ANÁLISIS EXPLORATORIO DE DATOS DEL CONSUMO DEL SERVICIO PÚBLICO DE ENERGÍA ELÉCTRICA EN BOGOTÁ.

Autores:
Gersain Guzmán Vargas
Diego Sánchez Bernal

Director:
Elio H. Cables Pérez, PhD

Universidad Antonio Nariño.
Facultad de Ingeniería de Sistemas
Especialización en Gobierno de Datos
Bogotá D.C.
2021.

CONTENIDO

LISTA DE FIGURAS.

LISTA DE TABLAS.

RESUMEN

El acceso a los servicios públicos impacta positivamente en la calidad de vida de los ciudadanos y es actualmente una necesidad intrínseca de la condición humana, impactando en factores primordiales como la alimentación, la salud, la educación, entre otras. No obstante, condiciones socioeconómicas de los ciudadanos, o problemas económicos o de cobertura de las empresas proveedoras vulneran el derecho fundamental de contar con los servicios públicos básicos.

El presente trabajo de investigación se centra en un buscar una relación entre las diferentes variables de los datos abiertos disponibles en el Sistema Único de Información (SUI) realizando un análisis de los datos del servicio público domiciliario de energía eléctrica de la ciudad de Bogotá Colombia, con el fin de entender mejor el comportamiento de consumo y pago de los subscriptores y con una muestra aleatoria de los datos desarrollar un modelo de aprendizaje automático que ayude a predecir potenciales suscriptores morosos.

INTRODUCCIÓN.

El estado social de derecho es la humanización de las políticas de un estado en pro del beneficio común de sus habitantes. Colombia en la Constitución Política de 1991 en su primer artículo se declara estado social de derecho y en artículos posteriores reconoce la importancia de los servicios públicos como de vital importancia para la nación y dejando en manos de terceros la responsabilidad de su suministro bajo la supervisión del estado. En adelante se abordará una exploración de los datos del consumo de energía en Bogotá de la última década y se obtendrá conocimiento que permita entender de mejor manera los hábitos de consumo y la viabilidad de los algoritmos de machine learning para identificar a futuros morosos.

La necesidad del gobierno colombiano de promover un estado transparente y de conceder acceso a los datos de las entidades del estado a los ciudadanos para que estos puedan ser utilizados libremente, la Superintendencia de Servicios Públicos Domiciliarios en cumplimiento de este lineamiento puso a disposición de la comunidad el Sistema Único de Información de Servicios Públicos SUI, agrupando los datos de más de 3.000 prestadores de servicios públicos heterogéneos en cuanto a los servicios prestados, tamaño, presupuesto, calidad y cobertura. Esta agrupación de datos trae por si misma grandes retos en la calidad de datos por la asimetría propia de las diferentes fuentes lo cual dificulta el entendimiento de los mismos, por lo cual el SUI dispone de formatos unificados de la información como lo es el Formato 2 de información comercial residencial de consumo y facturación del servicio de energía eléctrica y el cual será la fuente de datos principal para el presente estudio.

Así mismo, la disposición de estos datos al público en general permite a los agentes responsables de la regulación, vigilancia y control del servicio público de energía eléctrica a nivel nacional y territorial, recibir retroalimentación de la comunidad que utiliza estos datos para generar conocimiento.

2. PLANTEAMIENTO DEL PROBLEMA.

¿Cómo identificar los potenciales clientes en mora, sobre la base de los datos abiertos e históricos de información comercial de consumo y facturación del servició público domiciliario en la ciudad de Bogotá?

3. OBJETIVOS

3.1. OBJETIVO GENERAL.

Aplicar un modelo de aprendizaje automático que emplea técnicas de decisión en su entrenamiento con datos abiertos sobre el consumo de energía eléctrica y permita predecir potenciales clientes morosos en la ciudad de Bogotá.

3.2. OBJETIVOS ESPECIFICOS.

- Caracterizar los métodos existentes para el procesamiento de los datos abiertos históricos de información comercial residencial del servicio público de energía eléctrica en la ciudad de Bogotá.

- Implementar procesos internos de extracción y carga de datos, que permitan llevar la información de las fuentes originales a una fuente de almacenamiento.

- Procesar la información comercial residencial del servicio Público de Energía proveniente del SUI y la DTGE de la Superintendencia de Servicios Públicos Domiciliarios.

- Aplicar el modelo de clasificación de aprendizaje automático sobre la base de los datos abiertos e históricos de información comercial del consumo del servició público de energía eléctrica domiciliario en la ciudad de Bogotá.

- Generar salidas graficas del análisis exploratorio de los datos y el modelo predictivo de los potenciales clientes morosos del servicio público domiciliario de energía eléctrica en la ciudad de Bogotá, utilizando el lenguaje de programación Python.

4. MARCO DE REFERENCIA

4.1. MARCO TEORICO

De acuerdo a la (Art. 1 Constitución Política de Colombia de 1991) Colombia es un estado social de derecho que busca alejar al estado del poder absoluto, además de otorgar a los habitantes derechos sociales de igualdad y oportunidades ante el estado. El estado mediante esta constitución, y las leyes 142 y 143 de 1994 deja atrás la responsabilidad de proveedor de los servicios públicos y asume el rol de regulador.

Permitiendo así a organizaciones públicas y privadas brindar servicios a los ciudadanos, solventar las necesidades y aceptar los retos propios del crecimiento de las grandes urbes. En este aspecto, el artículo 365 de la constitución política, precisa que los servicios públicos son inherentes al propósito social del Estado. Es deber del Estado asegurar la prestación eficiente a todos los habitantes del territorio nacional. Enfatiza que estos pueden ser prestados por empresas de Estado o por particulares, todo en pro del bienestar social. La constitución no establece la prestación de estos servicios de forma gratuita, solamente se encargará de ejercer vigilancia y control.

Las empresas prestadoras de servicios públicos domiciliarios requieren del pago oportuno de las tarifas de servicios públicos para poder garantizar sus servicios y operar sin fallos, la historia reciente de Colombia muestra casos como el de la liquidada empresa prestadora de Electricaribe, empresa encargada del suministro de energía eléctrica en la Costa Caribe colombiana y que debido a sus problemas financieros dejo de invertir en su red eléctrica generando serios problemas en el suministro de energía.

La Corte constitucional mediante la sentencia C-580/92 considera la norma, que los servicios públicos se prestarán "en beneficio de la comunidad", como red de apoyo que, elimina la posibilidad que se establezca la prestación de estos, en condiciones favorables a solo una parte de la población. Tomando en cuenta las diferencias de los distintos estratos socioeconómicos que participan como usuarios de los servicios públicos y de acuerdo a la capacidad económica, establece un régimen tarifario diferencial, que distribuye el costo de los servicios, de acuerdo con la capacidad económica del usuario del servicio, para evitar que sea igual la tarifa para los sectores más ricos de la comunidad que para los más pobres.

Debido al alto incremento de la población conforme a la llegada de personas migrantes de otros países y la situación actual de pandemia que enfrenta la ciudad de Bogotá se evidencia un gran aumento de los datos de facturación y consumo del servicio público de energía eléctrica residencial en comparación con años anteriores, así mismo surge la necesidad de controlar, inspeccionar y vigilar la información reportada por las empresas prestadoras del servicio público de energía eléctrica en la ciudad de Bogotá por lo cual es preciso realizar el análisis y procesamiento de la información, aprovechando el auge y el surgimiento de nuevas tecnologías de solución a datos se busca analizar y predecir el comportamiento de consumo y pago de los suscriptores o clientes de las empresas prestadoras del servicio público domiciliario de energía eléctrica en la ciudad de Bogotá empleando una metodología de trabajo en equipo y herramientas informáticas.

Como metodología de trabajo en equipo para ciencia de datos adoptamos la CRISP DM en ingles “(Cross-Industry Standard Process for Data Mining)” (CRISP-DM, 1999) esta metodología es ágil e iterativa, de la cual se apropia y emplea las etapas de conocimiento del negocio, adquisición y comprensión de los datos, modelado e Implementación.

Anaconda Navigator como interfaz gráfica que facilita la creación de entornos de desarrollo y la ejecución rápida de utilidades como Jupyter Notebook que es una aplicación web de código que se ejecuta en un ambiente local y facilita crear y compartir documentos al igual que generar gráficas y visualizaciones, de forma similar funciona Google Colaboratory funciona como un notebook el cual se ejecuta en la nube utilizando la infraestructura suministrada por google por lo cual el internet es una necesidad fundamental para su funcionamiento; sin embargo y pese a las diferencias, ambas utilidades gracias a su integración con el lenguaje de programación Python y a las diferentes librerías como Pandas, NumPy, Scikit learn, entre otras, se utilizan en la Ciencia de Datos (Data Science) para crear y compartir proyectos que incluyen el tratamiento, la visualización, el análisis, y la transformación de los datos; igualmente otra de las librerías relevantes es Pymongo la cual brinda conectividad para realizar operaciones CRUD (Crear Leer, Actualizar y Borrar) en el popular gestor de bases de datos MongoDB uno de los principales referentes de las bases de NoSQL de la actualidad.

Por otro lado, la herramienta de la suite de Pentaho Data Integration de Hitachi Vantara (PDI) también conocido como Kettle permite aplicar técnicas de Extracción, Transformación y Carga (ETL) de una manera más ágil y eficiente.

Adicionalmente PDI o Kettle ofrece datos analíticos muy precisos, eliminando las codificaciones complejas y minimizando las cargas de trabajo manual difícil de desplegar como son los procesos de migración, explotación de Big Data o integración con otras herramientas.

Como base de datos NoSQL el gestor de base de datos documental Mongo DB ofrece una gran capacidad de almacenamiento a través del manejo de colecciones Llave Valor JSON o BSON el cual permite generar consultas más agiles, adicionalmente permite integrar el almacenamiento mediante drivers a diversas herramientas que emplean técnicas de Extracción Transformación y carga (ETL) como lo es Kettle y herramientas de análisis de datos como lo es el entorno de Anaconda Navigator y su Notebook Jupyter a través de su lenguaje Python con el driver de PyMongo.

4.2. ESTADO DEL ARTE

Actualmente de acuerdo a lo investigado en Colombia se han identificado proyectos de investigación similares enfocados al consumo de energía eléctrica y al análisis de la información del servicio público energético en la ciudad de Santiago de Cali, como lo es el caso de la *“Evaluación de índices de consumo de energía eléctrica en el sector residencial”.* En esta investigación consideran aspectos socioeconómicos y técnicos en el periodo comprendido entre el 2019 y 2020 aplicados por la universidad Autónoma de Occidente. Los índices y el análisis muestran evidencia de los comportamientos presentados mes a mes, resaltando que el 2020 el índice de consumo de energía eléctrica aumento al 146.26% en el mes de abril hasta el mes de diciembre manteniendo valores superiores al 100%.

En las tarifas eléctricas en el sector residencial su índice fue elevado con respecto al 2019 (ARÉVALO, 2021)

En Barranquilla existe un estudio realizado en la universidad del Norte referente al "*Pronóstico y Análisis del consumo de energía eléctrica en usuarios residenciales de Barranquilla*", está investigación se enfoca a través de encuestas en realizar un inventario de los electrodomésticos que se encuentran en las viviendas y el uso en horas a lo largo del durante la semana, con el fin de tener información detallada de los consumos y de esa forma podrán predecir el consumo promedio de un usuario. (Padilla Pineda, 2020)

Por otro lado, la Investigación, *"La equidad de las tarifas de los servicios públicos y su impacto en la capacidad de pago de los hogares de Bogotá*", realizada por el Centro de Investigaciones para el Desarrollo de la Universidad Nacional de Colombia, sede Bogotá. Su objetivo fue evaluar el impacto que tiene el pago de las tarifas de los servicios públicos en el consumo de las familias, para esto realiza una serie de encuestas de capacidad de pago dirigida a una muestra representativa de la ciudad de Bogotá (12.140 hogares) para indagar por: el conjunto de factores que inciden en la calidad de vida de las familias, la estructura de consumo de los hogares y como la forma de pago de los servicios públicos domiciliarios incide en el poder adquisitivo a través de los efectos ingreso y precio. Como resultados se encuentra que el sistema tarifario logra garantizar la estabilidad financiera, pero es inequitativo y afecta de manera negativa la capacidad de pago de las familias de escasos recursos. (Borrero, 2004)

En Alemania se realizó una investigación de implementación de modelos basados en el aprendizaje automático para el "*Pronóstico del consumo eléctrico de prosumidores con fuentes de energía renovables*", se define prosumidor como productor y consumidor de energía eléctrica el objetivo de la investigación fue desarrollar modelos basados en métodos de aprendizaje automático para pronosticar el consumo eléctrico horario de un prosumidor residencial ubicado en Alemania. Esta investigación, desarrolla una metodología para construir modelos de predicción basados en los métodos seleccionados «k-vecinos más cercanos» y «redes neuronales artificiales» (en inglés, k-Nearest Neighbors y Artificial Neural Networks, respectivamente), los cuales son aplicados a las mediciones históricas de consumo eléctrico de quince meses en combinación con datos meteorológicos de temperatura local, los modelos k-Nearest Neighbors y Artificial Neural Networks mostraron resultados de exactitud aceptables con un error porcentual absoluto medio en el orden del 30 % en tres escenarios diferentes con períodos de pronóstico de 48, 24 y 1 horas. (Peña, 2019)

Como factor común de estos estudios esta la utilización de lenguajes interpretados como Python y las librerías dispuestas para la aplicación de machine learning, en el presente estudio se trabaja con Python, un lenguaje de programación multi paradigma y multi plataforma, el cual posee una licencia de código abierto GNU para el tratamiento y uso de los datos, tomando importante relevancia las librerías NumPy, Pandas, Scikit-learn y PyMongo, las cuales permiten realizar un estudio de la data eficiente siendo muy versátil y de fácil manejo.

Finalmente, se utiliza la librería de Scikit learn para hacer uso de los modelos de aprendizaje automático y predictivo. Este tratamiento de los datos se realizó con el apoyo de las herramientas notebooks Google Colab y el entorno de Anaconda Navigator. Se empleó la herramienta Pentaho Data Integration de Hitachi Vantara para el procedimiento de extracción, transformación y carga de datos a una base de datos NOSQL orientada a documentos en la cual se almacenaron los datos que se extrajeron de la fuente.

4.3. IMPACTO

Se obtuvieron resultados significativos al realizar el análisis exploratorio sobre la base de los datos abiertos históricos del servicio público de energía eléctrica en la ciudad de Bogotá de los años 2019 y 2020 aplicando aprendizaje automático para identificar potenciales suscriptores morosos del servicio público domiciliario de energía eléctrica ; estos resultados aportarán en las actividades de control, inspección y vigilancia que realiza la Dirección Técnica de Energía y Gas de la Superintendencia de Servicios Públicos Domiciliarios SSPD, al igual que para evaluar la eficiencia de los árboles de decisiones para predecir fenómenos equivalentes con datos similares.

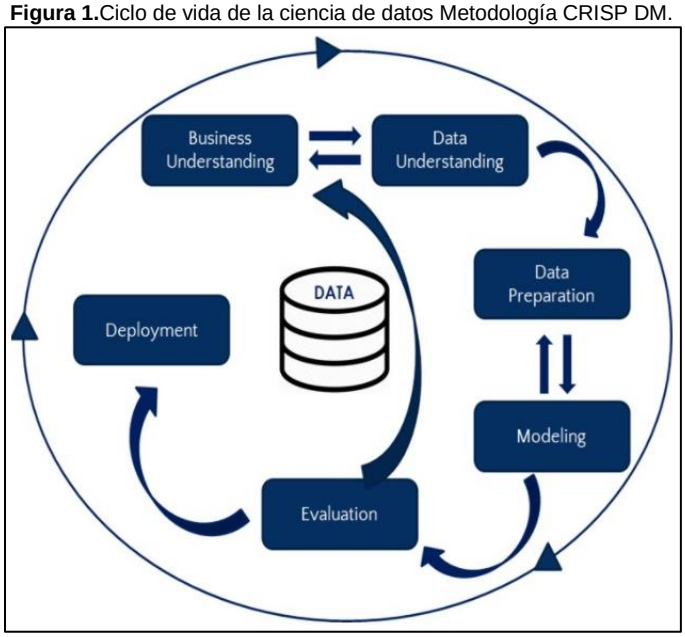
4.4. COMPONENTE DE INNOVACIÓN

Como componente de innovación mediante el análisis y exploración de datos (EDA) se extraen, limpian y tratan datos abiertos de los años 2019 y 2020 con un volumen de 2 millones de registros de consumo y facturación de energía eléctrica domiciliaria, en la fase de transformación se focalizó la ciudad de Bogotá y se empleó un modelo de aprendizaje automático predictivo el cual arrojó indicadores de consumo y falta de pago de los suscriptores o clientes del servicio.

5. METODOLOGIA

El presente trabajo se fundamenta en la metodología CRISP DM (Cross-Industry Standard Process for Data Mining) que permite orientar proyectos de minería de datos. Fue propuesta en 1999 como metodología ágil que proporciona un enfoque estructurado para la planificación y desarrollo de proyectos de análisis de datos que se caracteriza por ser robusta práctica, fácil de entender, y flexible. Esta metodología consta de etapas bien definidas y con una secuencia lógica; sin embargo, en muchas ocasiones es necesario adelantar o retroceder o repetir las etapas, dependiendo del problema y los obstáculos con los que se enfrente, cada una de sus etapas se relaciona en el ciclo de vida y se pueden visualizar en la Figura 1. (CRISP-DM, 1999).

Figura 1. Ciclo de vida de la ciencia de datos Metodología CRISP DM.



Nota: La figura representa el ciclo de vida y los procesos empleados en el análisis de los datos.
Fuente: (G., 2020).

De forma resumida la gráfica anterior detalla la teoría existente sobre aspectos metodológicos en las siguientes etapas:

Entendimiento del negocio: Durante esta etapa se determinan los objetivos de la analítica de datos y se produce el plan de proyecto.

Entendimiento de los datos: Durante esta etapa se recopila y acopian los datos de la fuente inicial, se realiza una descripción de los datos, se realiza una pre - exploración los datos y se verifica la calidad de estos.

Preparación de los datos: Se realiza una preselección de los datos y se aplican los procedimientos preliminares de limpieza de los datos.

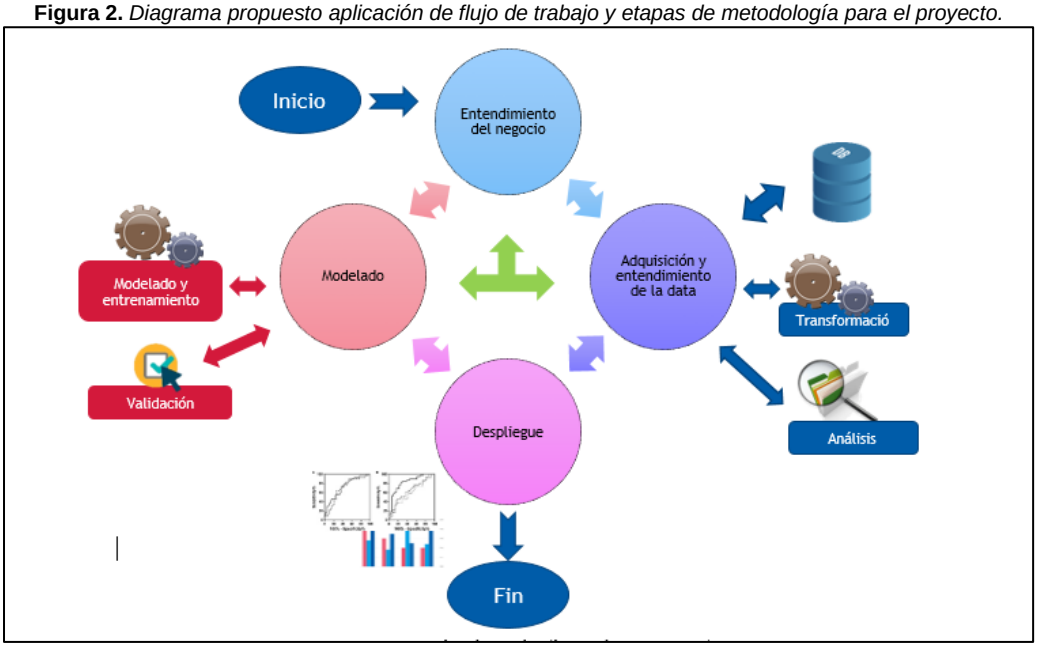
Modelamiento: Esta etapa es la más importante y tiene más relevancia de la metodología CRISP DM, En esta fase se selecciona la o las técnicas de modelado, se realiza una caracterización de las técnicas de aprendizaje automático. En esta etapa se genera un diseño de prueba del modelo y se evalúa el mismo.

Evaluación: La evaluación del modelo debe ser efectuada con el usuario, donde se resolverá si los resultados cumplen con los criterios de éxito; proceso de revisión; determinación de los próximos pasos, en caso de que sea exitoso o deba ser necesaria una nueva iteración.

Despliegue: En esta etapa se realiza la entrega del producto final a los interesados, esta puede consistir de planes, informes, modelos, o productos fundamentados en el uso y tratamiento de los datos.

De los componentes de la anterior metodología y soportado en el ciclo de los datos, se utilizará los fundamentos de CRISP DM con modificaciones adaptadas al presente proyecto acorde a como se ve en la siguiente figura. (Ver figura 2).

Figura 2. *Diagrama propuesto aplicación de flujo de trabajo y etapas de metodología para el proyecto.*



Nota: En esta figura se relaciona el flujo de trabajo que se utilizó y las etapas empleadas de la metodología CRISP DM.
Fuente: Elaboración propia, 2021 tomado de (Draw.io, 2021).

De acuerdo al flujo de trabajo utilizado y anteriormente descrito en la figura 2, a continuación, se detalla cómo se abordó cada una de las etapas.

Etapa 1 Aplicación del Entendimiento del negocio

Descripción general del contexto de uso de la data y los retos presentes en el ámbito de aplicación y uso, enfatizando en el beneficio del estudio para el negocio.

Etapa 2 Adquisición y entendimiento de los datos

En este punto se realiza un estudio inicial de los datos y sus variables con el fin de establecer e identificar los datos relevantes. Finalizado este conocimiento inicial, se realizará un perfilamiento de los datos y se aplicará tratamiento de calidad de la data puliéndola para su etapa de análisis.

Etapa 3 Adquisición de los datos de la fuente

En esta etapa se recibe los datos de su fuente en formato nativo sin tratamiento alguno.

a. Almacenamiento

Los datos crudos se almacenan en una base de datos NoSQL On-Premise para uso exclusivo en batch.

b. Estudio de entendimiento de los datos

Se entiende la naturaleza de los datos adquiridos (si son estructurados, semiestructurados, o no estructurados) con uso de Python y las librerías de NumPy y Pandas se realizará una exploración inicial de los datos, en esta etapa se entenderá la razón de cada columna y se estudiará el tipo de variable almacenada.

c. Limpieza y transformación de los datos

Con el conocimiento obtenido en la fase anterior se eliminará las incongruencias de la data, se unificará el nombre de variables y columnas con el fin de facilitar la lectura de la data, se unificará los tipos de datos (en caso de columnas con múltiples datos). Finalizada la limpieza de los datos se almacenará en una base de datos.

d. Análisis de los datos

Análisis inicial de la relación entre variables, y el comportamiento en el tiempo contenido en el set de datos.

Etapa 4 Modelado:

En esta etapa la data ya ha alcanzado un estado de preparación, y se procederá a aplicar técnicas de machine learning y se generará conocimiento a partir de la data.

a. **Modelado y entrenamiento de los datos con algoritmos de machine learning**

Siguiendo el principio de Pareto se destinará un 80% de la Data Gold para el entrenamiento de un modelo supervisado predictivo para detectar posibles clientes morosos en el servicio de energía utilizando como fuente las variables referentes al ***estrato socioeconómico***, ***familiar*** de ***vivienda del suscriptor*** y de ***consumo***.

b. **Evaluación del modelo.**

Con un 20% de la data restante se evalúa el modelo para medir su efectividad, la precisión, y cobertura.

Etapa 5 Despliegue:

Entrega del conocimiento obtenido de las etapas anteriores, es la finalización del presente estudio y el despliegue de los entregables, tales como:

a. Gráficos del conocimiento obtenido.

b. Resultados obtenidos en los modelos de machine learning

c. Conclusiones.

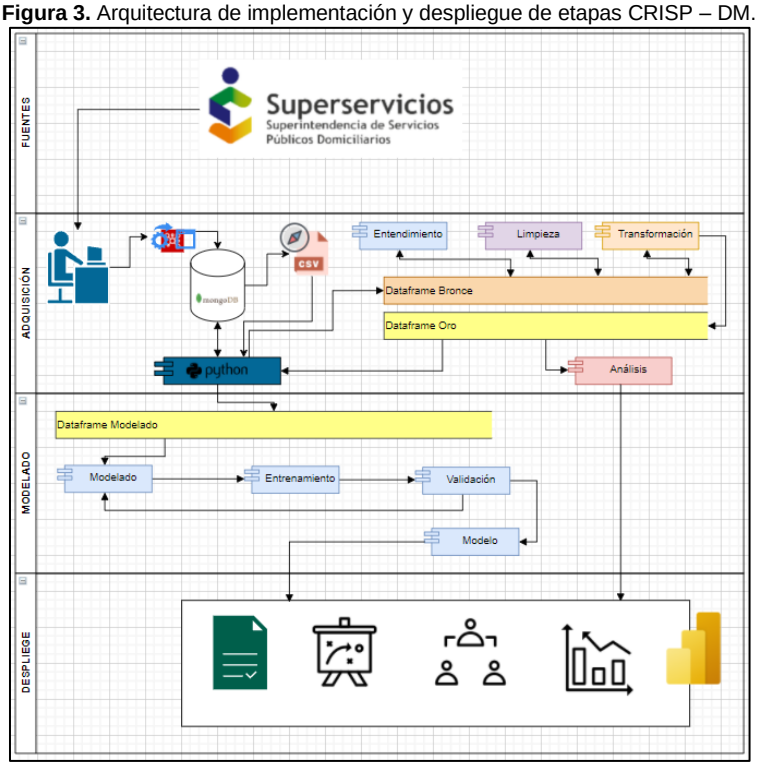
6. DESARROLLO DE LA PROPUESTA

6.1. ETAPA 1 Entendimiento del negocio

De acuerdo con lo descrito en la primera etapa de la metodología que vamos a emplear se inician y realizan las actividades relacionadas con el análisis y planeación de recursos físicos para el desarrollo del proyecto.

Como parte de la etapa 1 “Entendimiento del negocio” y tomando como base de referencia la metodología de CRISP DM se diseña un diagrama de arquitectura de implementación en la cual se contemplan las fases de fuentes, adquisición, modelado y despliegue tal como se visualiza en la Figura 3.

Figura 3. Arquitectura de implementación y despliegue de etapas CRISP – DM.



Nota: En esta figura se diseña la arquitectura de implementación y despliegue de etapas CRISP – DM utilizadas.
Fuente: Elaboración propia tomado de (Draw.io, 2021)

Especificaciones técnicas mínimas de un equipo de cómputo para implementar el proyecto

A continuación, se relacionan las características o especificaciones técnicas mínimas del equipo o máquina local donde se debe realizar la implementación del proyecto.

Tabla 1. Especificaciones técnicas mínimas del equipo de cómputo donde se realiza la implementación del proyecto.

Característica	Valor
Sistema operativo	Windows 10 Profesional
Memoria	12 GB DDR3-2666
Disco duro	500 GB
Procesador	Intel Core I3 Processor 2.20 GHz a 2.21GHz

Nota: En esta tabla se detallan las especificaciones técnicas mínimas del equipo de cómputo donde se debe realizar la implementación del proyecto.
Fuente: Elaboración propia tomado de (SW MS Word, 2021)

Como descripción del contexto de los datos se puede identificar que los datos provienen de la Dirección Técnica de Energía y Gas a través de la normativa Resolución 20102400008055 se identificó que los datos recopilados constan de 46 campos y la información es periódica con una frecuencia mensual. Dentro del análisis se observa que los datos son reportados por los operadores de red o prestadores del servicio público domiciliario de energía eléctrica y que a través de la actividad de comercialización que ejercen sobre el consumo de energía eléctrica a nivel domiciliario y territorial. Se realizó una revisión preliminar de los datos en la cual se construye el diccionario de datos del servicio público de energía. (Ver anexo A) para obtener un contexto más claro de los datos y determinar el objeto de investigación del proyecto se realiza el diseño de la arquitectura del proyecto a utilizar, la cual se relaciona a continuación.

6.2. ETAPA 2 Adquisición y entendimiento de los datos:

En esta etapa se obtienen los datos desde la fuente se accede a la página web del portal de sistema único de información (SUI) al servicio de energía en la sección de datos abiertos de la superintendencia de servicios públicos (SSPD) y hallan los datos correspondientes de información comercial residencial de energía eléctrica identificada como formato 2, a través de la revisión de esta fuente se determina que el origen de los datos parte desde el año 2007 y se encuentra información con frecuencia periódica mensual por cada año y esta almacenada hasta el año 2020.

Figura 4. *Portal SUI datos abiertos información comercial domiciliaria de energía eléctrica.*

Nota: En esta figura se muestra el portal web del Sistema Único de Información de servicios públicos domiciliarios, datos abiertos de información comercial domiciliaria de energía eléctrica.
Fuente: tomada de *(Superintendencia de servicios públicos., 2021)*

Se accede al portal SUI y se descarga los datos correspondientes a los años 2019 y 2020 y se almacenan en un folder o carpeta del equipo local.

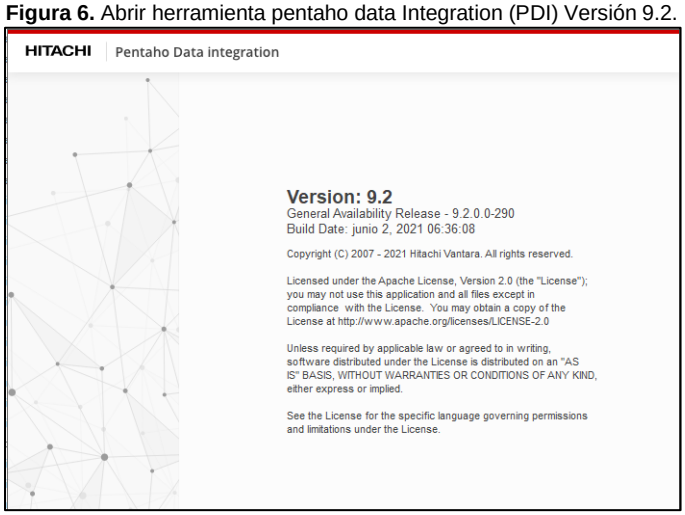
Figura 5. Carpeta local repositorio de los sets de datos años 2019 y 2020.

Nota: En la figura se ilustra el folder o carpeta cocal donde se almacena la data que se extrae de la fuente
Fuente: tomada de (SO Windows 10, 2021)

6.3. ETAPA 3 Preparación de los datos de la fuente

Se realiza el proceso de extracción y carga de datos, como herramienta de extracción, transformación y carga se utiliza Pentaho Data Integration (PDI) versión 9.2 en las Figura 6 a la Figura 17, se detalla el uso e implementación de la herramienta PDI:

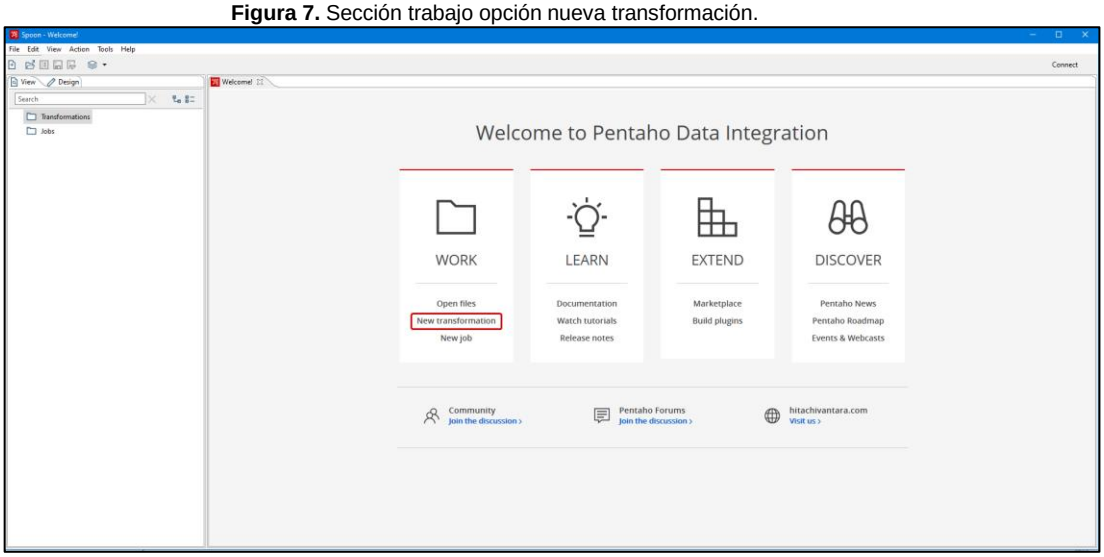
Figura 6. Abrir herramienta pentaho data Integration (PDI) Versión 9.2.



Nota: En la figura se visualiza la apertura de la herramienta pentaho data Integration (PDI) Versión 9.2.
Fuente: tomada de (SW Pentaho, 2021)

Como primer paso se crea una nueva transformación en la herramienta PDI (ver figura 7).

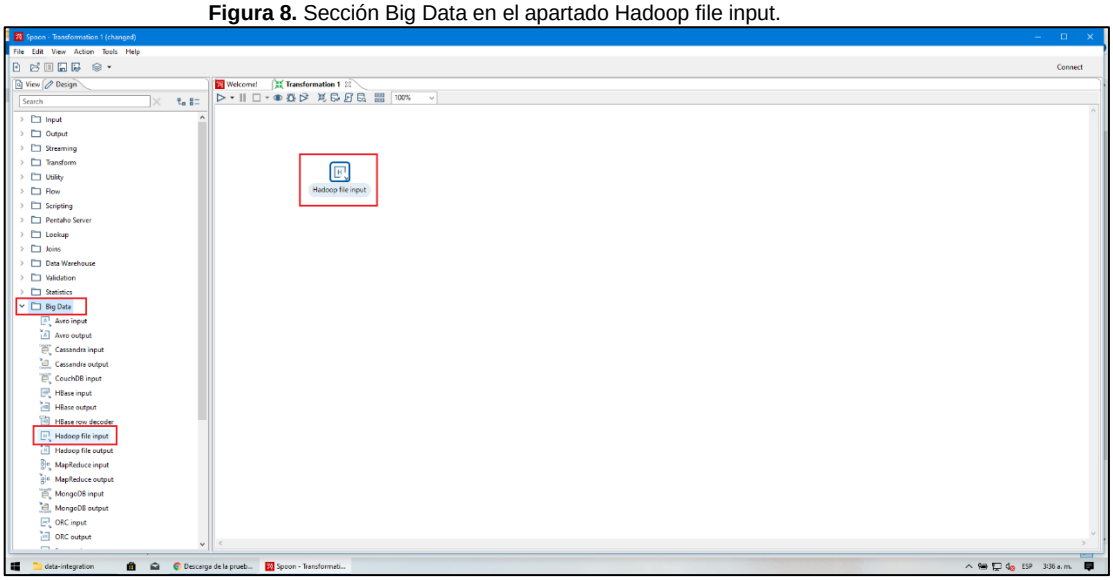
Figura 7. Sección trabajo opción nueva transformación.



Nota: En la figura se visualiza como crear una nueva transformación en la sección trabajo.
Fuente: tomada de (SW Pentaho, 2021).

Se Ubica el folder con el nombre de Big Data y la sección o el apartado Hadoop file Input se arrastra la opción al tablero de transformación (ver figura 8).

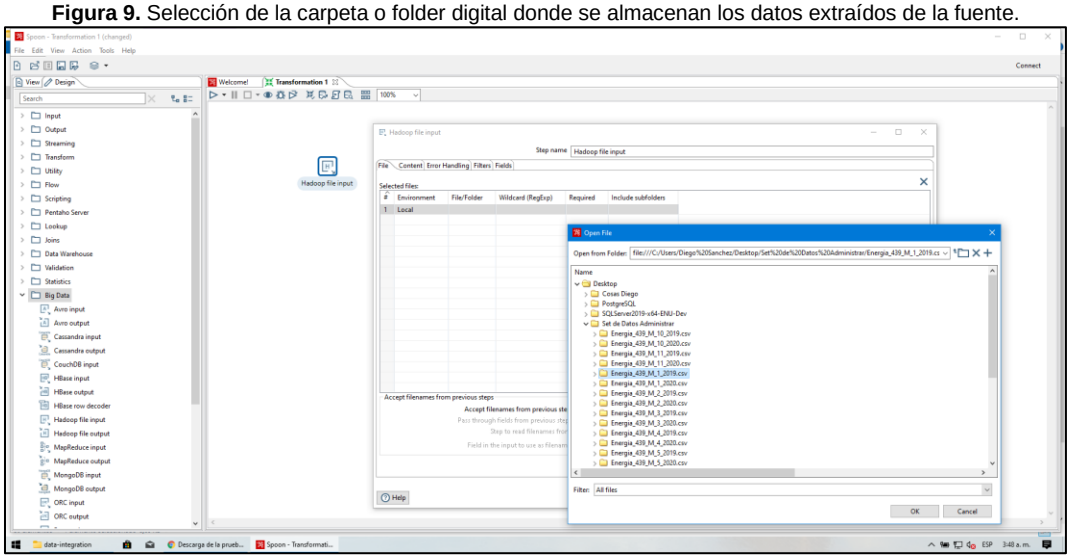
Figura 8. Sección Big Data en el apartado Hadoop file input.



Nota: En la figura se observa como seleccionar la sección Big Data y se da clic en el apartado Hadoop file input.
Fuente: tomada de (SW Pentaho, 2021)

Se edita el icono de hadoop file input y luego se ubica la carpeta o folder digital donde se almacenaron los datos originales extraídos de la fuente de datos del sistema único de información (SUI) de la superintendencia de servicios públicos (ver figura 9).

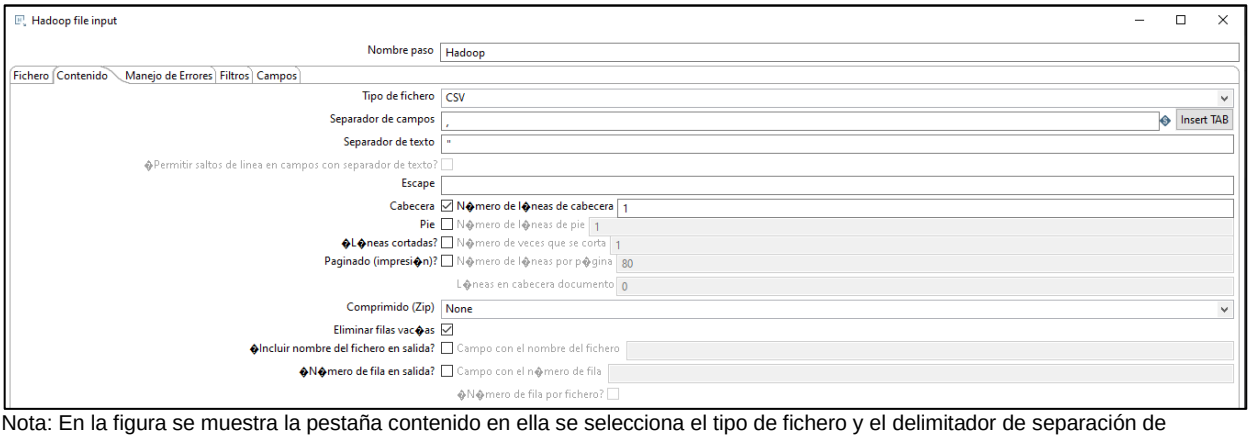
Figura 9. Selección de la carpeta o folder digital donde se almacenan los datos extraídos de la fuente.



Nota: En la figura se observa la casilla “file folder” en ella se selecciona la carpeta donde se tienen almacenados los archivos CSV.
Fuente: tomada de (SW Pentaho, 2021)

En la pestaña Contenido se parametriza el tipo de fichero CSV y el delimitador (,) con el cual separa cada columna de los sets de datos (ver figura 10).

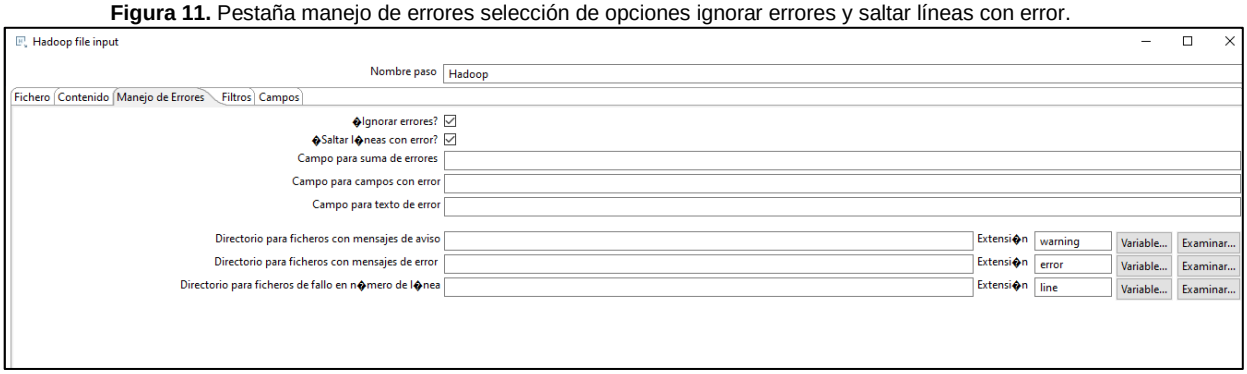
Figura 10. Pestaña Contenido selección del tipo de fichero y el delimitador de separación de columnas (,) del archivo de datos.



Nota: En la figura se muestra la pestaña contenido en ella se selecciona el tipo de fichero y el delimitador de separación de columnas (,) del archivo o fichero (.CSV)
Fuente: tomada de (SW Pentaho, 2021)

En la pestaña de manejo de errores se seleccionan las opciones casillas ignorar errores y saltar líneas con error (ver figura 11).

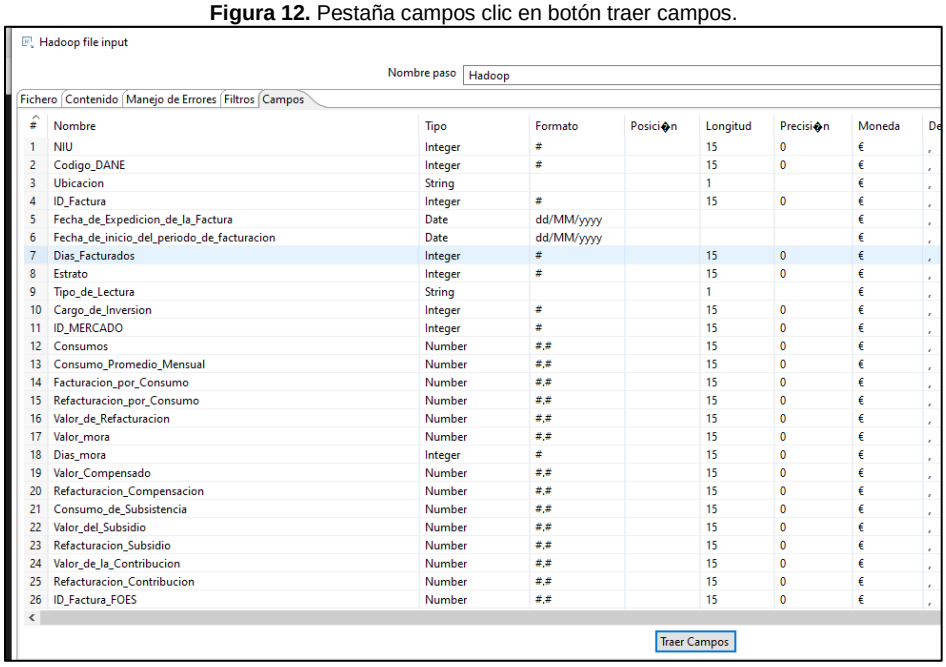
Figura 11. Pestaña manejo de errores selección de opciones ignorar errores y saltar líneas con error.



Nota: En la figura se observa la pestaña manejo de errores en ella se seleccionan las opciones ignorar errores y saltar líneas con error para omitir la data con falla.
Fuente: tomada de (SW Pentaho, 2021)

En la pestaña campos se da clic en el botón traer campos se visualizan los 46 campos que contienen los archivos fuente de datos, la herramienta PDI por defecto asocia el formato o tipo de dato que se encuentra en cada columna (ver figura 12).

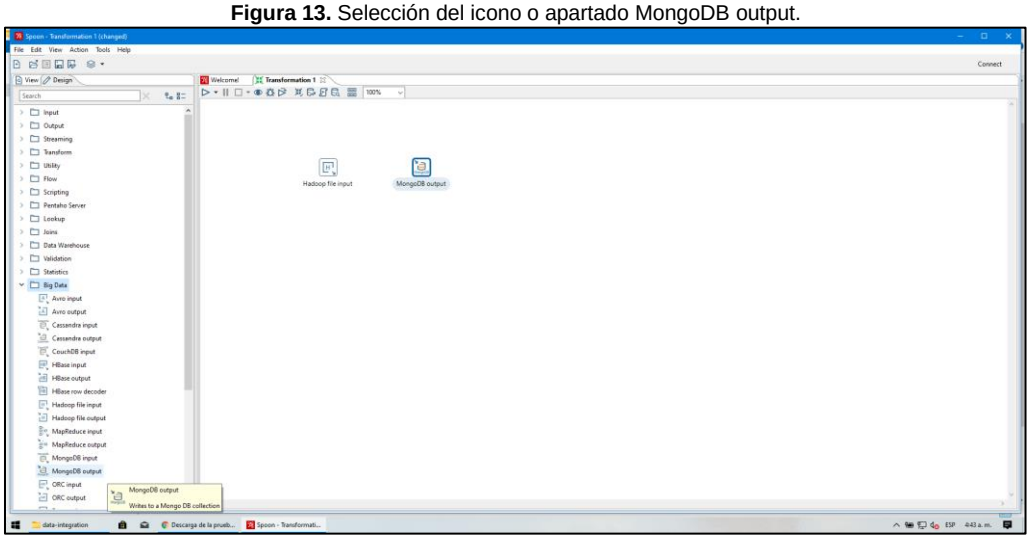
Figura 12. Pestaña campos clic en botón traer campos.



Nota: En la figura se observa la pestaña campos se da clic en el botón traer campos y visualizan los campos que contiene los archivos y su posible tipo de dato.
Fuente: tomada de (SW Pentaho, 2021)

Al lado Izquierdo en el panel de diseño se selecciona el folder con el nombre de Big Data se ubica la sección o el apartado MongoDB output luego se selecciona el icono y se da doble clic o en la opción editar (ver figura 13).

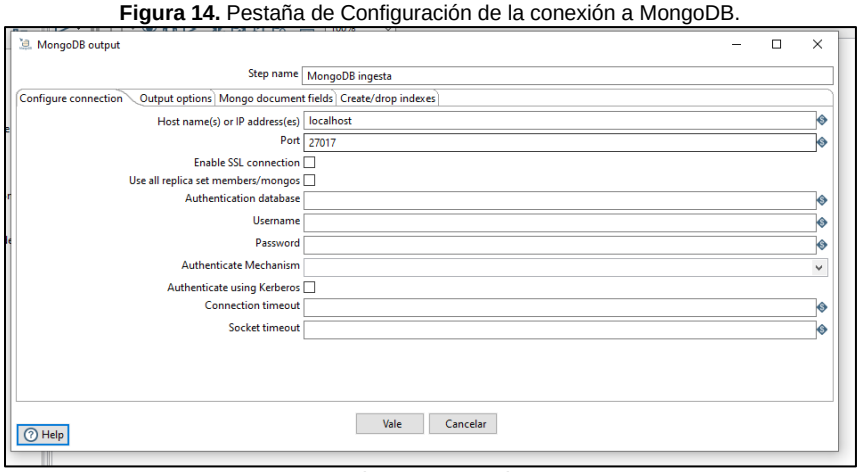
Figura 13. Selección del icono o apartado MongoDB output.



Nota: en la figura se observa la selección del icono o apartado MongoDB output.
Fuente: tomada de (SW Pentaho, 2021)

En la pestaña configuración de la conexión se parametrizan las variables de nombre de la conexión las variables hostname o dirección Ip de la maquina en la cual se va a ingesta los datos a la base de datos de mongoDB y el puerto de conexión (ver imagen 14).

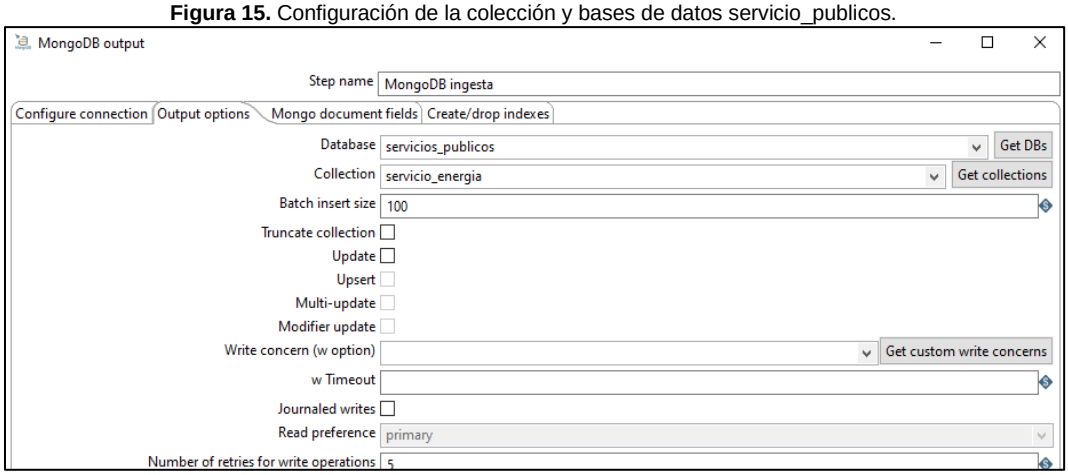
Figura 14. Pestaña de Configuración de la conexión a MongoDB.



Nota: en la figura se observa la pestaña de configuración de la conexión a mongoDB y las variables a diligenciar.
Fuente: tomada de (SW Pentaho, 2021)

En la pestaña opciones de salida se realiza la configuración de la base de datos y se le da el nombre de servicio_publicos luego se crea y nombra la colección servicio_energia, por defecto se indica en la variable tamaño de inserción en batch el valor de 100, las demás variables no se parametrizan (ver imagen 15).

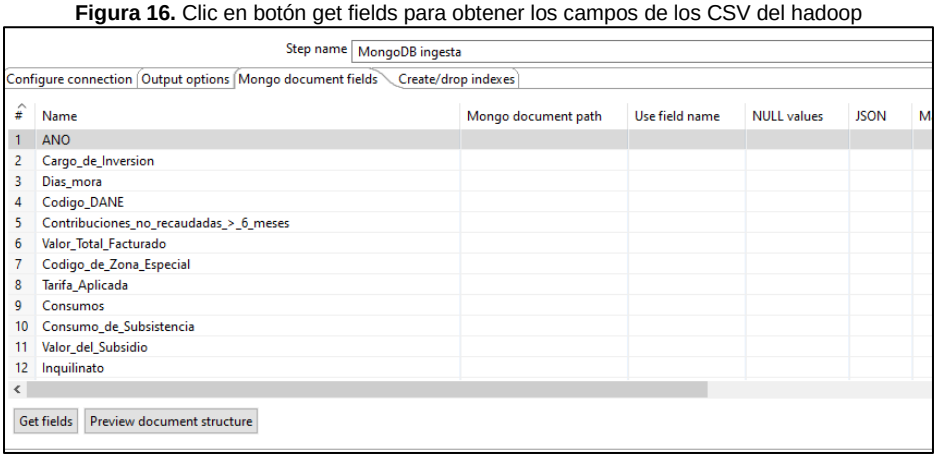
Figura 15. Configuración de la colección y bases de datos servicio_publicos.



Nota: En la figura se observa la configuración de la colección servicio_energia y bases de datos servicio_publicos.
Fuente: tomada de (SW Pentaho, 2021)

Se da clic o ubica la pestaña de campos del documento de mongo y damos clic en el botón obtener los nombres de los campos de los archivos fuente que se relacionar en la base de datos documental (ver imagen 16).

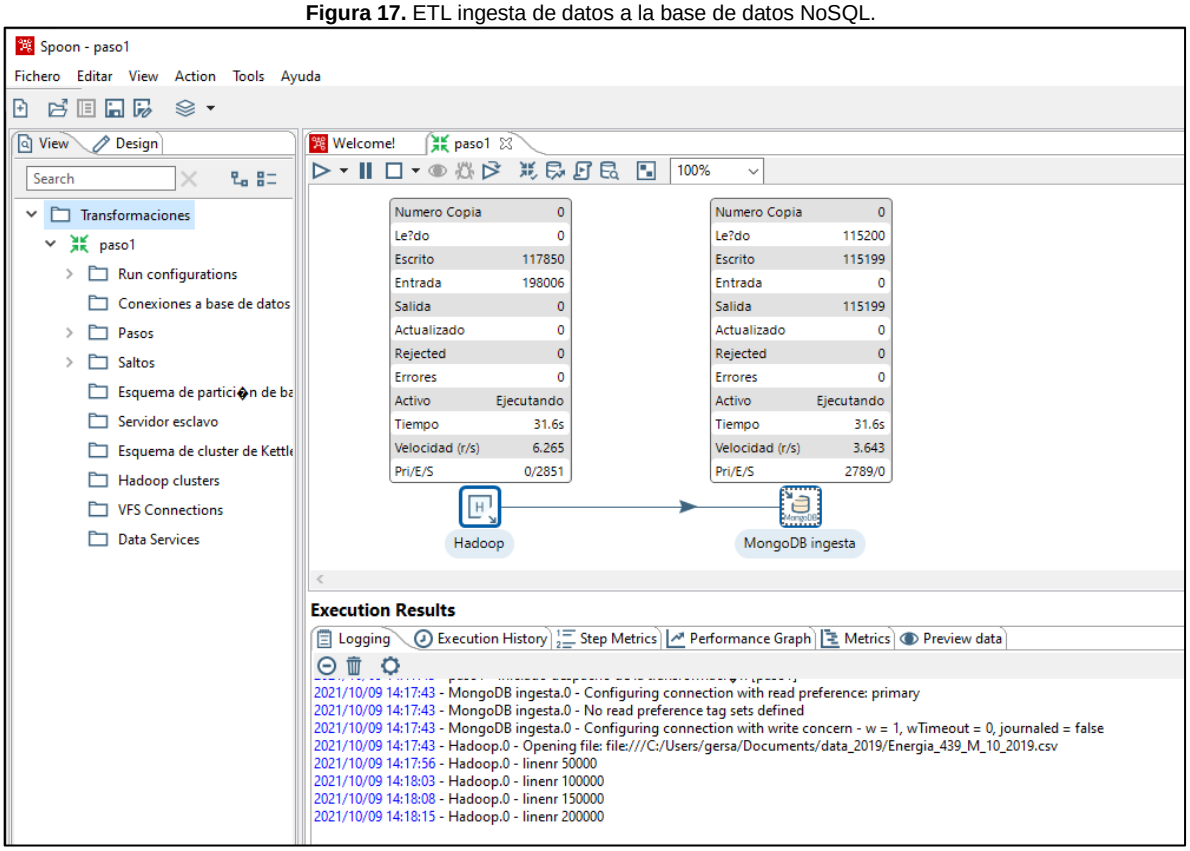
Figura 16. Clic en botón get fields para obtener los campos de los CSV del hadoop



Nota: En la figura se muestra la pestaña “mongo field documents” se da clic en el botón get fields para obtener los campos de los CSV del hadoop.
Fuente: tomada de (SW Pentaho, 2021)

Una vez parametrizadas y configuradas las 2 secciones se conectan y relacionan entre ellas, luego se da clic en el botón ejecutar para visualizar los resultados de la ejecución de los procesos de extracción y carga de datos a MongoDB (ver figura 17).

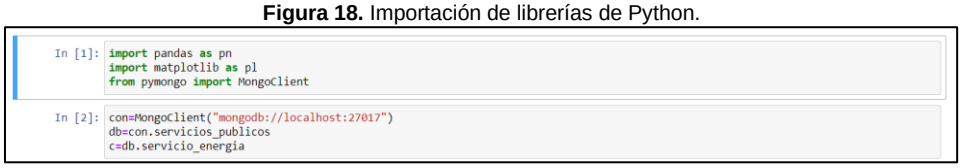
Figura 17. ETL ingesta de datos a la base de datos NoSQL.



Nota: En la figura se observa el proceso de ejecución de la extracción y carga o ingesta de datos a la base de datos NoSQL MongoDB.
Fuente: tomada de (SW Pentaho, 2021)

Se emplea el entorno de Anaconda Jupyter Notebook y utiliza el lenguaje de Python para realizar el análisis se importan las librerías de pandas, matplotlib y pymongo (ver figura 18).

Figura 18. Importación de librerías de Python.



```
In [1]: import pandas as pn
        import matplotlib as pl
        from pymongo import MongoClient

In [2]: con=MongoClient("mongodb://localhost:27017")
        db=con.servicios_publicos
        c=db.servicio_energia
```

Nota: Importación de librerías de Python para el análisis transformación y afinamiento de los datos.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

En el análisis realizado dado que se posee data de 2019 y 2020 con datos homogéneos se procede a importar mil registros de cada año para el análisis y así determinar las variables significativas para estudio.

Figura 19. Selección de datos de estudio.

```
df19=pn.DataFrame.from_records(c.find({"ANO":2019}).limit(2000000))
df20=pn.DataFrame.from_records(c.find({"ANO":2020}).limit(2000000))

In [53]: df=pn.concat([df19,df20],sort=False)

In [54]: del df19
         del df20
```

Nota: Selección de datos de estudio y concatenación para trabajar los datos mediante un data Frame en Python.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021)

Se realiza la verificación de los 46 campos de la información obtenida desde la fuente del sistema único de información (SUI) (ver imagen 20).

Figura 20. Detalle de los 46 campos de los datos analizar

```
In [55]: df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 4000000 entries, 0 to 1999999
Data columns (total 46 columns):
 #   Column                                                Dtype
---  ------                                                -----
 0   _id                                                   object
 1   ANO                                                   int64
 2   Cargo_de_Inversion                                    int64
 3   Dias_mora                                             int64
 4   Codigo_DANE                                           int64
 5   Contribuciones_no_recaudadas_>_6_meses                int64
 6   Valor_Total_Facturado                                 int64
 7   Codigo_de_Zona_Especial                               int64
 8   Tarifa_Aplicada                                       float64
 9   Consumos                                              int64
 10  Consumo_de_Subsistencia                               int64
 11  Valor_del_Subsidio                                    int64
 12  Inquilinato                                           bool
 13  Fecha_de_Expedicion_de_la_Factura                     datetime64[ns]
 14  Poblado                                               object
 15  Dias_Facturados                                       int64
 16  Fecha_de_Registro_Contable                            object
 17  Tipo_de_Factura                                       object
 18  Tipo_de_Lectura                                       object
 19  Valor_de_Refacturacion                                int64
 20  Refacturacion_Contribucion                            int64
 21  Valor_de_la_Contribucion                              int64
 22  Facturacion_por_Consumo                               int64
 23  Refacturacion_Subsidio                                int64
 24  Valor_mora                                            int64
 25  EMPRESA                                               object
 26  ID_EMPRESA                                            int64
 27  Departamento                                          object
 28  Mercado                                               object
 29  Consumo_Promedio_Mensual                              int64
 30  Municipio                                             object
 31  Ubicacion                                             object
 32  Fecha_de_inicio_del_periodo_de_facturacion            datetime64[ns]
 33  ID_MERCADO                                            int64
 34  FOES_Aplicado                                         int64
 35  Refacturacion_FOES                                    int64
 36  Estrato                                               int64
 37  Refacturacion_Compensacion                            int64
 38  PERIODO                                               int64
 39  Refacturacion_por_Consumo                             int64
 40  NIU                                                   int64
 41  ID_Factura                                            int64
 42  ID_Factura_FOES                                       int64
 43  Contribuciones_recaudadas_despues_de_conciliadas_su_no_recaudo  int64
 44  Valor_Compensado                                      int64
 45  Numero_de_Familias                                    object
dtypes: bool(1), datetime64[ns](2), float64(1), int64(31), object(11)
memory usage: 1.4+ GB
```

Nota: En la figura se detallan los 46 campos de los datos analizar y el tipo de dato de cada campo.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021)

Con el uso de estas herramientas se busca reducir y balancear la data seleccionando los campos objetivo, de acuerdo al análisis realizado se tendrán en cuenta las siguientes variables las cuales son relevantes para la aplicación dl módelo:

1. ANO: Correspondiente al año de consumo
2. Dias_mora: Cantidad de días en mora del suscriptor
3. Valor_total_facturado: Valor a pagar por concepto de servicio de energía
4. Consumos: cantidad de Kwh consumidos por el suscriptor en el periodo
5. Valor_del_Subsidio: Valor de subsidio a favor del subscriptor
6. Dias_facturados: cantidad de días facturados en el periodo
7. Tipo_factura: Tipo de factura (A anulada, I inicial o L reliquidación)
8. Tipo_de_lectura: Hace referencia a si la lectura del consumo fue (E estimada, R real, N sin medidor)
9. Consumo_promedio_mensual: Corresponde al consumo promedio mensual en Kwh (kilovatios hora)
10. Ubicación: Identifica la ubicación del inmueble (U urbano, R rural, C centro poblado)
11. Estrato: Estrato socioeconómico
12. Periodo: mes facturado

Para el análisis de exploración y entendimiento de los datos se tendrá en cuenta adicionalmente las variables:

13. Contribuciones_no_recaudadas_>_6_meses: valor de contribuciones no recaudadas después de 6 meses de facturadas
14. Valor_mora: valor en pesos de la mora

Figura 21. Selección de columnas relevantes.

```
In [6]: #Se evalua las columnas
        df_servicios.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 47723896 entries, 0 to 47723895
        Data columns (total 14 columns):
         #   Column                                Dtype
        ---  ------                                -----
         0   ANO                                   int64
         1   Consumo_Promedio_Mensual              int64
         2   Consumos                              int64
         3   Contribuciones_no_recaudadas_>_6_meses  int64
         4   Dias_Facturados                       int64
         5   Dias_mora                             int64
         6   Estrato                               int64
         7   PERIODO                               int64
         8   Tipo_de_Factura                       object
         9   Tipo_de_Lectura                       object
         10  Ubicacion                             object
         11  Valor_Total_Facturado                 int64
         12  Valor_del_Subsidio                    int64
         13  Valor_mora                            int64
        dtypes: int64(11), object(3)
        memory usage: 5.0+ GB
```

Nota: En la figura se evalúan las columnas seleccionadas y se determina su tipo de dato.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Análisis Exploratorio de Datos (EDA)

El primer análisis se aplicará sobre la variable "Consumos" que corresponde al consumo en KWh

Se responderá sí ¿existe diferencia de consumo entre los diferentes estratos?
Para ello se agrupa por estrato y se toma la media aritmética ver (Figura 22).

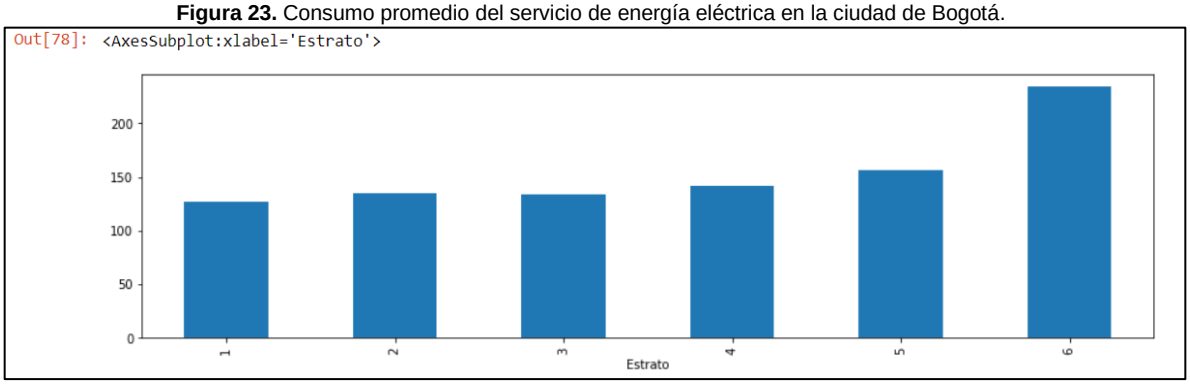
Figura 22. Script para hallar el consumo medio por estrato.

```
In [78]: print("Consumo medio por ",df_servicios.groupby(by=["Estrato"]).Consumos.mean())
         df_servicios.groupby(by=["Estrato"]).Consumos.mean().plot.bar(figsize=[15,4])

Consumo medio por  Estrato
1    126.600701
2    134.501834
3    133.346264
4    141.926444
5    155.953729
6    233.961109
Name: Consumos, dtype: float64
```

Nota: En la figura se observa el script para hallar el consumo medio por estrato.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Se evidencia que el mayor consumo promedio de energía en Bogotá se presenta en el estrato 6 como se puede observar en la (Figura 23)

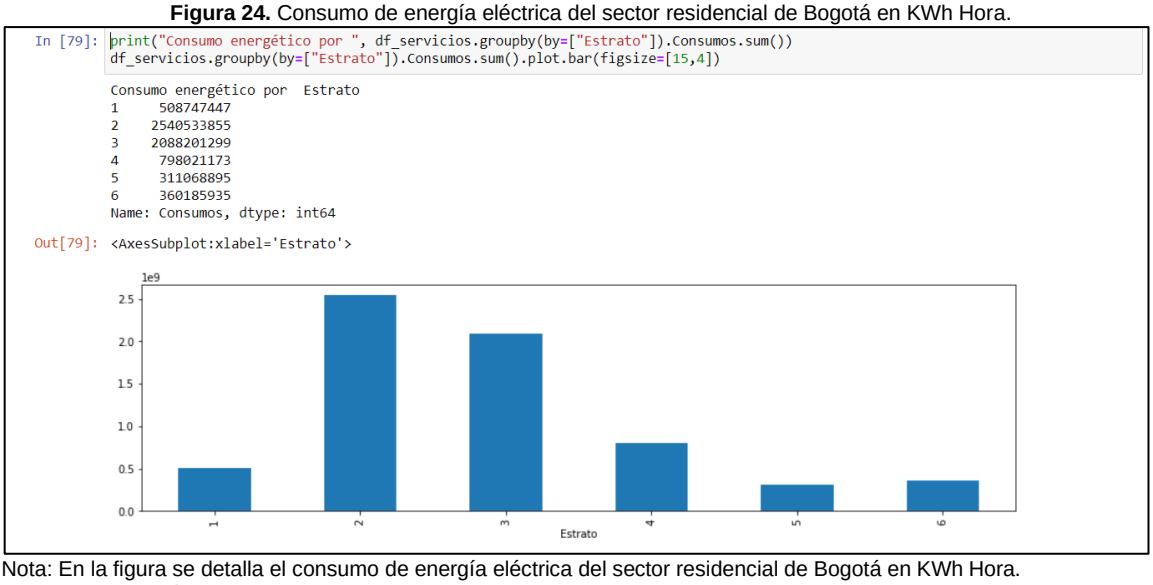
Figura 23. Consumo promedio del servicio de energía eléctrica en la ciudad de Bogotá.



Nota: En la figura se observa un diagrama de barras de mayor consumo promedio del servicio de energía eléctrica en la ciudad de Bogotá.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Sin embargo, el mayor consumo de energía eléctrica del sector residencial de Bogotá en KWh Hora lo presentan los estratos 2 y 3 como se puede observar en la (Figura 24)

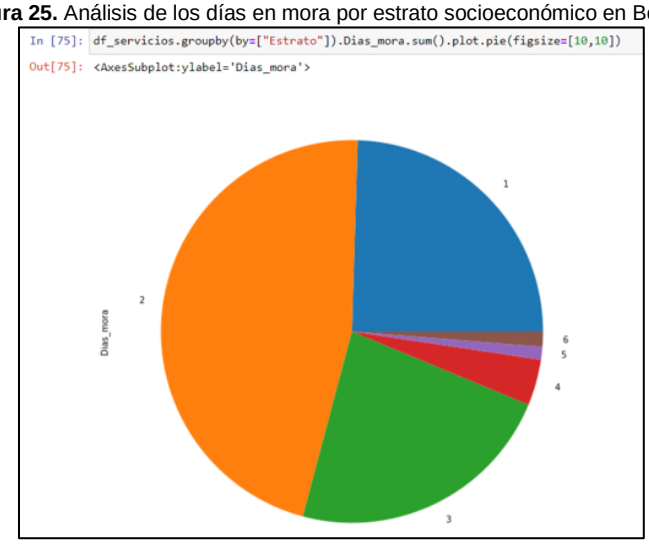
Figura 24. Consumo de energía eléctrica del sector residencial de Bogotá en KWh Hora.



Nota: En la figura se detalla el consumo de energía eléctrica del sector residencial de Bogotá en KWh Hora.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Se realiza el análisis de los días en mora por estrato socioeconómico en Bogotá de acuerdo con el set de datos como se visualiza en la (Figura 25).

Figura 25. Análisis de los días en mora por estrato socioeconómico en Bogotá.

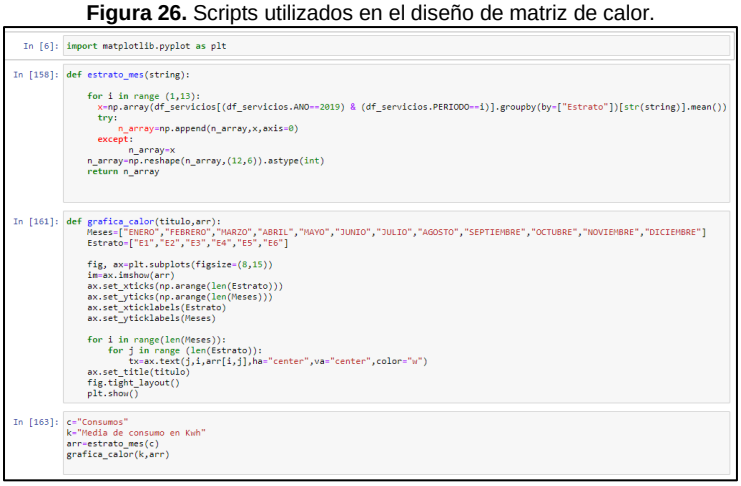


Nota: En la figura se visualiza el análisis de los días en mora por estrato socioeconómico en Bogotá.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Lo anterior evidencia que la mayor cantidad de días de mora se acumula en los estratos 1, 2 y 3 de la ciudad de Bogotá.

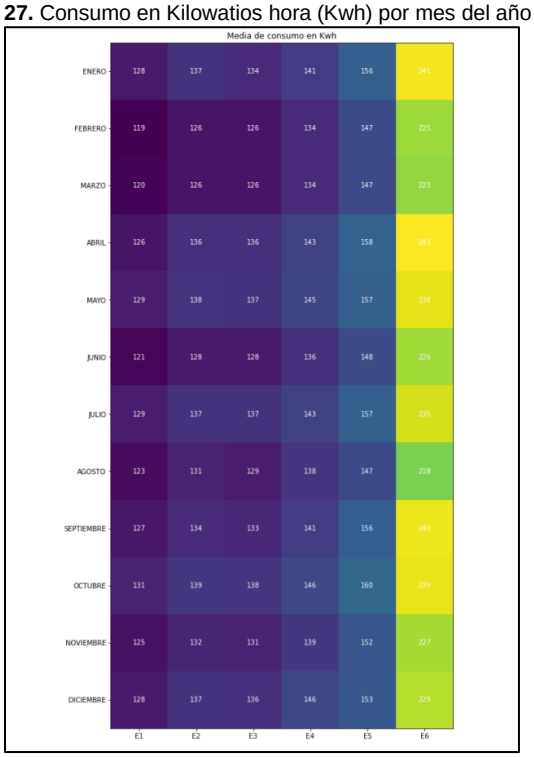
Con la ayuda de la librería matplotlib y los scripts relacionados en la (Figura 26) se diseña un gráfico de calor sobre el consumo por estrato y mes para el año 2019, se trabajá con este año debido a que la pandemia para el 2020 se considera atípica como se observa en la (Figura 27).

Figura 26. Scripts utilizados en el diseño de matriz de calor.



Nota: En la figura se observan los scripts utilizados en el diseño de matriz de calor.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Figura 27. Consumo en Kilowatios hora (Kwh) por mes del año 2019.



Nota: En la figura se observa la matriz de calor media de consumo en Kilowatios hora (Kwh) por mes del año 2019.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Con ayuda de la herramienta tecnológica Jupyter Notebook se realizó la aplicación del módelo de aprendizaje automático, para esto se utilizó un árbol de decisión y clasificación.

Como primer paso se definen las librerías a utilizar se relacionan los Scripts utilizados (ver Figura 28).

Figura 28. Librerías Python Modelo de aprendizaje automático.

```
In [2]: pip install tabulate

        Collecting tabulate
          Downloading tabulate-0.8.9-py3-none-any.whl (25 kB)
        Installing collected packages: tabulate
        Successfully installed tabulate-0.8.9
        Note: you may need to restart the kernel to use updated packages.

In [3]: #libreria de graficación
        import pylab as pl
        import matplotlib.pyplot as plt
        import seaborn as sns
        from tabulate import tabulate

        #librerias para manejo de datos
        import pandas as pn
        import numpy as np

        #librerias para análisis de datos
        from sklearn import tree
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import classification_report
```

Nota: En la figura se observan las librerías utilizadas en la aplicación del módelo de aprendizaje automático.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Se define las funciones a procesos recurrentes mediante el siguiente Script que se observa en la (Figura 29).

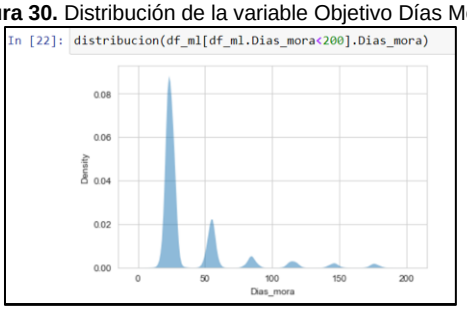
Figura 29. Script de procesos recurrentes.

```
In [5]: sns.set_style("whitegrid")
        def distribucion(a):
               sns.kdeplot(a, x="Valor de la variable",
               fill=True, common_norm=False, palette="crest",
               alpha=.5, linewidth=0,
        )
```

Nota: En la figura se detalla el script de procesos recurrentes.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Se genera la visualización de distribución de la variable objetivo días mora y se presenta en la (Figura 30).

Figura 30. Distribución de la variable Objetivo Días Mora.



Nota: En la figura se observa la distribución de la variable objetivo días mora.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Como se evidencia en las gráficas anteriores, los valores por encima de 200 tienden a ser homogéneos, por lo cual se depuran y se decide trabajar con los valores de la variable objetivo días mora que están por debajo de 200.

Mediante el siguiente Script se crea un clasificador para la variable objetivo días mora en 3 niveles como se visualiza en la (Figura 31).

Figura 31. Script clasificador de la variable objetivo en tres niveles.

```
In [25]: df_ml.moroso=0
         df_ml.loc[(df_ml.Dias_mora>0) & (df_ml.Dias_mora<31),"moroso"]=1
         df_ml.loc[(df_ml.Dias_mora>30) & (df_ml.Dias_mora<61),"moroso"]=2
         df_ml.loc[df_ml.Dias_mora>60,"moroso"]=3
```

Nota: En a figura se detalla el script clasificador para la variable objetivo en tres niveles.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Se cambia el tipo de dato de la variable objetivo días mora a tipo entero mediante el siguiente Script como se visualiza en la (Figura 32).

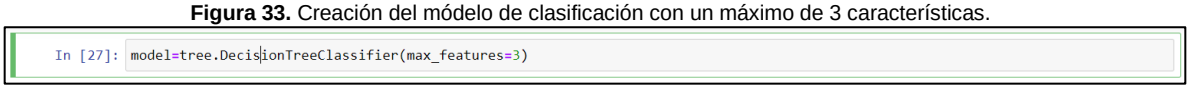
Figura 32. Script de tipo de dato a entero de la variable objetivo días mora.

```
In [26]: df_ml["moroso"]=df_ml["moroso"].astype(int)
```

Nota: En la figura se observa script de conversión de tipo de dato a entero de la variable objetivo días mora.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Mediante el siguiente Script se realizó la creación del módelo de clasificación con un máximo de tres características.

Figura 33. Creación del módelo de clasificación con un máximo de 3 características.



```
In [27]: model=tree.DecisionTreeClassifier(max_features=3)
```

Nota: Script de creación del módelo de clasificación con un máximo de 3 características.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

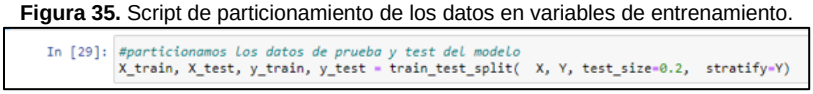
Mediante el siguiente Script se entrenó y evaluó el modelo con las siguientes variables como se observa en las figuras 34, 35 y 36.

Figura 34. Dataframe con variables seleccionadas para el modelo de aprendizaje.

```
In [28]: X=df_ml[["Consumo_Promedio_Mensual",
                  "Consumos",
                  "Contribuciones_no_recaudadas_>_6_meses",
                  "Dias_Facturados",
                  "Valor_Total_Facturado",
                  "Estrato",
                  "PERIODO",
                  "Valor_del_Subsidio",
                  "Refacturacion",
                  "Factura_Inicial",
                  "medicion_real",
                  "medicion_estimada",
                  "sin_medidor",
                  "Urbano",
                  "Rural",
                  "Poblado"]].values
         Y=df_ml["moroso"].values
```

Nota: En la figura se observa el script de selección de variables en un dataframe para la evaluación del módelo de aprendizaje.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Figura 35. Script de particionamiento de los datos en variables de entrenamiento.



```
In [29]: #particionamos los datos de prueba y test del modelo
         X_train, X_test, y_train, y_test = train_test_split(  X, Y, test_size=0.2,  stratify=Y)
```

Nota: En la figura se observa el script de particionamiento de los datos en variables de entrenamiento.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

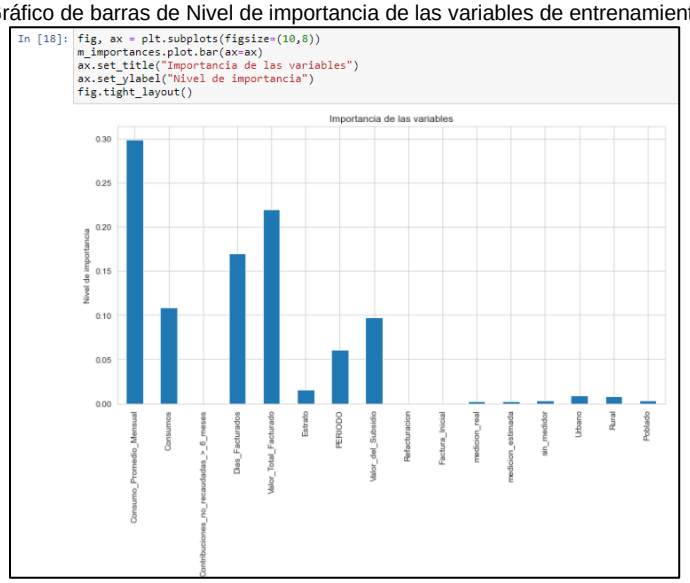
Figura 36. Script de datos de disminución de impureza de las variables relevantes.

```
In [30]: #evaluamos la importancia de variables y aplicamos el método de Disminucion media de la impureza
         nombres_variables = ["Consumo_Promedio_Mensual",
                              "Consumos",
                              "Contribuciones_no_recaudadas_>_6_meses",
                              "Dias_Facturados",
                              "Valor_Total_Facturado",
                              "Estrato",
                              "PERIODO",
                              "Valor_del_Subsidio",
                              "Refacturacion",
                              "Factura_Inicial",
                              "medicion_real",
                              "medicion_estimada",
                              "sin_medidor",
                              "Urbano",
                              "Rural",
                              "Poblado"]
         model.fit(X_train, y_train)

         importances = model.feature_importances_

         m_importances = pn.Series(importances, index=nombres_variables)
```

Nota: En la figura se ilustra el script de datos de disminución de impureza de las variables relevantes.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Mediante el siguiente Script se generó un gráfico el cual permite determinar el nivel de relevancia e importancia de las variables con las que estamos realizando el entrenamiento del módelo (ver Figura 37).

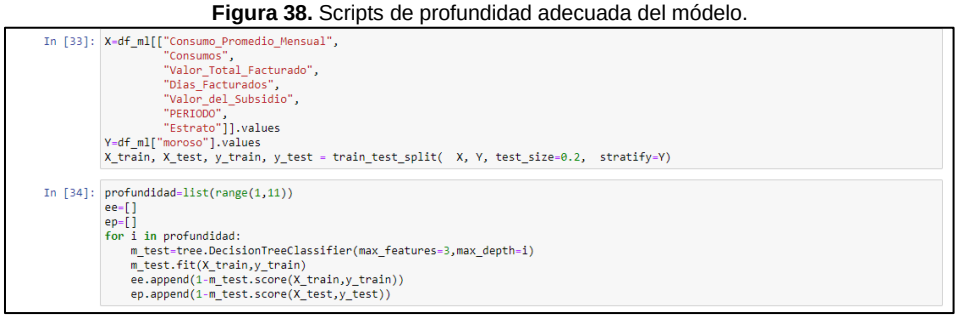
Figura 37. Gráfico de barras de Nivel de importancia de las variables de entrenamiento y pruebas.



En la figura se observa la generación de gráfico de barras de Nivel de importancia de las variables de entrenamiento y pruebas.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Mediante los siguientes Scripts se determinó la profundidad adecuada para el modelo como se visualiza en la (Figura 36).

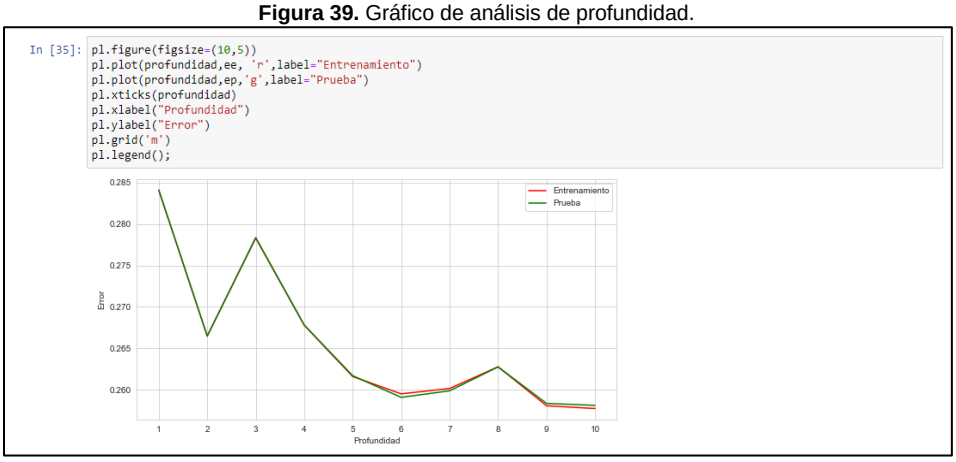
Figura 38. Scripts de profundidad adecuada del módelo.



Nota: En la figura se observa los scripts utilizados para determinar la profundidad adecuada del módelo.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Mediante el siguiente Script se generó gráfico de líneas con el cual se realiza el análisis de profundidad y se evidencia que el error mínimo está en una profundidad de 10.

Figura 39. Gráfico de análisis de profundidad.



Nota en la figura se muestra mediante el script la generación de gráfico de análisis de profundidad del módelo predictivo.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Mediante los siguientes Scripts se procede a entrenar el modelo con las variables importantes y la profundidad adecuada como se observa en la (Figura 38).

Figura 40. Primer Entrenamiento con variables de profundidad y relevancia adecuada.

```
In [38]: model_1=tree.DecisionTreeClassifier(max_features=3,max_depth=10)

In [39]: #entrenamiento del modelo para morosos nivel 1
         X=df_ml[["Consumo_Promedio_Mensual",
                  "Consumos",
                  "Valor_Total_Facturado",
                  "Dias_Facturados",
                  "Valor_del_Subsidio",
                  "PERIODO",
                  "Estrato"]].values

         Y=df_ml["moroso"].values
         X_train, X_test, y_train, y_test = train_test_split(  X, Y, test_size=0.2,  stratify=Y)
         model_1.fit(X_train, y_train)

Out[39]: DecisionTreeClassifier(max_depth=10, max_features=3)

In [40]: model_1.fit(X_train, y_train)

Out[40]: DecisionTreeClassifier(max_depth=10, max_features=3)

In [41]: y_pred=model_1.predict(X_test)

In [42]: print(classification_report(y_pred,y_test))

                precision    recall  f1-score   support

             1       0.98      0.75      0.85    300913
             2       0.25      0.63      0.36     22277
             3       0.01      0.40      0.02      1015

      accuracy                           0.74    324205
     macro avg       0.41      0.59      0.41    324205
  weighted avg       0.92      0.74      0.81    324205
```

Nota: En la figura se observa el script de entrenamiento con variables de profundidad y relevancia adecuada.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Mediante el siguiente Script se implementó una función que permite evaluar el cliente y predecir el grado de mora y cumplimiento de pagos utilizando el módelo DesicionTreeClassifier con relación a las variables consumo promedio mensual (cpm), consumos (c), valor total facturado (vtf), días facturados (df), valor del subsidio (vds), periodo (p) y estrato (E) (ver Figura 39).

Figura 41. Primer evaluación y predicción del grado de mora en los clientes.

```
In [43]: def evalua_cliente(cpm,c,vtf,df,vs,p,E):
             k=model_1.predict([[cpm,c,vtf,df,vs,p,E]])
             l=model_1.predict_proba([[cpm,c,vtf,df,vs,p,E]])
             pl.figure(figsize=(8,8))
             pl.pie([l[0,0],l[0,1],l[0,2]],labels=["Moroso tipo 1","Moroso tipo 2","Moroso tipo 3"])
             pl.show()
             print("_____")
             print("El modelo ha determinado que los datos corresponde a:")
             if k==1:
                 print("Moroso tipo 1 con un ",np.around(l[0,0]*100,1),"% de probabilidad")
             elif k==2:
                 print("Moroso tipo 2 con un ",np.around(l[0,1]*100,1),"% de probabilidad")
             else:
                 print("Moroso tipo 3 con un ",np.around(l[0,2]*100,1),"% de probabilidad")
```

Nota: En la figura se detalla el script de función que permite evaluar el cliente y predecir el grado de mora y cumplimiento de pagos utilizando el módelo DesicionTreeClassifier.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Mediante el siguiente Script se probó y evaluó el modelo DesicionTreeClassifier con un dato aleatorio y se generó y diseño un gráfico de torta la cual permite identificar el tipo de morosidad o falta de pagos por los clientes como se observa en la (Figura 40).

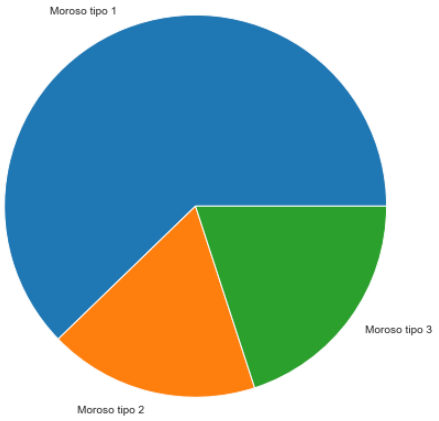
Figura 42. Gráfico de los resultados de la evaluación del primer módelo de aprendizaje automático.

```
In [44]: df_ml[["Consumo_Promedio_Mensual",
                 "Consumos",
                 "Valor_Total_Facturado",
                 "Dias_Facturados",
                 "Valor_del_Subsidio",
                 "PERIODO",
                 "Estrato","moroso"]].iloc[np.random.randint(df_ml.shape[0],size=1)]
```

Out[44]:

	Consumo_Promedio_Mensual	Consumos	Valor_Total_Facturado	Dias_Facturados	Valor_del_Subsidio	PERIODO	Estrato	moroso
7989397	2	4	1087	33	1025	12	2	3

```
In [46]: evalua_cliente(2,4,1087,33,1025,12,2)
```




```
_____
El modelo ha determinado que los datos corresponde a:
Moroso tipo 1 con un  62.2 % de probabilidad
```

Nota: En la figura se observa el script de prueba de módelo DesicionTreeClassifier con datos aleatorios el cual genera grafica de torta que de acuerdo a las tres características implementadas determina o predice el tipo de morosidad de indicando su porcentaje.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Se realizó la aplicación de cuatro modelos mediante sub sets de tipo 1 x 4 a partir de una grilla aleatoria de particionamiento como se muestra en la (Figura 43).

Figura 43. Generación de Dataframes a partir grilla aleatoria de particionamiento.

```
In [74]: df_ml
```

Out[74]:

	Consumo_Promedio_Mensual	Consumos	Valor_Total_Facturado	Dias_Facturados	Valor_del_Subsidio	PERIODO	Estrato	moroso
256	711	885	446170	33	10052	10	3	1
318	0	0	0	30	0	10	2	3
375	31	26	7033	33	6370	10	2	1
573	82	153	47151	33	32390	10	2	1
1035	86	67	29358	33	5181	10	3	2
...	...	...	...	...	...	...	...	...
47723885	1	0	0	32	0	5	2	1
47723886	62	45	11469	32	11151	5	2	1
47723889	2	1	255	32	248	5	2	2
47723894	6	3	765	32	743	5	2	2
47723895	2	3	765	32	743	5	2	3

1621025 rows × 8 columns

Nota: en la figura se observa la generación de Dataframes a partir grilla aleatoria de particionamiento.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Total, de muestras por moroso tipo 1: ",1160549/4 con este resultado se crearon 4 sub sets de datos como se muestra en la (Figura 44).

Figura 44. Creación de 4 sub sets de datos de forma aleatoria.

```
In [ ]: #se crea 4 subsets de datos de la variable tipo 1

In [76]: df_ml_1=df_ml[df_ml["moroso"]==1].sample(n=290137, random_state=1)

In [78]: df_ml_2=df_ml[df_ml["moroso"]==1].sample(n=290137, random_state=2)

In [79]: df_ml_3=df_ml[df_ml["moroso"]==1].sample(n=290137, random_state=3)

In [80]: df_ml_4=df_ml[df_ml["moroso"]==1].sample(n=290137, random_state=4)
```

Nota: En la figura se observa los scripts de creación de 4 sub sets de datos de forma aleatoria.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Se unen los sets de datos con la totalidad de datos de la variable moroso de tipo 2 y tipo 3 (Ver Figura 45).

Figura 45. Concatenación de Sets de datos con variables de tipo moroso 2 y 3.

```
In [81]: df_ml_1=pn.concat([df_ml_1,df_ml[df_ml["moroso"]==2],df_ml[df_ml["moroso"]==3]])

In [83]: df_ml_2=pn.concat([df_ml_2,df_ml[df_ml["moroso"]==2],df_ml[df_ml["moroso"]==3]])

In [84]: df_ml_3=pn.concat([df_ml_3,df_ml[df_ml["moroso"]==2],df_ml[df_ml["moroso"]==3]])

In [85]: df_ml_4=pn.concat([df_ml_4,df_ml[df_ml["moroso"]==2],df_ml[df_ml["moroso"]==3]])
```

Nota: En la figura se observa los scripts de concatenación de tipo 1 con los sets de las variables de tipo moroso 2 y 3.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

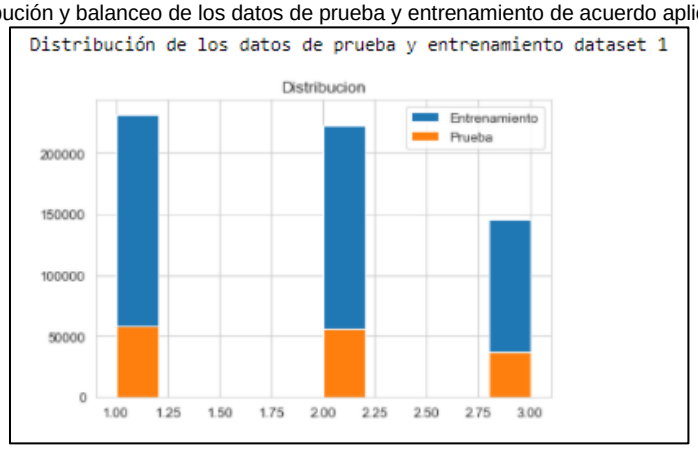
Una vez se concatenaron los Dataframes se procede a entrenar los modelos de forma individual como se observa en la (Figura 46).

Figura 46. Script de entrenamiento módelo 1.

```
In [101]: X=df_ml_1[["Consumo_Promedio_Mensual",
                     "Consumos",
                     "Valor_Total_Facturado",
                     "Dias_Facturados",
                     "Valor_del_Subsidio",
                     "PERIODO",
                     "Estrato"]].values

          Y=df_ml_1["moroso"].values
          X_train, X_test, y_train, y_test = train_test_split(  X, Y, test_size=0.2,  stratify=Y)
          print("Distribución de los datos de prueba y entrenamiento dataset 1")
          pl.hist(y_train, label="Entrenamiento")
          pl.hist(y_test, label="Prueba")
          pl.legend()
          pl.title("Distribucion")
          pl.show()

          print("_____________________________________________________________________________")
```

Nota: En la figura se observa el script de entrenamiento módelo 1.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Figura 47. Redistribución y balanceo de los datos de prueba y entrenamiento de acuerdo aplicación del módelo 1



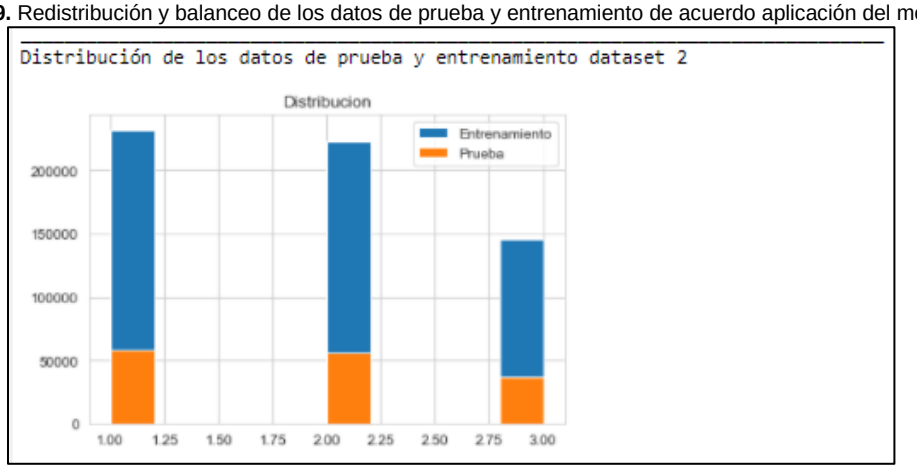
Nota: En la figura se observa la Redistribución y balanceo de los datos de prueba y entrenamiento Modelo 1.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Figura 48. Script de entrenamiento módelo 2.

```
X_2=df_ml_2[["Consumo_Promedio_Mensual",
             "Consumos",
             "Valor_Total_Facturado",
             "Dias_Facturados",
             "Valor_del_Subsidio",
             "PERIODO",
             "Estrato"]].values

Y_2=df_ml_2["moroso"].values
X_train_2, X_test_2, y_train_2, y_test_2 = train_test_split(  X_2, Y_2, test_size=0.2,  stratify=Y_2)
print("Distribución de los datos de prueba y entrenamiento dataset 2")
pl.hist(y_train_2, label="Entrenamiento")
pl.hist(y_test_2, label="Prueba")
pl.legend()
pl.title("Distribucion")
pl.show()

print("_____________________________________________________________________________")
```

Nota: En la figura se observa el script de entrenamiento módelo 2.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Figura 49. Redistribución y balanceo de los datos de prueba y entrenamiento de acuerdo aplicación del módelo 2.



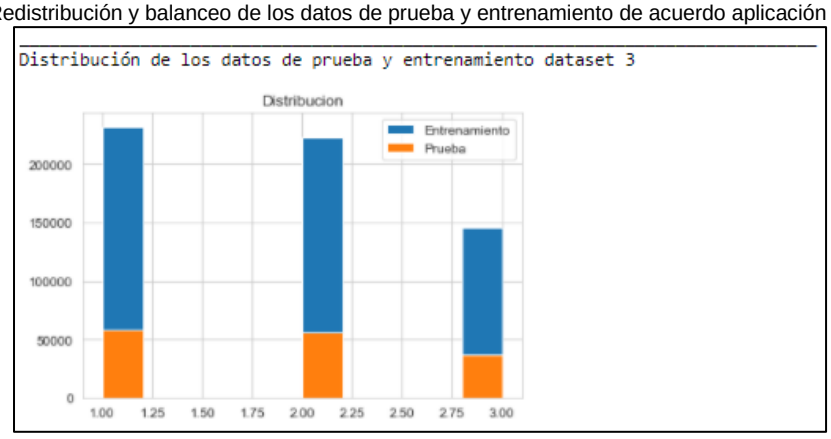
Nota: En la figura se observa la Redistribución y balanceo de los datos de prueba y entrenamiento Modelo 2.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Figura 50. Script de entrenamiento módelo 3.

```
X_3=df_ml_3[["Consumo_Promedio_Mensual",
         "Consumos",
         "Valor_Total_Facturado",
         "Dias_Facturados",
         "Valor_del_Subsidio",
         "PERIODO",
         "Estrato"]].values

Y_3=df_ml_3["moroso"].values
X_train_3, X_test_3, y_train_3, y_test_3 = train_test_split(  X_3, Y_3, test_size=0.2,  stratify=Y_3)
print("Distribución de los datos de prueba y entrenamiento dataset 3")
pl.hist(y_train_3, label="Entrenamiento")
pl.hist(y_test_3, label="Prueba")
pl.legend()
pl.title("Distribucion")
pl.show()

print("_____")
```

Nota: En la figura se observa el script de entrenamiento módelo 3.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Figura 51. Redistribución y balanceo de los datos de prueba y entrenamiento de acuerdo aplicación del módelo 3.



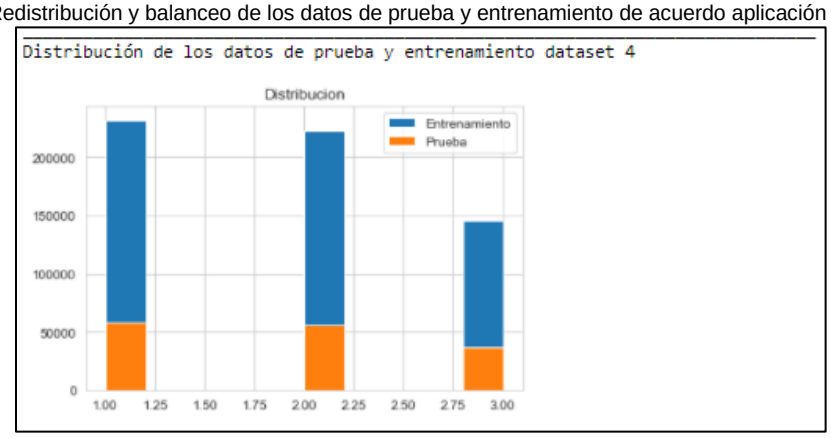
Nota: En la figura se observa la Redistribución y balanceo de los datos de prueba y entrenamiento Modelo 3.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Figura 52. Script de entrenamiento módelo 4.

```
X_4=df_ml_4[["Consumo_Promedio_Mensual",
             "Consumos",
             "Valor_Total_Facturado",
             "Dias_Facturados",
             "Valor_del_Subsidio",
             "PERIODO",
             "Estrato"]].values

Y_4=df_ml_4["moroso"].values
X_train_4, X_test_4, y_train_4, y_test_4 = train_test_split(  X_4, Y_4, test_size=0.2,  stratify=Y_4)
print("Distribución de los datos de prueba y entrenamiento dataset 4")
pl.hist(y_train_4, label="Entrenamiento")
pl.hist(y_test_4, label="Prueba")
pl.legend()
pl.title("Distribucion")
pl.show()
```

Nota: En la figura se observa el script de entrenamiento módelo 4.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Figura 53. Redistribución y balanceo de los datos de prueba y entrenamiento de acuerdo aplicación del módelo 4.



Nota: En la figura se observa la Redistribución y balanceo de los datos de prueba y entrenamiento Modelo 4.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Figura 54. Scripts de aplicación de módelo de árbol de decisión y clasificación a los modelos 1, 2, 3 y 4.

```
In [93]: model_1=tree.DecisionTreeClassifier(max_features=7,max_depth=7)
         model_2=tree.DecisionTreeClassifier(max_features=7,max_depth=7)
         model_3=tree.DecisionTreeClassifier(max_features=7,max_depth=7)
         model_4=tree.DecisionTreeClassifier(max_features=7,max_depth=7)

In [94]: model_1.fit(X_train,y_train)
         model_2.fit(X_train_2,y_train_2)
         model_3.fit(X_train_3,y_train_3)
         model_4.fit(X_train_4,y_train_4)

Out[94]: DecisionTreeClassifier(max_depth=7, max_features=7)

In [96]: y_pred_1=model_1.predict(X_test)
         y_pred_2=model_1.predict(X_test_2)
         y_pred_3=model_1.predict(X_test_3)
         y_pred_4=model_1.predict(X_test_4)

In [98]: print("Desempeño del modelo 1\n",classification_report(y_pred_1,y_test))
         print("Desempeño del modelo 2\n",classification_report(y_pred_2,y_test_2))
         print("Desempeño del modelo 3\n",classification_report(y_pred_3,y_test_3))
         print("Desempeño del modelo 4\n",classification_report(y_pred_4,y_test_4))
```

Nota: En la figura se observan los scripts de entrenamiento y aplicación de árbol de decisión a los modelos 1, 2, 3 y 4.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Figura 55. Resultados de la aplicación de variables de entrenamiento y aplicación de módelo del árbol de decisión.

```
Desempeño del modelo 1
              precision    recall  f1-score   support

           1       0.77      0.55      0.64     81404
           2       0.42      0.63      0.51     36994
           3       0.42      0.48      0.44     31725

    accuracy                           0.55    150123
   macro avg       0.53      0.55      0.53    150123
weighted avg       0.61      0.55      0.56    150123

Desempeño del modelo 2
              precision    recall  f1-score   support

           1       0.77      0.55      0.64     81665
           2       0.42      0.64      0.50     36784
           3       0.42      0.48      0.44     31674

    accuracy                           0.55    150123
   macro avg       0.53      0.55      0.53    150123
weighted avg       0.61      0.55      0.56    150123

Desempeño del modelo 3
              precision    recall  f1-score   support

           1       0.76      0.54      0.63     81277
           2       0.42      0.63      0.50     36895
           3       0.42      0.47      0.44     31951

    accuracy                           0.55    150123
   macro avg       0.53      0.55      0.53    150123
weighted avg       0.60      0.55      0.56    150123

Desempeño del modelo 4
              precision    recall  f1-score   support

           1       0.76      0.55      0.64     81367
           2       0.42      0.64      0.51     36920
           3       0.42      0.48      0.45     31836

    accuracy                           0.55    150123
   macro avg       0.54      0.55      0.53    150123
weighted avg       0.61      0.55      0.56    150123
```

Nota: En la figura se muestran los resultados de la aplicación de variables de entrenamiento y aplicación de módelo del árbol de decisión.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Figura 56. Script de evaluación y predicción de los 4 modelos en relación a los 3 tipos de clientes morosos definidos.

```
In [102]: def evalua_cliente_2(cpm,c,vtf,df,vs,p,E):
              k=model_1.predict([[cpm,c,vtf,df,vs,p,E]])
              l=model_1.predict_proba([[cpm,c,vtf,df,vs,p,E]])
              pl.figure(figsize=(8,8))
              pl.pie([l[0,0],l[0,1],l[0,2]],labels=["Moroso tipo 1","Moroso tipo 2","Moroso tipo 3"])
              pl.show()
              print("_____")
              print("El modelo 1 ha determinado que los datos corresponde a:")
              if k==1:
                  print("Moroso tipo 1 con un ",np.around(l[0,0]*100,1),"% de probabilidad")
              elif k==2:
                  print("Moroso tipo 2 con un ",np.around(l[0,1]*100,1),"% de probabilidad")
              else:
                  print("Moroso tipo 3 con un ",np.around(l[0,2]*100,1),"% de probabilidad")

              k=model_2.predict([[cpm,c,vtf,df,vs,p,E]])
              l=model_2.predict_proba([[cpm,c,vtf,df,vs,p,E]])
              pl.figure(figsize=(8,8))
              pl.pie([l[0,0],l[0,1],l[0,2]],labels=["Moroso tipo 1","Moroso tipo 2","Moroso tipo 3"])
              pl.show()
              print("_____")
              print("El modelo 2 ha determinado que los datos corresponde a:")
              if k==1:
                  print("Moroso tipo 1 con un ",np.around(l[0,0]*100,1),"% de probabilidad")
              elif k==2:
                  print("Moroso tipo 2 con un ",np.around(l[0,1]*100,1),"% de probabilidad")
              else:
                  print("Moroso tipo 3 con un ",np.around(l[0,2]*100,1),"% de probabilidad")

              k=model_3.predict([[cpm,c,vtf,df,vs,p,E]])
              l=model_3.predict_proba([[cpm,c,vtf,df,vs,p,E]])
              pl.figure(figsize=(8,8))
              pl.pie([l[0,0],l[0,1],l[0,2]],labels=["Moroso tipo 1","Moroso tipo 2","Moroso tipo 3"])
              pl.show()
              print("_____")
              print("El modelo 3 ha determinado que los datos corresponde a:")
              if k==1:
                  print("Moroso tipo 1 con un ",np.around(l[0,0]*100,1),"% de probabilidad")
              elif k==2:
                  print("Moroso tipo 2 con un ",np.around(l[0,1]*100,1),"% de probabilidad")
              else:
                  print("Moroso tipo 3 con un ",np.around(l[0,2]*100,1),"% de probabilidad")

              k=model_4.predict([[cpm,c,vtf,df,vs,p,E]])
              l=model_4.predict_proba([[cpm,c,vtf,df,vs,p,E]])
              pl.figure(figsize=(8,8))
              pl.pie([l[0,0],l[0,1],l[0,2]],labels=["Moroso tipo 1","Moroso tipo 2","Moroso tipo 3"])
              pl.show()
              print("_____")
              print("El modelo 4 ha determinado que los datos corresponde a:")
              if k==1:
                  print("Moroso tipo 1 con un ",np.around(l[0,0]*100,1),"% de probabilidad")
              elif k==2:
                  print("Moroso tipo 2 con un ",np.around(l[0,1]*100,1),"% de probabilidad")
              else:
                  print("Moroso tipo 3 con un ",np.around(l[0,2]*100,1),"% de probabilidad")
```

Nota: En la figura se observa el script de evaluación y predicción de los 4 modelos en relación a los 3 tipos de clientes morosos definidos.
Fuente: Elaboración propia tomado de (SW Jupyter Notebook Python, 2021).

Se realiza mediante script la aplicación de modelo de evaluación aleatoria de Moroso de Tipo 1 el cual arrojo los siguientes resultados.

El modelo 1 ha determinado que los datos corresponden a: Moroso tipo 1 con un 50.2 % de probabilidad.
El modelo 2 ha determinado que los datos corresponden a: Moroso tipo 1 con un 50.2 % de probabilidad.
El modelo 3 ha determinado que los datos corresponden a: Moroso tipo 1 con un 50.3 % de probabilidad.
El modelo 4 ha determinado que los datos corresponden a: Moroso tipo 1 con un 50.3 % de probabilidad.

Se realiza mediante script la aplicación de modelo de evaluación aleatoria de Moroso de Tipo 3 el cual arrojo los siguientes resultados.

El modelo 1 ha determinado que los datos corresponden a: Moroso tipo 3 con un 42.5 % de probabilidad.
El modelo 2 ha determinado que los datos corresponden a: Moroso tipo 3 con un 41.7 % de probabilidad.
El modelo 3 ha determinado que los datos corresponden a: Moroso tipo 3 con un 42.7 % de probabilidad.
El modelo 4 ha determinado que los datos corresponden a: Moroso tipo 3 con un 42.1 % de probabilidad

7. CONCLUSIONES

Con los datos obtenidos de consumo y facturación de energía eléctrica y las variables, no fue posible determinar si un subscriptor o cliente puede ser moroso, de acuerdo al análisis realizado se entrenaron 4 modelos de los subscriptores en mora y de acuerdo a lo reflejado en los datos de consumo y facturación de energía de los años 2019 y 2020, se obtiene una precisión media tendiendo a baja, para lograr un nivel de aseguramiento más alto se debe continuar con el análisis y estudio de ajuste de los hiper - parámetros del módelo predictivo lo cual por motivos de tiempo y conocimiento de modificación de los hiper - parámetros no se alcanza a cubrir en el desarrollo de este proyecto de análisis y predicción e identificación de clientes morosos sobre datos abiertos de consumo de energía eléctrica.

8. REFERENCIAS

ARÉVALO, C. A. (Marzo de 2021). *Índices de energía eléctrica en el sector residencial de la ciudad de Santiago de Cali*. Obtenido de https://red.uao.edu.co/bitstream/handle/10614/12952/T09729.pdf?sequence=1&isAllowed=y

Borrero, J. I. (2004). *Centro de investigaciones para el desarrollo (CID) Universidad Nacional*. Obtenido de http://www.fce.unal.edu.co/pec/198-ceditorial/catalogo/i-salud-educacion-y-bienestar/1846-equidad-en-las-tarifas-de-los-servicios-publicos.html

CRISP-DM. (1999). *The CRISP-DM process model*. Obtenido de http://www.crisp-dm.org/

Elaboración propia. (Agosto de 2021). *Definición realizada por el autor.*

G., O. (Junio de 2020). *Linked In - DATA DRIVEN & Decisiones basadas en datos, pero ¿cómo comenzar con los Datos?* Obtenido de https://es.linkedin.com/pulse/data-driven-decisiones-basadas-en-datos-pero-c%C3%B3mo-con-gac-pabst

Padilla Pineda, J. D. (Noviembre de 2020). *Pronóstico y análisis del consumo de energía eléctrica en usuarios residenciales de Barranquilla - Universidad del Norte.* Obtenido de https://manglar.uninorte.edu.co/handle/10584/9269

Peña, J. C. (Julio de 2019). *Revista Academia - Métodos de aprendizaje automático.* Obtenido de https://d1wqtxts1xzle7.cloudfront.net/61186929/Metodos_de_aprendizaje_automatico_para_el_pronostico_de_consumo_electrico_Revista_Tecsup_V13_201920191111-82807-8kb1f7-with-cover-page-v2.pdf?Expires=1634032935&Signature=Vhvn99SfIUznrIpAXpCLoA1MPdHj89Qcivwif

Superintendencia de servicios públicos. (Julio de 2021). *Superintendencia de servicios públicos domiciliarios - Glosario de terminos generales.* Obtenido de https://www.superservicios.gov.co/glosario-de-terminos-basicos-y-generales

SW Jupyter Notebook Python, A. N. (2021). Jupyter Notebook Python, Anaconda Navigator.

SW Pentaho, H. V. (11 de 2021). Software De Extracción Transformación y Carga de datos ETL.

Russell, S. and P. Norvig, Artificial Intelligence: A Modern Approach. Second ed. Upper Saddle River (N J): Prentice Hall/ Pearson Education; 2003.

Quinlan JR. Programs for Machine Learning. The Morgan Kaufmann Series in Machine Learning. San Mateo (California): Morgan Kaufmann Publisher; 1993.

Cruz-Ramírez N, Acosta-Mesa HG, Carrillo-Calvet H, Barrientos Martínez RE. Comparison of the Performance of Seven Classifiers as Effective Decision Support Tools for the Cytodiagnosis of Breast Cancer: A Case Study. Analysis and Design of Intelligent Systems using Soft Computing Techniques. Advances in soft computing; 41: 79 - 87. 2. Russell, S. and P. Norvig, Artificial Intelligence: A Modern Approach.

9. ANEXOS.

9.1. ANEXO A - DICCIONARIO DE DATOS.

DICCIONARIO DE DATOS	
Columnas	47

ID	Campo	Detalle	Formato	Descripción
1	Niu	633729974	Número	Número de Identificación del Usuario o Suscriptor
2	codigo_dane	81001000	Número	Corresponde a la codificación dada por el DANE a la división político-administrativa de Colombia
3	Ubicación	“U”	Texto	Indica si la factura reportada corresponde a un inmueble rural disperso ©, urbano (U) o centro poblado ©. Se consideran Urbanos, aquellos inmuebles localizados en la Cabecera Municipal. Centro Poblado © es un área con características urbanas.
4	Id_factura	7843885	Número	Corresponde al número de la factura o identificación de la factura asignada por el comercializado
5	fecha_de_expedicion_de_la_factura	13/03/2020	Fecha	Se refiere a la fecha de expedición de la factura.
6	fecha_de_ini_peri_factu	04/02/2020	Fecha	Se refiere a la fecha desde la cual comienza a registrarse el consumo a facturar, de acuerdo al formato
7	Días facturados	28	Número	Corresponde al número de días facturados en el periodo.
8	Estrato	2	Número	Se refiere al estrato socio económico asociado a la estructura tarifaría aplicada. El valor reportado debe corresponder a la siguiente clasificación: 1 Bajo-Bajo, 2 Bajo, 3 Medio-Bajo, 4 Medio, 5 Medio-Alto, 6 Alto
9	Tipo_de_lectura	“R”	Texto	Hace referencia al tipo de lectura y se clasifica de acuerdo a los siguientes códigos: R – Real, E – Estimada, N – No tiene medidor.
10	Cargo de inversión	100	Número	Corresponde a la fracción del cargo máximo en el nivel de tensión 1, por concepto de inversión que liquidó el Operador de Red al comercializador para cada usuario. Podrá tomar los valores de referencia: 0 - Cuando el comercializador liquida el 0% del cargo Máximo de Nivel de tensión 1 por concepto de inversión al usuario.

ID	Campo	Detalle	Formato	Descripción
				50 - Cuando el comercializador liquida el 50% del cargo Máximo de Nivel de tensión 1 por concepto de inversión al usuario. 100 - Cuando el comercializador liquida el 100% del cargo Máximo de Nivel de tensión 1 por concepto de inversión al usuario
11	Id_Mercado	159	Número	Es el código del mercado de comercialización donde se efectuó la venta que se está facturando, los cuales publica la Superintendencia en la página del SUI
12	Consumos	426	Número	Es el consumo de energía eléctrica en kWh que es facturado y reportado para el respectivo periodo.
13	Consumo_promedio_mensual	281	Número	Corresponde al consumo promedio mensual del usuario en Kwh, durante el trimestre calendario en el cual se realizó el cálculo del valor compensado al usuario peor servido
14	facturacion_por_consumo	249899	Número	Corresponde al valor en $ del consumo facturado durante el periodo reportado (no incluye subsidios ni contribuciones).
15	refacturacion_por_consumo	129	Número	Corresponde al valor en kWh, de consumos que se facturaron de más o se dejaron de facturar, durante periodos anteriores al que corresponde la factura que se reporta.
16	valor_de_refacturacion	320726	Número	Corresponde al valor en $ de los kWh consumidos que se facturaron de más o se dejaron de facturar, durante los periodos anteriores al que corresponde la factura que se reporta. Este valor será negativo si la empresa facturó de más y positivo si dejó de facturar. (No incluye Subsidios ni contribuciones).
17	valor_mora	3325	Número	Valor en pesos, facturados por concepto de todas las deudas atrasadas que se cobran en la factura hasta el periodo anterior al que se factura
18	dias_mora	159	Número	Corresponde al número de días facturados por concepto de todas las deudas atrasadas que se cobran en la factura hasta el periodo anterior al que se factura
19	Valor_compensado	0	Número	Corresponde al valor en pesos de la compensación que se realiza al usuario peor servido, para el nivel de

ID	Campo	Detalle	Formato	Descripción
				tensión correspondiente y transformador que esté conectado en el mes respectivo.
20	refacturacion_compensacion	0	Número	Corresponde al valor en $ de las compensaciones que se facturaron de más o se dejaron de facturar durante periodos anteriores al que corresponde la factura que se reporta. Este valor será negativo si la empresa facturó de más y positivo si dejó de facturar.
21	consumo_de_subsistencia	3	Número	Corresponde a la cantidad mínima de electricidad definida por la UPME, utilizada en un mes por un usuario típico para satisfacer las necesidades básicas que solamente puedan ser satisfechas mediante esta forma de energía final. Podrá tomar uno de los siguientes valores: 1 - 173 kWh-mes para alturas inferiores a 1.000 metros sobre el nivel del mar 2 - 130 kWh- mes para alturas iguales o superiores a 1.000 metros sobre el nivel del mar 3 - 184 kWh-mes para alturas inferiores a 1.000 metros sobre el nivel del mar para Barrios Subnormales 4 - 138 kWh-mes para alturas iguales o superiores a 1.000 metros sobre el nivel del mar para Barrios Subnormales 5 - Cuando se refiera a usuarios de estratos 4, 5 y 6.
22	valor_del_subsidio	46919	Número	Corresponde al valor facturado en pesos por concepto de subsidios (Fondo de Solidaridad para Subsidios y Redistribución de Ingresos - FSSRI) de acuerdo a la normatividad vigente.
23	refacturacion_subsidio	42	Número	Corresponde al valor en pesos de los subsidios que se facturaron de más o se dejaron de facturar de acuerdo con la normatividad vigente, durante periodos anteriores al que corresponde la factura que se reporta y a los ajustes por concepto de subsidios efectuados en el mismo mes de facturación.

ID	Campo	Detalle	Formato	Descripción
				Este valor será negativo si la empresa facturó de más y positivo si dejó de facturar
24	valor_de_la_contribucion	10117	Número	Corresponde al valor facturado por concepto de contribución de acuerdo a la normatividad.
25	refacturacion_contribucion	0	Número	Corresponde al valor en $ por concepto de contribución, que se facturaron demás o se dejaron de facturar durante periodos anteriores al que corresponde la factura que se reporta. Este valor será negativo si la empresa facturó demás y positivo si dejó de facturar.
26	id_factura_foes	85505950	Número	Número de la(s) factura(s) de referencia de donde se tomó el valor del consumo que sirvió de base para la liquidación del valor del Subsidio, otorgado por el Fondo de Energía Social (FOES). En el caso en que se reporte más de una factura, estas se deberán registrar separadas únicamente por un guion (-). Para el caso de un usuario que no recibe beneficio del subsidio FOES, se deberá registrar con las letras “NA”
27	foes_aplicado	6716	Número	Corresponde al valor facturado por concepto del subsidio del Fondo de Energía Social (FOES)
28	refacturacion_foes	6394	Número	Corresponde al valor en $ de los subsidios por concepto del Fondo de Energía Social (FOES), consumidos durante períodos anteriores que se facturaron demás o se dejaron de facturar durante periodos anteriores al que corresponde la factura que se reporta. Este valor será negativo si la empresa facturó demás y positivo sí dejó de facturar.
29	Valor_total_facturado	206520	Número	Corresponde al valor total facturado en pesos ($) al usuario, en el periodo reportado por concepto de consumos, incluyendo todos los conceptos liquidados tales como refacturaciones,

ID	Campo	Detalle	Formato	Descripción
				cuotas de acuerdos de pago, intereses de mora, pagos anticipados, entre otros.
30	codigo_de_zona_especial	4100	Número	Corresponde a un código de cuatro (4) dígitos asignado por el comercializador el cual debe coincidir con los códigos de los formatos 11, 12 y 13, de la presente Resolución. Se deberá reportar con valor cero "0": si el usuario no pertenece a una zona especial.
31	contribuciones_no_recaudadas_mayor_6_meses	46735.24	Numérico Decimal	Valor total en pesos ($) de las contribuciones No recaudadas después de seis meses de facturadas.
32	contri_recau_desp_de_conci_su_no_recaudo	105557.32	Numérico Decimal	Valor total en pesos ($) del Recaudo de Cartera por concepto de Contribución después de haberse conciliado su no recaudo.
33	Tarifa_aplicada	315.4099	Numérico decimal	Corresponde al valor en pesos de la tarifa aplicada a consumos inferiores o iguales al Consumo de Subsistencia (CS), cuando se trate de un usuario subsidiado; o a la tarifa con contribución cuando se trate de un usuario contribuyente; o al Costo Unitario (CU) cuando se trate de un usuario a quien se le aplique la tarifa plena.
34	fecha_de_registro_contable	13/03/2020	Fecha	Corresponde a la fecha en la cual se efectúa el registro de la factura en la contabilidad de la Empresa.
35	tipo_de_factura	“I”	Texto	Corresponde al tipo de facturación y se clasifica así: I - Inicial A - Anulada L - Reliquidación y Refacturación
36	Inquilinato	“S”	Texto	Se deberá reportar con la letra “S”, si el inmueble asociado al NIU registrado en el campo 1 En caso de no cumplir con lo anteriormente mencionado, se deberá reportar con la letra “N”.
37	no_de_familias	3	Número	Corresponde al número de familias que hacen parte del inquilinato. Este campo deberá ser mayor o igual a 3. En los casos que la columna 37 (Inquilinato) contenga la letra “N”, este campo no se deberá diligenciar.
38	acto_administrativo	ABC11	Alfa numérico	En este campo se debe consignar el acto administrativo que sustenta la asignación de estrato, aplica solo

ID	Campo	Detalle	Formato	Descripción
				para el Formato 2, información comercial residencial.
39	Id_empresa	599	Número	Corresponde al código de identificación en base de datos, de la empresa que realiza el reporte de la información al SUI.
40	Empresa	“EMPRESA DE ENERGIA DE ARAUCA”	Texto	Nombre de la empresa
41	Anio	2020	Número	año
42	Periodo	3	Número	mes
43	Departamento	“ARAUCA”	Texto	departamento
44	Municipio	“TAME”	Texto	municipio
45	centro_poblado	“TAME”	Texto	Centro poblado
46	Mercado	“ARAUCA”	Texto	mercado
47	Servicio	“Energía Eléctrica”	Texto	servicio

9.2. ANEXO B - NOTEBOOKS DE ANACONDA JUPYTER (SCRIPTS PYTHON)

Notebook_Analisis_EDA.ipynb
Notebook_transformación_data.ipynb
Notebook_notebook_DecisionTreeClassifier.ipynb