sexo'] != 'NO REGISTRA')].groupby(["des_estado_civil"])["VENTA_TOTAL"].mean().reset_ind
ventas_estcivil
```

In [66]:
```
# VENTA TOTAL por Edad por ESTADO CIVIL
plt.title('VENTA por Edad por ESTADO CIVIL')
sns.scatterplot(data = ventas,x='EDAD',y='VENTA_TOTAL',hue='des_estado_civil')
```

In [67]:
```
# Venta Prom por MES
ventas_mes = ventas[(ventas['des_sexo'] != 'NO REGISTRA')].groupby(["rango_mes"])["VENTA_TOTAL"].mean().reset_index()
ventas_mes
```

In [68]:
```
# VENTA TOTAL por Edad por MES
plt.title('VENTA por Edad por MES')
sns.scatterplot(data = ventas,x='EDAD',y='VENTA_TOTAL',hue='rango_mes')
```

In [69]:
```
# Unidades vs Venta por Sexo
plt.title('VENTA por Unidades por Plazos')
sns.scatterplot(data = ventas,x='UNIDADES_VENDIDAS',y='VENTA_TOTAL',hue='plazo')
```

In [70]:
```
# Tabla rango de venta por Sexo.
demografrango = ventas[(ventas['des_sexo'] != 'NO REGISTRA')].pivot_table(index='rango_compra', columns=['des_sexo'],
                            values=['VENTA_TOTAL'],
                            aggfunc='count', fill_value=0)

demografrango = demografrango.reindex(demografrango['VENTA_TOTAL'].sort_values(by='rango_compra', ascending=False).index).hea

# Grafico por Rango de Venta vs Sexo
fig, ax = plt.subplots(figsize=(10,8))
_=sns.heatmap(demografrango ,annot=True,fmt=".0f",cmap="YlGnBu",
            xticklabels = ["FEMENINO","MASCULINO"])
_=plt.yticks(rotation=0)
_=plt.xlabel("Cantidad Compras")
_=plt.suptitle("Cantidad de transacciones por Genero por Rango Compra")
```

In [71]:
```
# tabla de promedio de venta, Estado civil vs Sexo.
demograf = ventas[(ventas['des_sexo'] != 'NO REGISTRA')].pivot_table(index='des_estado_civil', columns=['des_sexo'],
                            values=['VENTA_TOTAL'],
                            aggfunc='mean', fill_value=0)

demograf = demograf.reindex(demograf['VENTA_TOTAL'].sort_values(by='des_estado_civil', ascending=False).index).head(20)

# Grafico de venta, Estado civil vs Sexo
fig, ax = plt.subplots(figsize=(10,8))
_=sns.heatmap(demograf ,annot=True,fmt=".0f",cmap="YlGnBu",
            xticklabels = ["FEMENINO","MASCULINO"])
_=plt.yticks(rotation=0)
_=plt.xlabel("prom_venta_total")
_=plt.suptitle("Promedio Venta por Género por Estado Civil")
```

In [147]:
```
# Porcentaje Promedio de Descuento por Compra
descuento =  ventas.groupby(["porc_dscto_compras"])['id_persona'].nunique().reset_index()
descuento.columns = ["porc_dscto_compras","#clientes"]
plt.plot(descuento)
plt.show()
```

In [156]:
```
# Promedio de Venta y Unidad por Edad
Edad_monto=ventas.groupby(['edad'])['prom_venta_total'].agg(['mean']).reset_index()
Edad_monto['Type']='Monto'

Edad_und=ventas.groupby(['edad'])['unidades_vendidas_totales'].agg(['mean']).reset_index()
Edad_und['Type']='Unidades'

# Grafico Edad vs Promedio Venta
import seaborn as sns
x=Edad_monto.edad
y=Edad_monto.mean

sns.lmplot('edad', 'mean',  data=Edad_monto, fit_reg=True)
```

In [157]:
```
# Grafico Edad vs Unidades Venta
import seaborn as sns
x=Edad_und.edad
y=Edad_und.mean

sns.lmplot('edad', 'mean',  data=Edad_und, fit_reg=True)
```

Analisis por Productos

In [238]:
```
# funcion de Rango de Linea de Producto
def rango_linea(linea):
   if (linea == "VIDEO") or (linea == "AUDIO")or (linea == "COMPUTACION") or (linea == "TECNOLOGIA") or (linea == "TELECOMUNI
      return "TECNO"
   elif (linea == "BLANCA") :
      return "BLANCA"
   elif (linea == "CLIMATIZACION"):
     return "CLIMA"
   elif (linea == "SMALL APPLIANCE") :
      return "SMALL APPLIANCE"
   else:
      return "OTROS"
```

In [239]:
```
# Creamos las caracteristicas agrupadas y agregamos al dataset
ventas['rango_linea'] = ventas['LINEA'].map(lambda x: rango_linea(x))
```

In [242]:
```
# Rango LINEA
plt.title('Transacciones por  Rango Linea Producto', size = 14)
sns.countplot(data = ventas, x='rango_linea')
print(ventas.rango_linea.value_counts().sort_index())
```

In [243]:
```
# Linea de Producto
data2 = ventas[(ventas['LINEA'] != 'PROMOCIONALES')].groupby(["LINEA"])["UNIDADES_VENDIDAS"].count().reset_index()

# GRAFICO
sns.barplot( data = data2, x="UNIDADES_VENDIDAS", y="LINEA", orient ="h",
            order=data2.sort_values('UNIDADES_VENDIDAS',ascending = False).LINEA[:10])
_=plt.xlabel("Cantidad de Unidades")
_=plt.ylabel("Linea")
_=plt.suptitle("Top 10 de Linea de Productos")
```

In [244]:
```
# Grupo de Producto
data2 = ventas[(ventas['LINEA'] != 'PROMOCIONALES')].groupby(["GRUPO"])["UNIDADES_VENDIDAS"].count().reset_index()

# GRAFICO
sns.barplot( data = data2, x="UNIDADES_VENDIDAS", y="GRUPO", orient ="h",
             order=data2.sort_values('UNIDADES_VENDIDAS',ascending = False).GRUPO[:10])
_=plt.xlabel("Cantidad de Unidades")
_=plt.ylabel("Grupo")
_=plt.suptitle("Top 10 de Grupo de Productos")
```

In [245]:
```
# Descripcion de Producto
data2 = ventas[(ventas['LINEA'] != 'PROMOCIONALES')].groupby(["PRODUCTO"])["UNIDADES_VENDIDAS"].count().reset_index()

# GRAFICO
sns.barplot( data = data2, x="UNIDADES_VENDIDAS", y="PRODUCTO", orient ="h",
             order=data2.sort_values('UNIDADES_VENDIDAS',ascending = False).PRODUCTO[:10])
_=plt.xlabel("Cantidad de Unidades")
_=plt.ylabel("PRODUCTO")
_=plt.suptitle("Top 10 de Productos")
```

Analisis por Tiempo

In [246]:
```
# Ventas por Años
# ============================================================================================
ventas['year'] = ventas['FECHA'].dt.year
pivoted = ventas.pivot_table(index='month', columns=['year'], values='VENTA_TOTAL', aggfunc='sum' , fill_value=0)

sns.set()
plt.figure(figsize=(10,6))
p1=plt.plot(pivoted)
plt.xticks(range(1,13,1))
plt.ylabel('Monto Ventas')
plt.xlabel('Meses')
plt.legend((p1), ('2015', '2016','2017','2018', '2019','2020'), loc='upper left', shadow=True, title='Año')
plt.title('Comparación Ventas por Año', fontsize=16)
plt.show()
```

Modelo RFM

In [144]:
```
# Cuando fue la ultima vez que compraron los Clientes?
dias=ventas.groupby("dias_entre_primera_ultima_compra").agg({"dias_entre_primera_ultima_compra":"sum"})
plt.plot(dias)
plt.xlabel('Dias',fontsize=12)
plt.ylabel('Cantidad de Clientes',fontsize=12)
plt.title('Ultima dia que compraron los Clientes',fontsize=14)
plt.grid(True)
plt.legend()
```

```
WARNING:matplotlib.legend:No handles with labels found to put in legend.
```

In [143]:
```
# Cuantas Veces Compran los Clientes?
frec=ventas.groupby("frecuencia").agg({"frecuencia":"sum"})
plt.plot(frec)
plt.xlabel('Frecuencia',fontsize=12)
plt.ylabel('Cantidad de Clientes',fontsize=12)
plt.title('Frecuencia de Compra por Clientes',fontsize=14)
plt.grid(True)
plt.legend()
```

```
WARNING:matplotlib.legend:No handles with labels found to put in legend.
```

In [145]:
```
# Cuanto compraron los Clientes?
dias=ventas.groupby("VENTA_TOTAL").agg({"VENTA_TOTAL":"sum"})
plt.plot(dias)
plt.xlabel('Monto',fontsize=12)
plt.ylabel('Cantidad de Clientes',fontsize=12)
plt.title('Cuanto compraron los Clientes',fontsize=14)
plt.grid(True)
plt.legend()
```

```
WARNING:matplotlib.legend:No handles with labels found to put in legend.
```

In [168]:
```
# seleccionamos las variables para el modelo RFM
rfm =ventas[['id_persona','prom_venta_total','recencia','frecuencia']]
rfm.head()
```

Out[168]:

	id_persona	prom_venta_total	recencia	frecuencia
0	5956	574.926666	663	3
1	5972	655.096666	674	3
2	6556	425.993333	680	3
3	6590	659.965000	1212	2
4	6642	395.320000	1597	1

In [169]:
```
# Renombramos las columnas de la tabla rfm
rfm.columns = ['Id_Cliente', 'Monetary', 'Recency', 'Frecuency']
rfm.columns
```

Out[169]:
```
Index(['Id_Cliente', 'Monetary', 'Recency', 'Frecuency'], dtype='object')
```

In [170]:
```
# Asignamos puntuaciones a Recency, Frecuency y Monetary utilidando pd.qcut(). Ej: si recency es 1, el cliente es atractivo.
rfm["RecencyScore"]=pd.qcut(rfm["Recency"], 5, labels=[5,4,3,2,1])
rfm["FrecuencyScore"]=pd.qcut(rfm["Frecuency"].rank(method="first"),5, labels=[1,2,3,4,5])
rfm["MonetaryScore"]=pd.qcut(rfm["Monetary"],5, labels=[1,2,3,4,5])
rfm.head()
```

Out[170]:

	Id_Cliente	Monetary	Recency	Frecuency	RecencyScore	FrecuencyScore	MonetaryScore
0	5956	574.926666	663	3	4	3	3
1	5972	655.096666	674	3	4	3	3
2	6556	425.993333	680	3	4	3	2
3	6590	659.965000	1212	2	2	2	3
4	6642	395.320000	1597	1	1	1	2

In [171]:
```
# puntuaciones finales
(rfm['RecencyScore'].astype(str) +
 rfm['FrecuencyScore'].astype(str) +
 rfm['MonetaryScore'].astype(str)).head()
```

Out[171]:
```
0    433
1    433
2    432
3    223
4    112
dtype: object
```

In [172]:
```
# transformamos las puntuaciones en variable y agregamos a la data.
rfm["RFM_SCORE"]=(rfm['RecencyScore'].astype(str) +
                  rfm['FrecuencyScore'].astype(str) +
                  rfm['MonetaryScore'].astype(str))
rfm.head()
```

Out[172]:

	Id_Cliente	Monetary	Recency	Frecuency	RecencyScore	FrecuencyScore	MonetaryScore	RFM_SCORE
0	5956	574.926666	663	3	4	3	3	433
1	5972	655.096666	674	3	4	3	3	433
2	6556	425.993333	680	3	4	3	2	432
3	6590	659.965000	1212	2	2	2	3	223
4	6642	395.320000	1597	1	1	1	2	112

In [173]:
```
# descriptivo de la tabla RFM
rfm.describe().T
```

Out[173]:

	count	mean	std	min	25%	50%	75%	max
Id_Cliente	181082.0	869182.407296	653902.867619	11.00	381567.75	762764.50000	1.144483e+06	2809056.0
Monetary	181082.0	679.749485	478.717303	1.98	377.32	577.64375	8.532188e+02	5179.3
Recency	181082.0	1063.054335	437.539960	213.00	713.00	1043.00000	1.387000e+03	2314.0
Frecuency	181082.0	3.607377	2.811221	1.00	2.00	3.00000	5.000000e+00	94.0

In [174]:
```
# Segmentamos los clientes para identificar los buenos y malos.
#seg_map = {
#    r'[1-2][1-2]':'91. Perdidos',
#    r'[1-2][3-4]':'9. En Riesgo',
#    r'[1-2]5':'8. Por Perder',
#    r'3[1-2]':'7. Dormilones',
#    r'33':'6. Necesitan Atencion',
#    r'[3-4][4-5]':'5. Leales',
#    r'41':'4. Prometedores',
#    r'51':'3. Nuevos',
#    r'[4-5][2-3]':'2. Potenciales',
#    r'5[4-5]':'1. Campeones'
#}
seg_map = {
    r'[1-2][1-2]':'2. Malos',
    r'[1-2][3-4]':'2. Malos',
    r'[1-2]5':'2. Malos',
    r'3[1-2]':'2. Malos',
    r'33':'2. Malos',
    r'[3-4][4-5]':'1. Buenos',
    r'41':'1. Buenos',
    r'51':'1. Buenos',
    r'[4-5][2-3]':'1. Buenos',
    r'5[4-5]':'1. Buenos'
}
```

In [176]:
```
# Creamos la variable Segment
rfm['Segment']= rfm['RecencyScore'].astype(str) + rfm['FrecuencyScore'].astype(str)
rfm.head()
```

Out[176]:

	Id_Cliente	Monetary	Recency	Frecuency	RecencyScore	FrecuencyScore	MonetaryScore	RFM_SCORE	Segment
0	5956	574.926666	663	3	4	3	3	433	43
1	5972	655.096666	674	3	4	3	3	433	43
2	6556	425.993333	680	3	4	3	2	432	43
3	6590	659.965000	1212	2	2	2	3	223	22
4	6642	395.320000	1597	1	1	1	2	112	11

In [177]:
```
# reemplazamos los nombres de los segmentos
rfm['Segment']=rfm['Segment'].replace(seg_map, regex=True)
rfm.head()
```

Out[177]:

	Id_Cliente	Monetary	Recency	Frecuency	RecencyScore	FrecuencyScore	MonetaryScore	RFM_SCORE	Segment
0	5956	574.926666	663	3	4	3	3	433	1. Buenos
1	5972	655.096666	674	3	4	3	3	433	1. Buenos
2	6556	425.993333	680	3	4	3	2	432	1. Buenos
3	6590	659.965000	1212	2	2	2	3	223	2. Malos
4	6642	395.320000	1597	1	1	1	2	112	2. Malos

In [178]:
```
# Resumen del modelo RFM
rfm[["Segment","Recency","Frecuency","Monetary"]].groupby("Segment").agg(["mean","median","count"]).sort_values(by = 'Segment
```

Out[178]:

	Recency			Frecuency			Monetary		
	mean	median	count	mean	median	count	mean	median	count
Segment									
1. Buenos	697.837856	702	86997	3.588009	3	86997	636.343359	537.640	86997
2. Malos	1400.756816	1371	94085	3.625286	3	94085	719.885562	619.465	94085

In [179]:
```
# Cantidad de Clientes Buenos vs Malos
sns.factorplot('Segment',data=rfm,kind="count")
# Se evidencia data balanceada
```

In [181]:
```
# Grafico de Segmentos de clientes Buenos vs Malos
ggplot(ventas, aes(x='Recency', y='Monetary',
    color = 'Segment')) + geom_point()
```

In [183]:
```
ggplot(ventas[(ventas['Segment'] == '1. Buenos') & ((ventas['generation'] == '2.Millenial') | ( ventas['generation'] == '3.Ge
    aes(y='frecuencia', x='Monetary',
    color = 'estado_civil')) + geom_point() + facet_wrap('~sexo', ncol = 2)+ theme(figure_size=(10, 6))
```

In [184]:
```
# correlacion entre Recency, Frecuency y Monetary
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
sns.pairplot(rfm.dropna(), hue='Segment',size=4,vars=["Recency","Frecuency","Monetary"],kind='scatter')
```

Anexo 4. Segmentación

Cluster

Objetivo.- Reducir las dimensiones para agrupar objetos basados en características de venta. Clasificar los objetos en clases ó conglomerados de tal forma que cada objeto sea parecido a los que hay en el conjunto de su conglomerado.

Dendrogram of top variance explained variables

In [22]:
```
from scipy.spatial.distance import pdist, squareform
from scipy.cluster.hierarchy import linkage, dendrogram
data_dist = pdist(datos.T) # computing the distance
data_link = linkage(datos.T) # computing the linkage
```

In [23]:
```
# plot dendrogram of top variance explained variables
fig = plt.figure(figsize = (14,6))

dendrogram(data_link,labels=datos.columns)
plt.xticks(rotation = 45)
plt.xlabel('Features')
plt.ylabel('Distance')
plt.suptitle('Clustering', fontweight='bold', fontsize=18)
plt.show()
```

Metodo Elbow Curve

In [25]:
```
# Obtener el valor K, Metodo Elbow Curve
X = np.array(datos)
Nc = range(1, 10)
kmeans = [KMeans(n_clusters=i) for i in Nc]
score = [kmeans[i].fit(X).score(X) for i in range(len(kmeans))]
plt.plot(Nc,score)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```

Metodo Gaussian

In [24]:
```
# Creamos los clusters con la funcion GaussianMixture
gmm = GaussianMixture(n_components=4, covariance_type='full', random_state=123).fit(datos)
preds_gm = gmm.predict(datos)
centers_gm = gmm.means_
```

In [25]:
```
# Analisis para cada cluster
segments_gm = ['Segment {}'.format(i) for i in range(0,len(centers_gm))]
true_centers_gm = pd.DataFrame( centers_gm *100, columns = datos.columns)
true_centers_gm.index = segments_gm
np.transpose(true_centers_gm)
```

In [26]:
```
# Grafico de los clusters
from sklearn.preprocessing import StandardScaler

pca2d = PCA(n_components=7)
pca_data2d = pca2d.fit_transform(datos)
print("Total variation: {:.2f}%".format(pca2d.explained_variance_ratio_.sum()*100))

y_pred = gmm.predict(datos)
centers = gmm.means_
%matplotlib inline
plt.subplots(2,2,figsize=(10,4))

# grafico de la data Original
plt.subplot(121)
plt.scatter(pca_data2d[:,0], pca_data2d[:,1])
plt.xlabel('Dimensión 1')
plt.ylabel('Dimensión 2')
plt.title('Data Original')

# grafico de la data Clasificada
plt.subplot(122)
plt.scatter(pca_data2d[:, 0],pca_data2d[:, 1], c=y_pred, cmap='autumn')
plt.plot(centers[:, 0],centers[:, 1],'b^',markersize=10)
plt.xlabel('Dimensión 1')
plt.ylabel('Dimensión 2')
plt.title('Data Clasificada con Metodo Gaussiano')
plt.show()
```

Metodo KMeans

In [27]:
```
# standardizing the data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data_scaled = scaler.fit_transform(datos)

# statistics of scaled data
pd.DataFrame(data_scaled).describe()
```

In [28]:
```
# K means using 5 clusters and k-means++ initialization
kmeans = KMeans(n_jobs = -1, n_clusters = 4, init='k-means++')
#centroids = kmeans.cluster_centers_
kmeans.fit(data_scaled)
pred = kmeans.predict(data_scaled)
```

In [29]:
```
datos['target'] = pred
datos['target'].value_counts()
datos.head()
```

In [33]:
```
# Segmentos
fig, ax = plt.subplots(figsize=(6,4))
plt.title('Segmentos', fontsize = 'medium')
sns.countplot(data = datos, x='target')
print(datos.target.value_counts().sort_index())
```

In [34]:

```
dataGrupo= datos.groupby('target').agg({'FORMA_PAGO_CONTADO': [np.mean],
                                        'FORMA_PAGO_CREDITO': [np.mean],
                                        'plazo_1. 0m': [np.mean],
                                        'plazo_2. 1m-6m': [np.mean],
                                        'plazo_3. 6m-12m': [np.mean],
                                        'plazo_4. 12m-18m': [np.mean],
                                        'plazo_5. >=18m': [np.mean],
                                        'rango_compra_1.<100': [np.mean],
                                        'rango_compra_2.100-400': [np.mean],
                                        'rango_compra_3.400-1100': [np.mean],
                                        'rango_compra_4.>1100': [np.mean],
                                        'linea_TECNO': [np.mean],
                                        'linea_BLANCA': [np.mean],
                                        'linea_OTROS': [np.mean],
                                        'linea_CLIMA': [np.mean],
                                        'linea_SMALL APPLIANCE': [np.mean],
                                        '%_descuento_1.0': [np.mean],
                                        '%_descuento_2.1-30': [np.mean],
                                        '%_descuento_3.30-50': [np.mean],
                                        '%_descuento_4.>50': [np.mean],
                                        'regalo_0': [np.mean],
                                        'regalo_1': [np.mean]})
```

In [35]:

```
fig, ax = plt.subplots(figsize=(5,10))
_=sns.heatmap(np.transpose(dataGrupo) ,annot=True,fmt=".1%",cmap="YlGnBu",
              xticklabels = ["0","1","2" ,"3"])#
_=plt.yticks(rotation=0)
_=plt.xlabel("Segmento")
_=plt.ylabel("Variable")
_=plt.suptitle("Perfil de segmentos")
```

In [36]:

```
# Grafico de los clusters
from sklearn.preprocessing import StandardScaler

pca2d = PCA(n_components=3)
pca_data2d = pca2d.fit_transform(data_scaled)
print("Total variation: {:.2f}%".format(pca2d.explained_variance_ratio_.sum()*100))

y_pred = kmeans.predict(data_scaled)
centers = kmeans.cluster_centers_
%matplotlib inline
plt.subplots(2,2,figsize=(10,4))

# grafico de la data Original
plt.subplot(121)
plt.scatter(pca_data2d[:,0], pca_data2d[:,1])
plt.xlabel('Dimensión 1')
plt.ylabel('Dimensión 2')
plt.title('Data Original')

# grafico de la data Clasificada
plt.subplot(122)
plt.scatter(pca_data2d[:, 0],pca_data2d[:, 1], c=y_pred, cmap='autumn')
plt.plot(centers[:, 0],centers[:, 1],'b^',markersize=10)
plt.xlabel('Dimensión 1')
plt.ylabel('Dimensión 2')
plt.title('Data Clasificada con Metodo KMeans')
plt.show()
```

Anexo 5. Modelos de Clasificación

Algoritmos Supervisados - Modelos de Clasificacion

Data Training and Data Test

In [52]:
```
# Training and Test
# ------------------------------------------------------------------------------

from sklearn.utils import resample
from sklearn.preprocessing import MinMaxScaler
# Se escoge una muestra para entrenar el dataset
n = 20000 # tamaño de la muestra
datosmodelo = resample(datos , replace=False, n_samples=n, random_state=123)
X = datosmodelo.drop(columns = "target").values
y = datosmodelo['target'].values

print('Tamaño de la muestra dataset: %s' % len(datosmodelo))
porc_muestra=len(datosmodelo)/len(datos)
print('Porcentaje de muestra: %s' % porc_muestra)

# se crea los datos en train y test
x_train, x_eval, y_train, y_eval = train_test_split(X,y,test_size=0.30,train_size=0.70,random_state=123)
#X_train, X_eval, y_train, y_eval = train_test_split(X, y, test_size=0.3, stratify=y, random_state=123)

scaler = MinMaxScaler()
x_train = scaler.fit_transform(x_train)
x_eval = scaler.transform(x_eval)
```

```
Tamaño de la muestra dataset: 20000
Porcentaje de muestra: 0.0188790382262766
```

In [53]:
```
# Segmentos
fig, ax = plt.subplots(figsize=(6,4))
plt.title('Segmentos', fontsize = 'medium')
sns.countplot(data = datosmodelo, x='target')
print(datosmodelo.target.value_counts().sort_index())
```

Naive Bayes

In [54]:
```
# modelo naive bayes Gaussiano (BernoulliNB)
from sklearn.naive_bayes import GaussianNB
start = datetime.now()
clf1 = GaussianNB()
clf1.fit(x_train,y_train)
expected = y_eval
predicted = clf1.predict(x_eval)
timeScile = datetime.now() - start
timeScile=timeScile.seconds + (timeScile.microseconds * 1e-6)
print ('Time: %s' % timeScile)
```

```
Time: 0.03503
```

In [55]:
```
# Validación cruzada
# ------------------------------------------------------------------------

import sklearn.metrics as metrics
kf = KFold(n_splits=5)
score = clf1.score(x_train,y_train)
print("Metrica del modelo", score)
scores = cross_val_score(clf1, x_train, y_train, cv=kf, scoring="accuracy")
print("Metricas cross_validation", scores)
print("Media de cross_validation", scores.mean())
score_pred = metrics.accuracy_score(y_eval, predicted)
print("Metrica en Test", score_pred)
```

```
Metrica del modelo 0.9685714285714285
Metricas cross_validation [0.97035714 0.98928571 0.96428571 0.98642857 0.95535714]
Media de cross_validation 0.9731428571428573
Metrica en Test 0.9696666666666667
```

In [56]:
```
# Ajustando el modelo
clf1.fit(x_train, y_train)
print("Accuracy on training set: {:.3f}".format(clf1.score(x_train, y_train)))
print("Accuracy on test set: {:.3f}".format(clf1.score(x_eval, y_eval)))
```

```
Accuracy on training set: 0.969
Accuracy on test set: 0.970
```

In [57]:
```
# Matriz de Confusion
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(clf1, x_eval, y_eval)
plt.show()
```

In [58]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_eval, predicted)
```

Out[58]:
```
array([[1607,    6,   63,    0],
       [   0, 1074,    0,    0],
       [ 113,    0, 1247,    0],
       [   0,    0,    0, 1890]], dtype=int64)
```

In [62]:
```
# reporte de clasificador
from sklearn.metrics import classification_report
print(classification_report(y_eval, predicted))
```

```
              precision    recall  f1-score   support

           0       0.93      0.96      0.95      1676
           1       0.99      1.00      1.00      1074
           2       0.95      0.92      0.93      1360
           3       1.00      1.00      1.00      1890

    accuracy                           0.97      6000
   macro avg       0.97      0.97      0.97      6000
weighted avg       0.97      0.97      0.97      6000
```

In [53]:
```
# Bolas vs Variance
from mlxtend.evaluate import bias_variance_decomp
avg_expected_loss, avg_bias, avg_var = bias_variance_decomp(clf1, x_train, y_train, x_eval, y_eval,
                                                            loss='0-1_loss',
                                                            random_seed=123)

print('Average expected loss: %.3f' % avg_expected_loss)
print('Average bias: %.3f' % avg_bias)
print('Average variance: %.3f' % avg_var)
```

```
Average expected loss: 0.013
Average bias: 0.014
Average variance: 0.002
```

In [149]:
```
# Learning Curve
# ==============================================================================

X_entren, X_test, y_entren, y_test = train_test_split(X, y, test_size=0.3, stratify=y, random_state=123)
# Create a pipeline; This will be passed as an estimator to learning curve method
pipeline = make_pipeline(StandardScaler(), GaussianNB())

# Use learning curve to get training and test scores along with train sizes
train_sizes, train_scores, test_scores = learning_curve(estimator=pipeline, X=X_entren, y=y_entren,
                                                        cv=10, train_sizes=np.linspace(0.1, 1.0, 10),
                                                       n_jobs=1)

# Calculate training and test mean and std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Plot the learning curve
plt.plot(train_sizes, train_mean, color='blue', marker='o', markersize=5, label='Training Accuracy')
plt.fill_between(train_sizes, train_mean + train_std, train_mean - train_std, alpha=0.15, color='blue')
plt.plot(train_sizes, test_mean, color='green', marker='+', markersize=5, linestyle='--', label='Validation Accuracy')
plt.fill_between(train_sizes, test_mean + test_std, test_mean - test_std, alpha=0.15, color='green')
plt.title('Learning Curve Naive Bayes')
plt.xlabel('Training Data Size')
plt.ylabel('Model accuracy')
plt.grid()
plt.legend(loc='lower right')
plt.show()
```

K-Nearest Neighbords

In [63]:
```
# modelo K-Nearest Neighbords
from sklearn.neighbors import KNeighborsClassifier
start = datetime.now()
clf2 = KNeighborsClassifier(n_neighbors = 4,p= 2) # numero de vecinos 5; distancia 1=manhattan, 2=euclideana
clf2.fit(x_train,y_train)
expected2 = y_eval
predicted2 = clf2.predict(x_eval)
timeScile = datetime.now() - start
timeScile=timeScile.seconds + (timeScile.microseconds * 1e-6)
print ('Time: %s' % timeScile)
```

```
Time: 3.066019
```

In [64]:
```
# Validación cruzada
# ================================================================

import sklearn.metrics as metrics
kf = KFold(n_splits=5)
score2 = clf2.score(x_train,y_train)
print("Metrica del modelo", score2)
scores2 = cross_val_score(clf2, x_train, y_train, cv=kf, scoring="accuracy")
print("Metricas cross_validation", scores2)
print("Media de cross_validation", scores2.mean())
score_pred2 = metrics.accuracy_score(y_eval, predicted2)
print("Metrica en Test", score_pred2)
```

```
Metrica del modelo 0.9989285714285714
Metricas cross_validation [0.99892857 0.99892857 0.9975     0.99821429 0.99928571]
Media de cross_validation 0.9985714285714286
Metrica en Test 0.9985
```

In [65]:
```
# Ajustando el modelo
clf2.fit(x_train, y_train)
print("Accuracy on training set: {:.3f}".format(clf2.score(x_train, y_train)))
print("Accuracy on test set: {:.3f}".format(clf2.score(x_eval, y_eval)))
```

```
Accuracy on training set: 0.999
Accuracy on test set: 0.999
```

In [66]:
```
# reporte de clasificador
from sklearn.metrics import classification_report
print(classification_report(y_eval, predicted2))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1676
           1       1.00      1.00      1.00      1074
           2       1.00      1.00      1.00      1360
           3       1.00      1.00      1.00      1890

    accuracy                           1.00      6000
   macro avg       1.00      1.00      1.00      6000
weighted avg       1.00      1.00      1.00      6000
```

In [67]:
```
# Matriz de Confusion
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(clf2, x_eval, y_eval)
plt.show()
```

In [114]:

```
# Bolas vs Variance
from mlxtend.evaluate import bias_variance_decomp
avg_expected_loss, avg_bias, avg_var = bias_variance_decomp(clf2, x_train, y_train, x_eval, y_eval,
                                                            loss='0-1_loss',
                                                            random_seed=123)

print('Average expected loss: %.3f' % avg_expected_loss)
print('Average bias: %.3f' % avg_bias)
print('Average variance: %.3f' % avg_var)
```

```
Average expected loss: 0.004
Average bias: 0.003
Average variance: 0.003
```

In [635]:

```
# Learning Curve
# ================================================================================

X_entren, X_test, y_entren, y_test = train_test_split(X, y, test_size=0.3, stratify=y, random_state=123)
#x_train, x_eval, y_train, y_eval = train_test_split(X,y,test_size=0.30,train_size=0.70,random_state=123)
# Create a pipeline; This will be passed as an estimator to learning curve method
pipeline = make_pipeline(StandardScaler(), KNeighborsClassifier(n_neighbors = 4,p= 2))

# Use learning curve to get training and test scores along with train sizes
train_sizes, train_scores, test_scores = learning_curve(estimator=pipeline, X=X_entren, y=y_entren,
                                                        cv=10, train_sizes=np.linspace(0.1, 1.0, 10),
                                                       n_jobs=1)

# Calculate training and test mean and std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Plot the learning curve
plt.plot(train_sizes, train_mean, color='blue', marker='o', markersize=5, label='Training Accuracy')
plt.fill_between(train_sizes, train_mean + train_std, train_mean - train_std, alpha=0.15, color='blue')
plt.plot(train_sizes, test_mean, color='green', marker='+', markersize=5, linestyle='--', label='Validation Accuracy')
plt.fill_between(train_sizes, test_mean + test_std, test_mean - test_std, alpha=0.15, color='green')
plt.title('Learning Curve KNN')
plt.xlabel('Training Data Size')
plt.ylabel('Model accuracy')
plt.grid()
plt.legend(loc='lower right')
plt.show()
```

Support Vector Machine (Kernel Linear)

In [68]:
```
# modelo SVM Linear
from sklearn import svm
start = datetime.now()
clf3 = svm.SVC(C=1, kernel='linear')
clf3.fit(x_train,y_train)
expected3 = y_eval
predicted3 = clf3.predict(x_eval)
timeScile = datetime.now() - start
timeScile=timeScile.seconds + (timeScile.microseconds * 1e-6)
print ('Time: %s' % timeScile)
```

```
Time: 0.228792
```

In [69]:
```
# Validación cruzada
# ================================================================

import sklearn.metrics as metrics
kf = KFold(n_splits=5)
score3 = clf3.score(x_train,y_train)
print("Metrica del modelo", score3)
scores3 = cross_val_score(clf3, x_train, y_train, cv=kf, scoring="accuracy")
print("Metricas cross_validation", scores3)
print("Media de cross_validation", scores3.mean())
score_pred3 = metrics.accuracy_score(y_eval, predicted3)
print("Metrica en Test", score_pred3)
```

```
Metrica del modelo 0.9993571428571428
Metricas cross_validation [0.99785714 0.99928571 0.99857143 0.99964286 0.99928571]
Media de cross_validation 0.9989285714285716
Metrica en Test 0.9993333333333333
```

In [70]:
```
# Ajustando el modelo
clf3.fit(x_train, y_train)
print("Accuracy on training set: {:.3f}".format(clf3.score(x_train, y_train)))
print("Accuracy on test set: {:.3f}".format(clf3.score(x_eval, y_eval)))
```

```
Accuracy on training set: 0.999
Accuracy on test set: 0.999
```

In [71]:
```
# reporte de clasificador
from sklearn.metrics import classification_report
print(classification_report(y_eval, predicted3))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1676
           1       1.00      1.00      1.00      1074
           2       1.00      1.00      1.00      1360
           3       1.00      1.00      1.00      1890

    accuracy                           1.00      6000
   macro avg       1.00      1.00      1.00      6000
weighted avg       1.00      1.00      1.00      6000
```

In [72]:
```
# Matriz de Confusion
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(clf3, x_eval, y_eval)
plt.show()
```

In [73]:

```
# Bolas vs Variance
from mlxtend.evaluate import bias_variance_decomp
avg_expected_loss, avg_bias, avg_var = bias_variance_decomp(clf3, x_train, y_train, x_eval, y_eval,
                                                            loss='0-1_loss',random_seed=123)

print('Average expected loss: %.3f' % avg_expected_loss)
print('Average bias: %.3f' % avg_bias)
print('Average variance: %.3f' % avg_var)
```

```
Average expected loss: 0.001
Average bias: 0.001
Average variance: 0.000
```

In [636]:

```
# Learning Curve
# ============================================================================

# Create a pipeline; This will be passed as an estimator to learning curve method
pipeline = make_pipeline(StandardScaler(), svm.SVC(C=1, kernel='linear'))

# Use learning curve to get training and test scores along with train sizes
train_sizes, train_scores, test_scores = learning_curve(estimator=pipeline, X=X_entren, y=y_entren,
                                                        cv=10, train_sizes=np.linspace(0.1, 1.0, 10),
                                                       n_jobs=1)

# Calculate training and test mean and std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Plot the learning curve
plt.plot(train_sizes, train_mean, color='blue', marker='o', markersize=5, label='Training Accuracy')
plt.fill_between(train_sizes, train_mean + train_std, train_mean - train_std, alpha=0.15, color='blue')
plt.plot(train_sizes, test_mean, color='green', marker='+', markersize=5, linestyle='--', label='Validation Accuracy')
plt.fill_between(train_sizes, test_mean + test_std, test_mean - test_std, alpha=0.15, color='green')
plt.title('Learning Curve SVM Linear')
plt.xlabel('Training Data Size')
plt.ylabel('Model accuracy')
plt.grid()
plt.legend(loc='lower right')
plt.show()
```

Support Vector Machine (Kernel RBF)

In [74]:
```
# modelo SVM RBF
from sklearn import svm
start = datetime.now()
clf4 = svm.SVC(C=1, kernel='rbf', # kernel gaussiano(funcion de base radial)
               gamma=0.1) # Grado del kernel
clf4.fit(x_train,y_train)
expected4 = y_eval
predicted4 = clf4.predict(x_eval)
timeScile = datetime.now() - start
timeScile=timeScile.seconds + (timeScile.microseconds * 1e-6)
print ('Time: %s' % timeScile)
```

```
Time: 1.34619
```

In [75]:
```
# Validación cruzada
# ============================================================================

import sklearn.metrics as metrics
kf = KFold(n_splits=5)
score4 = clf4.score(x_train,y_train)
print("Metrica del modelo", score4)
scores4 = cross_val_score(clf4, x_train, y_train, cv=kf, scoring="accuracy")
print("Metricas cross_validation", scores4)
print("Media de cross_validation", scores4.mean())
score_pred4 = metrics.accuracy_score(y_eval, predicted4)
print("Metrica en Test", score_pred4)
```

```
Metrica del modelo 0.9985
Metricas cross_validation [0.99678571 0.9975     0.99714286 0.99964286 0.9975     ]
Media de cross_validation 0.9977142857142857
Metrica en Test 0.9981666666666666
```

In [76]:
```
# Ajustando el modelo
clf4.fit(x_train, y_train)
print("Accuracy on training set: {:.3f}".format(clf4.score(x_train, y_train)))
print("Accuracy on test set: {:.3f}".format(clf4.score(x_eval, y_eval)))
```

```
Accuracy on training set: 0.999
Accuracy on test set: 0.998
```

In [77]:
```
# reporte de clasificador
from sklearn.metrics import classification_report
print(classification_report(y_eval, predicted4))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1676
           1       0.99      1.00      1.00      1074
           2       1.00      1.00      1.00      1360
           3       1.00      1.00      1.00      1890

    accuracy                           1.00      6000
   macro avg       1.00      1.00      1.00      6000
weighted avg       1.00      1.00      1.00      6000
```

In [78]:
```
# Matriz de Confusion
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(clf4, x_eval, y_eval)
plt.show()
```

In [79]:
```
# Bolas vs Variance
from mlxtend.evaluate import bias_variance_decomp
avg_expected_loss, avg_bias, avg_var = bias_variance_decomp(clf4, x_train, y_train, x_eval, y_eval,
                                                            loss='0-1_loss',random_seed=123)

print('Average expected loss: %.3f' % avg_expected_loss)
print('Average bias: %.3f' % avg_bias)
print('Average variance: %.3f' % avg_var)
```

```
Average expected loss: 0.002
Average bias: 0.002
Average variance: 0.001
```

In [637]:
```
# Learning Curve
# ================================================================================

# Create a pipeline; This will be passed as an estimator to learning curve method
pipeline = make_pipeline(StandardScaler(), svm.SVC(C=1, kernel='rbf', gamma=0.1))

# Use learning curve to get training and test scores along with train sizes
train_sizes, train_scores, test_scores = learning_curve(estimator=pipeline, X=X_entren, y=y_entren,
                                                        cv=10, train_sizes=np.linspace(0.1, 1.0, 10),
                                                       n_jobs=1)

# Calculate training and test mean and std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Plot the learning curve
plt.plot(train_sizes, train_mean, color='blue', marker='o', markersize=5, label='Training Accuracy')
plt.fill_between(train_sizes, train_mean + train_std, train_mean - train_std, alpha=0.15, color='blue')
plt.plot(train_sizes, test_mean, color='green', marker='+', markersize=5, linestyle='--', label='Validation Accuracy')
plt.fill_between(train_sizes, test_mean + test_std, test_mean - test_std, alpha=0.15, color='green')
plt.title('Learning Curve SVM RBF')
plt.xlabel('Training Data Size')
plt.ylabel('Model accuracy')
plt.grid()
plt.legend(loc='lower right')
plt.show()
```

Arboles de Decision

In [80]:
```
# Definimos la profundidad del arbol
DecisionTreeClassifier(class_weight=None, criterion='entropy',
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0,
                       random_state=123, splitter='best')
```

Out[80]:
```
DecisionTreeClassifier(criterion='entropy', random_state=123)
```

In [81]:
```
# creando el modelo inicial sin control de profundidad, va a continuar hasta
clf5 = DecisionTreeClassifier(criterion='entropy' )
clf5.fit(x_train, y_train) # Ajustando el modelo
# Ajustando el modelo
clf5.fit(x_train, y_train)
print("Accuracy on training set: {:.3f}".format(clf5.score(x_train, y_train)))
print("Accuracy on test set: {:.3f}".format(clf5.score(x_eval, y_eval)))
```

```
Accuracy on training set: 1.000
Accuracy on test set: 1.000
```

In [82]:
```
# profundidad del arbol de decisión.
clf5.tree_.max_depth
```

Out[82]:
```
7
```

In [83]:
```
# Grafico de ajuste del arbol de decisión
train_prec = []
eval_prec = []
max_deep_list = list(range(2, 9))

for deep in max_deep_list:
    arbol3 = DecisionTreeClassifier(criterion='entropy', max_depth=deep)
    arbol3.fit(x_train, y_train)
    train_prec.append(arbol3.score(x_train, y_train))
    eval_prec.append(arbol3.score(x_eval, y_eval))

# graficar los resultados.
plt.plot(max_deep_list, train_prec, color='r', label='entrenamiento')
plt.plot(max_deep_list, eval_prec, color='b', label='evaluacion')
plt.title('Grafico de ajuste arbol de decision')
plt.legend()
plt.ylabel('precision')
plt.xlabel('cant de nodos')
plt.show()
```

In [84]:
```
# Modelo Arboles Decisiones
from sklearn.tree import DecisionTreeClassifier
start = datetime.now()
clf5 = DecisionTreeClassifier(criterion='entropy', max_depth = 5 )
expected5 = y_eval
predicted5 = clf4.predict(x_eval)
timeScile = datetime.now() - start
timeScile=timeScile.seconds + (timeScile.microseconds * 1e-6)
print ('Time: %s' % timeScile)
```

```
Time: 0.505215
```

In [85]:
```
# Validación cruzada
# --------------------------------------------------------------------

import sklearn.metrics as metrics
kf = KFold(n_splits=5)
#score5 = clf5.score(x_train,y_train)
#print("Metrica del modelo", score5)
scores5 = cross_val_score(clf5, x_train, y_train, cv=kf, scoring="accuracy")
print("Metricas cross_validation", scores5)
print("Media de cross_validation", scores5.mean())
score_pred5 = metrics.accuracy_score(y_eval, predicted5)
print("Metrica en Test", score_pred5)
```

```
Metricas cross_validation [0.99821429 0.99892857 0.99892857 0.99964286 0.99892857]
Media de cross_validation 0.9989285714285714
Metrica en Test 0.9988333333333334
```

In [86]:
```
# Ajustando el modelo
clf5.fit(x_train, y_train)
print("Accuracy on training set: {:.3f}".format(clf5.score(x_train, y_train)))
print("Accuracy on test set: {:.3f}".format(clf5.score(x_eval, y_eval)))
```

```
Accuracy on training set: 0.999
Accuracy on test set: 0.999
```

In [129]:
```
# Estructura del arbol creado
# --------------------------------------------------------------------------------
fig, ax = plt.subplots(figsize=(28, 12))

print(f"Profundidad del árbol: {clf5.get_depth()}")
print(f"Número de nodos terminales: {clf5.get_n_leaves()}")

plot = plot_tree(
            decision_tree = clf5,
            feature_names = datos.drop(columns = "target").columns,
            class_names   = 'target',
            filled        = True,
            impurity      = False,
            fontsize      = 10,
            precision     = 2,
            ax            = ax
       )
```

```
Profundidad del árbol: 5
Número de nodos terminales: 20
```

In [130]:
```
from sklearn.tree import export_text
texto_modelo = export_text(
                    decision_tree = clf5,
                    feature_names = list(datos.drop(columns = "target").columns)
                )
print(texto_modelo)
```

In [87]:
```
# reporte de clasificador
from sklearn.metrics import classification_report
predicted5 = clf5.predict(x_eval)
print(classification_report(y_eval, predicted5))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1676
           1       1.00      1.00      1.00      1074
           2       1.00      1.00      1.00      1360
           3       1.00      1.00      1.00      1890

    accuracy                           1.00      6000
   macro avg       1.00      1.00      1.00      6000
weighted avg       1.00      1.00      1.00      6000
```

In [88]:
```
# Matriz de Confusion
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(clf5, x_eval, y_eval)
plt.show()
```

In [89]:
```
# Bias vs Variance
from mlxtend.evaluate import bias_variance_decomp
avg_expected_loss, avg_bias, avg_var = bias_variance_decomp(clf5, x_train, y_train, x_eval, y_eval,
                                                            loss='0-1_loss',random_seed=123)

print('Average expected loss: %.3f' % avg_expected_loss)
print('Average bias: %.3f' % avg_bias)
print('Average variance: %.3f' % avg_var)
```

```
Average expected loss: 0.001
Average bias: 0.002
Average variance: 0.001
```

In [638]:
```
# Learning Curve
# ============================================================================

# Create a pipeline; This will be passed as an estimator to learning curve method
pipeline = make_pipeline(StandardScaler(), DecisionTreeClassifier(criterion='entropy', max_depth = 5))

# Use learning curve to get training and test scores along with train sizes
train_sizes, train_scores, test_scores = learning_curve(estimator=pipeline, X=X_entren, y=y_entren,
                                                        cv=10, train_sizes=np.linspace(0.1, 1.0, 10),
                                                       n_jobs=1)

# Calculate training and test mean and std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Plot the learning curve
plt.plot(train_sizes, train_mean, color='blue', marker='o', markersize=5, label='Training Accuracy')
plt.fill_between(train_sizes, train_mean + train_std, train_mean - train_std, alpha=0.15, color='blue')
plt.plot(train_sizes, test_mean, color='green', marker='+', markersize=5, linestyle='--', label='Validation Accuracy')
plt.fill_between(train_sizes, test_mean + test_std, test_mean - test_std, alpha=0.15, color='green')
plt.title('Learning Curve Decision Tree')
plt.xlabel('Training Data Size')
plt.ylabel('Model accuracy')
plt.grid()
plt.legend(loc='lower right')
plt.show()
```

In [133]:
```
# Podar el arbol
tree = DecisionTreeClassifier(random_state=123)

path = tree.cost_complexity_pruning_path(x_train, y_train) #Compute the pruning path during Minimal Cost-Complexity Pruning.
ccp_alphas, impurities = path.ccp_alphas, path.impurities

fig, ax = plt.subplots()
ax.plot(ccp_alphas, impurities, marker='o', drawstyle="steps-post")
ax.set_xlabel("effective alpha")
ax.set_ylabel("total impurity of leaves")
ax.set_title("Total Impurity vs effective alpha for training set")
```

In [134]:
```
ccp_alphas = ccp_alphas[:-1]
impurities = impurities[:-1]
fig, ax = plt.subplots()
ax.plot(ccp_alphas, impurities, marker='o', drawstyle="steps-post")
ax.set_xlabel("effective alpha")
ax.set_ylabel("total impurity of leaves")
ax.set_title("Total Impurity vs effective alpha for training set")
```

Redes Neuronales

In [90]:
```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
import multiprocessing
```

Se procede a crear 4 modelos en orden creciente de complejidad (número de neuronas y capas), para comprobar cómo la arquitectura de la red afecta a su capacidad de aprendizaje.

In [91]:
```
# Modelos
# ==============================================================
modelo_1 = MLPClassifier(
                hidden_layer_sizes=(5),
                learning_rate_init=0.01,
                solver = 'lbfgs',
                max_iter = 1000,
                random_state = 123)

modelo_2 = MLPClassifier(
                hidden_layer_sizes=(10),
                learning_rate_init=0.01,
                solver = 'lbfgs',
                max_iter = 1000,
                random_state = 123)

modelo_3 = MLPClassifier(
                hidden_layer_sizes=(20, 20),
                learning_rate_init=0.01,
                solver = 'lbfgs',
                max_iter = 5000,
                random_state = 123)

modelo_4 = MLPClassifier(
                hidden_layer_sizes=(50, 50, 50),
                learning_rate_init=0.01,
                solver = 'lbfgs',
                max_iter = 5000,
                random_state = 123)

modelo_1.fit(X=X, y=y)
modelo_2.fit(X=X, y=y)
modelo_3.fit(X=X, y=y)
modelo_4.fit(X=X, y=y)
```

In [97]:

```
# Gráfico de predicciones
# ------------------------------------------------------------------------------
fig, axs = plt.subplots(2, 2, figsize=(12,8))
axs = axs.flatten()
grid_x1 = np.linspace(start=min(# Gráfico de predicciones
# ------------------------------------------------------------------------------
fig, axs = plt.subplots(2, 2, figsize=(12,8))
axs = axs.flatten()
grid_x1 = np.linspace(start=min(X['FORMA_PAGO_CONTADO']), stop=max(X['FORMA_PAGO_CREDITO']), num=100)
grid_x2 = np.linspace(start=min(X['FORMA_PAGO_CONTADO']), stop=max(X['FORMA_PAGO_CREDITO']), num=100)
xx, yy = np.meshgrid(grid_x1, grid_x2)
X_grid = np.column_stack([xx.flatten(), yy.flatten()])

for i, modelo in enumerate([modelo_1, modelo_2, modelo_3, modelo_4]):

    predicciones = modelo.predict(X_grid)

    for j in np.unique(predicciones):
        axs[i].scatter(
            x = X_grid[predicciones == j, 0],
            y = X_grid[predicciones == j, 1],
            c = plt.rcParams['axes.prop_cycle'].by_key()['color'][j],
            #marker = 'o',
            alpha = 0.3,
            label= f"Grupo {j}"
        )

    for j in np.unique(y):
        axs[i].scatter(
            x = X[y == j, 0],
            y = X[y == j, 1],
            c = plt.rcParams['axes.prop_cycle'].by_key()['color'][j],
            marker    = 'o',
            edgecolor = 'black'
        )

    axs[i].set_title(f"Capas ocultas: {modelo.hidden_layer_sizes}")
    axs[i].axis('off')
axs[0].legend();), stop=max(X[:, 0]), num=100)
grid_x2 = np.linspace(start=min(X[:, 1]), stop=max(X[:, 1]), num=100)
xx, yy = np.meshgrid(grid_x1, grid_x2)
X_grid = np.column_stack([xx.flatten(), yy.flatten()])

for i, modelo in enumerate([modelo_1, modelo_2, modelo_3, modelo_4]):

    predicciones = modelo.predict(X_grid)

    for j in np.unique(predicciones):
        axs[i].scatter(
            x = X_grid[predicciones == j, 0],
            y = X_grid[predicciones == j, 1],
            c = plt.rcParams['axes.prop_cycle'].by_key()['color'][j],
            #marker = 'o',
            alpha = 0.3,
            label= f"Grupo {j}"
        )

    for j in np.unique(y):
        axs[i].scatter(
            x = X[y == j, 0],
            y = X[y == j, 1],
            c = plt.rcParams['axes.prop_cycle'].by_key()['color'][j],
            marker    = 'o',
            edgecolor = 'black'
        )

    axs[i].set_title(f"Capas ocultas: {modelo.hidden_layer_sizes}")
    axs[i].axis('off')
axs[0].legend();
```

Optimizacion de hiperparametro

se muestra cómo afectan al aprendizaje algunos de los hiperparámetros más influyentes. Como los 2 predictores tienen la misma escala, no es estrictamente necesarios aplicarles una normalización previo entrenamiento.

In [92]:
```
# Número de neuronas
# ==============================================================================
from sklearn.neural_network import MLPClassifier
param_grid = {'hidden_layer_sizes':[1, 5, 10, 15, 25, 50]}

grid = GridSearchCV(
        estimator = MLPClassifier(
                        learning_rate_init=0.01,
                        solver = 'lbfgs',
                        alpha  = 0,
                        max_iter = 5000,
                        random_state = 123
                    ),
        param_grid = param_grid,
        scoring    = 'accuracy',
        cv         = 5,
        refit      = True,
        return_train_score = True
       )

_ = grid.fit(X, y)
```

In [93]:
```
fig, ax = plt.subplots(figsize=(6, 3.84))
scores = pd.DataFrame(grid.cv_results_)
scores.plot(x='param_hidden_layer_sizes', y='mean_train_score', yerr='std_train_score', ax=ax)
scores.plot(x='param_hidden_layer_sizes', y='mean_test_score', yerr='std_test_score', ax=ax)
ax.set_ylabel('accuracy')
ax.set_xlabel('número de neuronas')
ax.set_title('Error de validacion cruzada');
```

In [94]:
```
# Learning rate
# ==============================================================================
param_grid = {'learning_rate_init':[0.0001, 0.001, 0.01, 0.1, 1, 10, 100]}

grid = GridSearchCV(
        estimator = MLPClassifier(
                        hidden_layer_sizes=(10),
                        solver = 'adam',
                        alpha  = 0,
                        max_iter = 5000,
                        random_state = 123
                    ),
        param_grid = param_grid,
        scoring    = 'accuracy',
        cv         = 5,
        refit      = True,
        return_train_score = True
       )

_ = grid.fit(X, y)
```

In [95]:
```
fig, ax = plt.subplots(figsize=(6, 3.84))
scores = pd.DataFrame(grid.cv_results_)
scores.plot(x='param_learning_rate_init', y='mean_train_score', yerr='std_train_score', ax=ax)
scores.plot(x='param_learning_rate_init', y='mean_test_score', yerr='std_test_score', ax=ax)
ax.set_xscale('log')
ax.set_xlabel('log(learning rate)')
ax.set_ylabel('accuracy')
ax.set_title('Error de validacion cruzada');
```

In [96]:
```
# Espacio de búsqueda de cada hiperparámetro
# ==============================================================================
param_distributions = {
    'hidden_layer_sizes': [(10), (10, 10), (20, 20)],
    'alpha': np.logspace(-3, 3, 7),
    'learning_rate_init': [0.001, 0.01, 0.1],
}

# Búsqueda por validación cruzada
# ==============================================================================
grid = RandomizedSearchCV(
        estimator  = MLPClassifier(solver = 'lbfgs', max_iter= 2000),
        param_distributions = param_distributions,
        n_iter     = 50, # Número máximo de combinaciones probadas
        scoring    = 'accuracy',
        n_jobs     = multiprocessing.cpu_count() - 1,
        cv         = 3,
        verbose    = 0,
        random_state = 123,
        return_train_score = True
       )

grid.fit(X = X, y = y)

```

In [97]:
```
# Resultados del grid
# ------------------------------------------------------------------------
resultados = pd.DataFrame(grid.cv_results_)
resultados.filter(regex = '(param.*|mean_t|std_t)')\
    .drop(columns = 'params')\
    .sort_values('mean_test_score', ascending = False)\
    .head(10)
```

In [98]:
```
modelo = grid.best_estimator_
modelo
```

Out[98]:
```
MLPClassifier(alpha=0.001, hidden_layer_sizes=(10, 10), learning_rate_init=0.01,
              max_iter=2000, solver='lbfgs')
```

In [99]:
```
# modelo Redes Neuronales
start = datetime.now()
clf6 = MLPClassifier(
                hidden_layer_sizes=(10,10),
                learning_rate_init=0.1,
                solver = 'lbfgs',
                max_iter = 2000,
                random_state = 123)
clf6.fit(x_train,y_train)
expected6 = y_eval
predicted6 = clf6.predict(x_eval)
timeScile = datetime.now() - start
timeScile=timeScile.seconds + (timeScile.microseconds * 1e-6)
print ('Time: %s' % timeScile)
```

```
Time: 1.169646
```

In [100]:
```
# Validación cruzada
# ------------------------------------------------------------------------

import sklearn.metrics as metrics
kf = KFold(n_splits=5)
score6 = clf6.score(x_train,y_train)
print("Metrica del modelo", score6)
scores6 = cross_val_score(clf6, x_train, y_train, cv=kf, scoring="accuracy")
print("Metricas cross_validation", scores6)
print("Media de cross_validation", scores6.mean())
score_pred6 = metrics.accuracy_score(y_eval, predicted6)
print("Metrica en Test", score_pred6)
```

```
Metrica del modelo 1.0
Metricas cross_validation [0.99964286 0.99964286 0.99928571 1.         0.99964286]
Media de cross_validation 0.9996428571428572
Metrica en Test 0.9996666666666667
```

In [101]:
```
# Ajustando el modelo
clf6.fit(x_train, y_train)
print("Accuracy on training set: {:.3f}".format(clf6.score(x_train, y_train)))
print("Accuracy on test set: {:.3f}".format(clf6.score(x_eval, y_eval)))
```

```
Accuracy on training set: 1.000
Accuracy on test set: 1.000
```

In [102]:
```
# reporte de clasificador
from sklearn.metrics import classification_report
print(classification_report(y_eval, predicted6))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1676
           1       1.00      1.00      1.00      1074
           2       1.00      1.00      1.00      1160
           3       1.00      1.00      1.00      1890

    accuracy                           1.00      6000
   macro avg       1.00      1.00      1.00      6000
weighted avg       1.00      1.00      1.00      6000
```

In [103]:
```
# Matriz de Confusion
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(clf6, x_eval, y_eval)
plt.show()
```

In [104]:
```
# Bias vs Variance
from mlxtend.evaluate import bias_variance_decomp
avg_expected_loss, avg_bias, avg_var = bias_variance_decomp(clf6, x_train, y_train, x_eval, y_eval,
                                                            loss='0-1_loss',random_seed=123)

print('Average expected loss: %.3f' % avg_expected_loss)
print('Average bias: %.3f' % avg_bias)
print('Average variance: %.3f' % avg_var)
```

```
Average expected loss: 0.001
Average bias: 0.000
Average variance: 0.000
```

In [639]:
```
# Learning Curve
# ================================================================================================

# Create a pipeline; This will be passed as an estimator to learning curve method
pipeline = make_pipeline(StandardScaler(), MLPClassifier(hidden_layer_sizes=(10),learning_rate_init=0.1,
                                                         solver = 'lbfgs',max_iter = 2000, random_state = 123))

# Use learning curve to get training and test scores along with train sizes
train_sizes, train_scores, test_scores = learning_curve(estimator=pipeline, X=X_entren, y=y_entren,
                                                        cv=10, train_sizes=np.linspace(0.1, 1.0, 10),
                                                       n_jobs=1)

# Calculate training and test mean and std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Plot the learning curve
plt.plot(train_sizes, train_mean, color='blue', marker='o', markersize=5, label='Training Accuracy')
plt.fill_between(train_sizes, train_mean + train_std, train_mean - train_std, alpha=0.15, color='blue')
plt.plot(train_sizes, test_mean, color='green', marker='+', markersize=5, linestyle='--', label='Validation Accuracy')
plt.fill_between(train_sizes, test_mean + test_std, test_mean - test_std, alpha=0.15, color='green')
plt.title('Learning Curve Redes Neuronales')
plt.xlabel('Training Data Size')
plt.ylabel('Model accuracy')
plt.grid()
plt.legend(loc='lower right')
plt.show()
```

Reglas de Asociación

Modelo Apriori

In [62]:
```
# cruzamos a la data original, los cluster clasificados
base1 = base
base1['FECHA'] = pd.to_datetime(base1['FECHA'])

base1 = base1.merge(ventas, how='left', left_on=["FACTURA","FECHA",
                                                 "Id_Cliente"], right_on=["FACTURA","FECHA",
                                                                          "Id_Cliente"],
                                    copy=True, indicator=False,    validate=None)
```

In [63]:
```
base1 =  base1[(base1['LINEA'] != 'PROMOCIONALES')]
```

In [64]:
```
# se crean las bases para cada segmento del cluster.
segm0 = base1.query('target == 0')
segm1 = base1.query('target == 1')
segm2 = base1.query('target == 2')
segm3 = base1.query('target == 3')
```

In [121]:
```
# Tipo string las variables Factura y Grupo de Producto
ventasP['GRUPO'] = ventasP['GRUPO'].str.strip()
ventasP.dropna(axis=0, subset=['FACTURA'],inplace=True)
ventasP['FACTURA']= ventasP['FACTURA'].astype('str')
```

In [122]:
```
# se crea la data, transacciones por items
basket = (ventasP.groupby(['FACTURA','GRUPO'])['UNIDADES_VENDIDAS']
                .sum().unstack().reset_index().fillna(0)
                .set_index('FACTURA'))
```

In [123]:
```
# Soporte para cada items
frequent_itemsets = apriori(basket > 0, min_support=0.0001,use_colnames=True).sort_values(by = 'support', ascending = False)
frequent_itemsets.head()
```

In [124]:
```
# frecuencia de los itemsets
frequent_itemsets['itemsets_cl'] = frequent_itemsets['itemsets'].astype(str)
frequent_itemsets['itemsets_cl'] = frequent_itemsets['itemsets_cl'].str.replace("frozenset","").str.replace("'","").str.repl
frequent_itemsets_top = frequent_itemsets.head(20)
```

In [125]:
```
# Grafico
fig,axs = plt.subplots(figsize=(10,8))
sns.barplot( data = frequent_itemsets_top , x='support', y='itemsets_cl', orient ="h",
            order=frequent_itemsets_top.sort_values('support',ascending = False).itemsets_cl)
_=plt.xlabel("support")
_=plt.ylabel("Grupos")
_=plt.suptitle("Top 20 por Items con mayor Support de ventas - Clientes Crediticio")
```

In [128]:
```
## Modelo de Reglas de Asociacion (Algoritmo Apriori)
# Reglas con mayor Nro de Confianza.
rules = association_rules(frequent_itemsets, metric="confidence",min_threshold=0.0001).sort_values('confidence',
                                                                                                   ascending = False)
rules.head()
```

In [129]:
```
rules['rules'] = rules['antecedents'].astype(str) +' --> '+ rules['consequents'].astype(str)
rules = rules.head(20)
```

In [130]:
```
rules['rules'] = rules['rules'].str.replace("frozenset","").str.replace("'","").str.replace("{","").str.replace("(","").str.
rules.head()
```

In [131]:
```
# Grafico
fig,axs = plt.subplots(figsize=(10,8))
sns.barplot( data = rules , x='confidence', y='rules', orient ="h",
            order=rules.sort_values('confidence',ascending = False).rules)
_=plt.xlabel("confidence")
_=plt.ylabel("Rules")
_=plt.suptitle("Top 20 de Reglas con mayor Probabilidad de Venta - Clientes Crediticio")
```