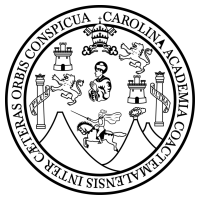


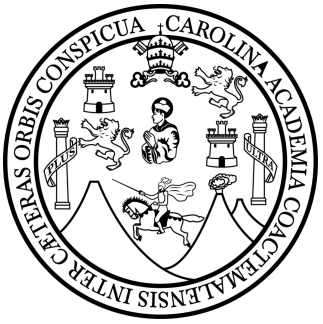
Universidad de San Carlos de Guatemala
Escuela de Ciencias Físicas y Matemáticas
Departamento de Matemática

CONSULTORÍA EN CIENCIA DE DATOS Y BUSINESS INTELLIGENCE PARA LA OPTIMIZACIÓN EMPRESARIAL: ESTRATEGIAS Y DESARROLLO DE SOLUCIONES ANALÍTICAS Y DE REPORTERÍA

Luis Alfredo Alvarado Rodríguez
Asesorado por Lic. Luis Eduardo Mack Alvizures

Guatemala, noviembre 2024

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA



ESCUELA DE CIENCIAS FÍSICAS Y MATEMÁTICAS

CONSULTORÍA EN CIENCIA DE DATOS Y BUSINESS INTELLIGENCE PARA LA OPTIMIZACIÓN EMPRESARIAL: ESTRATEGIAS Y DESARROLLO DE SOLUCIONES ANALÍTICAS Y DE REPORTERÍA

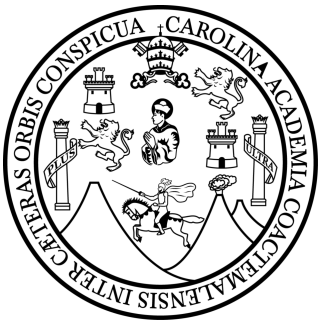
TRABAJO DE GRADUACIÓN
PRESENTADO A LA JEFATURA DEL
DEPARTAMENTO DE MATEMÁTICA
POR

LUIS ALFREDO ALVARADO RODRÍGUEZ
ASESORADA POR LIC. LUIS EDUARDO MACK ALVIZURES

AL CONFERÍRSELE EL TÍTULO DE
LICENCIADO EN MATEMÁTICA APLICADA

GUATEMALA, noviembre 2024

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
ESCUELA DE CIENCIAS FÍSICAS Y MATEMÁTICAS



CONSEJO DIRECTIVO

Director	M.Sc. Jorge Marcelo Ixquiac Cabrera
Representante Docente	Arqta. Ana Verónica Carrera Vela
Representante Docente	M.A. Pedro Peláez Reyes
Representante de Egresados	Lic. Urías Amitaí Guzmán Mérida
Representante de Estudiantes	Elvis Erique Ramírez Mérida
Representante de Estudiantes	Oscar Eduardo García Orantes
Secretario Académico	M.Sc. Freddy Estuardo Rodríguez Quezada

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

Director	M.Sc. Jorge Marcelo Ixquiac Cabrera
Examinador	Lic. José Carlos Alberto Bonilla Aldana
Examinador	MSc. Juan Diego Chi-Wen Chang Santizo
Examinador	Lic. Héctor Eduardo Pérez Figueroa
Secretario Académico	M.Sc. Freddy Estuardo Rodríguez Quezada

Ref. D.DTG. 012-2024
Guatemala, 11 de noviembre de 2024

El Director de la Escuela de Ciencias Físicas y Matemáticas de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del jefe de la Licenciatura en Matemática Aplicada, al trabajo de graduación titulado: "**CONSULTORÍA EN CIENCIA DE DATOS Y BUSINESS INTELLIGENCE PARA LA OPTIMIZACIÓN EMPRESARIAL: ESTRATEGIAS Y DESARROLLO DE SOLUCIONES ANALÍTICAS Y DE REPORTERÍA**", presentado por el estudiante universitario, Luis Alfredo Alvarado Rodríguez, autoriza la impresión del mismo.

IMPRÍMASE.

"ID Y ENSEÑAD A TODOS"

M.Sc. Jorge Marcelo Ixquiac Cabrera
Director

Universidad de San Carlos de Guatemala
Escuela de Ciencias Físicas y Matemáticas
Dirección
-USAC-ECFM-

165-2024
JMIC/Paola

AGRADECIMIENTOS

A Karla Rodríguez: Por darme la vida, guiarme con su amor y ser un ejemplo a seguir.

A Bogar Alvarado: Por ver el matemático que llevo dentro y creer en él.

A Mariajosé Chinchilla: Por ayudarme siempre y mostrarme el tipo de matemático que quiero ser.

A Jorge Paredes: Por ser el primer compañero que hice en la Universidad y por volverse un gran amigo.

A Luis Mack: Porque por él me atreví a cambiarme de carrera.

A Mynor Rodríguez: Por hacerme sentir acompañado en el proceso de cambio de carrera y terminar siendo un gran amigo.

A Anna García: Por ser una gran amiga en mi paso por ingeniería y seguirlo siendo hasta hoy en día.

Mi Macbook Air 2014 Por acompañarme desde programar una Voyage 200, Wolfram Mathematica, LaTeX, Python y hasta el entretenerme en los tiempos más difíciles de la carrera.

Mis profesores José Carlos Bonilla, Damián Ochoa, Hugo García y William Gutiérrez, por permitirme aprender de ustedes y por guiarme a lo largo de este camino académico.

DEDICATORIA

A mis padres, que siempre creyeron en mí.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
$E(X)$	Valor esperado de X
$\hat{\theta}$	Estimador del parámetro θ
$V(X)$	Varianza de X
$\mathrm{Cov}(X, Y)$	Covarianza de X y Y
$\overline{X}$	Media de X
$L(X_1, ..., X_n)$	Función de verosimilitud de $X_1, ..., X_n$
$SE(\hat{X})$	Error estándar del estimador de X
$\|x\|$	Norma de x
$\dfrac{\partial f}{\partial x}$	Derivada parcial de f respecto de x
$N(\mu, \sigma^2)$	Distribución normal con media μ y varianza σ^2
$T_{(n)}$	Distribución t de Student con n grados de libertad.

LISTA DE ABREVIATURAS

Abreviatura	Significado
SSE	Suma de cuadrados del error
ETL	*Extract, Transform and Load* (Extraer, Transformar y Cargar)
ERP	*Enterprise Resource Planning* (Planificación de Recursos Empresariales)
CRM	*Customer Relationship Management* (Gestión de Relaciones con Clientes)
KPI	*Key Performance Indicator* (Indicador Clave de Desempeño)
AI	*Artificial Intelligence* (Inteligencia Artificial)

LISTA DE REPOSITORIOS

Hipervínculo	Descripción
`https://github.com/1u1s4/3C_ML`	Jupyter Notebooks con el desarrollo de los modelos de inteligencia artificial.
`https://github.com/1u1s4/3C_ETL`	Código base de la creación del Data Warehouse.

OBJETIVOS

Objetivo General

Desarrollar e implementar una infraestructura de inteligencia de negocios mediante la construcción de un *Data Warehouse* y la aplicación de modelos avanzados de ciencia de datos, que permitan a la empresa Tres Cielos optimizar sus procesos de toma de decisiones estratégicas y mejorar su desempeño comercial.

Objetivos Específicos

1. Crear una arquitectura de *Data Warehouse* que centralice y automatice la integración de datos provenientes de diversas fuentes empresariales, garantizando la consistencia, integridad y accesibilidad de la información para su análisis.

2. Desarrollar y evaluar modelos de regresión lineal múltiple, regresión logística y clusterización K-medias para extraer *insights* valiosos sobre el comportamiento de ventas, segmentación de productos y predicción de demandas, asegurando su validez matemática y aplicabilidad en el contexto empresarial.

3. Implementar y configurar el sistema Odoo como plataforma centralizada de ERP/CRM, integrando tecnologías de inteligencia artificial como ChatGPT y DALL-E para mejorar la gestión de relaciones con clientes, optimizar procesos operativos y potenciar la generación de contenido.

INTRODUCCIÓN

En el entorno empresarial actual, la gestión y análisis de grandes volúmenes de datos son fundamentales para la toma de decisiones estratégicas. La ciencia de datos y la inteligencia de negocios permiten transformar datos en información valiosa, optimizando procesos y mejorando el rendimiento comercial.

Este Ejercicio Profesional Supervisado (EPS) tiene como objetivo desarrollar una infraestructura de inteligencia de negocios para Tres Cielos, una pastelería y panadería en expansión en Retalhuleu, Guatemala. El proyecto incluye la construcción de un *Data Warehouse* para centralizar y automatizar la integración de datos, garantizando su consistencia y accesibilidad para análisis posteriores.

Además, se implementaron modelos de regresión lineal múltiple, regresión logística y clusterización K-medias para analizar el impacto de variables clave en las ventas, predecir comportamientos de compra y segmentar productos. Paralelamente, se integró el sistema *ERP/CRM* Odoo, mejorando la gestión de procesos empresariales y relaciones con clientes. Se capacitó al personal en el uso de herramientas de inteligencia artificial como ChatGPT y DALL-E, potenciando la generación de contenido y la atención al cliente.

Este proyecto no solo se enfoca en la implementación técnica, sino también en la capacitación continua del personal, asegurando una correcta adopción de las herramientas y promoviendo una cultura de decisiones basadas en datos. Se busca proporcionar a Tres Cielos una infraestructura robusta de inteligencia de negocios, apoyada en técnicas avanzadas de análisis de datos y sistemas integrados de gestión, para optimizar su capacidad de decisión estratégica y mejorar su desempeño comercial.

1. CONTEXTUALIZACIÓN DEL NEGOCIO

1.1. Descripción de la Empresa

Tres Cielos es una pastelería y panadería ubicada en Retalhuleu, Guatemala, que ha experimentado un notable crecimiento desde su fundación. Originalmente, los fundadores de Tres Cielos se dedicaban a la venta y distribución de libros para jóvenes en diversos centros educativos del país. Sin embargo, con el inicio de la pandemia de COVID-19, este modelo de negocio se volvió insostenible debido a las restricciones impuestas y la disminución de la demanda.

Ante esta situación, los fundadores decidieron aprovechar sus conocimientos en cocina y pastelería para diversificar su oferta. Así, comenzaron a ofrecer postres, almuerzos y refacciones a sus contactos, obteniendo los permisos necesarios de las autoridades para distribuir alimentos durante el confinamiento. Al inicio, el equipo estaba compuesto por cinco miembros de la familia, quienes se distribuían en distintas áreas: cocina, reparto a domicilio y atención al cliente.

A principios de 2021, con el creciente interés por sus productos, los fundadores construyeron instalaciones adicionales en el patio de su hogar y contrataron a los primeros dos colaboradores. El 17 de junio de 2021, se inauguró oficialmente Tres Cielos como pastelería y panadería. El rápido ritmo de crecimiento llevó a la apertura de dos nuevas instalaciones en los dos años siguientes, ampliando el equipo a 23 colaboradores.

Actualmente, Tres Cielos opera en tres áreas principales:

- **Tres Cielos Central**: Encargada de la elaboración de pasteles personalizados, pasteles fríos y postres, ofreciendo servicios tanto al público general como a restaurantes.

- **Tres Cielos Panadería**: Responsable de la producción de pan de alta calidad para clientes en general y restaurantes del área, además de gestionar una ruta diaria de distribución a puntos de venta en cadenas de gasolineras.

- **Tres Cielos Café**: Punto de venta y servicio de mesa donde los clientes pueden degustar café acompañado de una variedad de productos de pastelería y panadería.

Esta estructura organizativa permite a Tres Cielos atender una amplia gama de clientes y adaptar sus operaciones a las necesidades del mercado, manteniendo un enfoque en la calidad y la satisfacción del cliente.

1.2. Requerimientos del Proyecto

Desde el inicio del proyecto, se realizaron reuniones periódicas con el personal de Tres Cielos para identificar y priorizar las necesidades clave que optimizarían la administración y el crecimiento del negocio. A continuación, se detallan los principales requerimientos abordados en este EPS:

- **Desarrollo de un *Data Warehouse***: Crear una infraestructura que centralice y automatice la integración de datos provenientes de diversas fuentes empresariales, asegurando la consistencia y accesibilidad para análisis avanzados.

- **Implementación de Modelos de Ciencia de Datos**: Desarrollar y validar modelos de regresión lineal múltiple, regresión logística y clusterización K-medias para analizar tendencias de ventas, predecir comportamientos de clientes y segmentar productos.

- **Integración de un Sistema ERP/CRM (Odoo)**: Configurar e implementar Odoo como plataforma unificada para la gestión de procesos empresariales y relaciones con clientes, mejorando la eficiencia operativa y la colaboración interna.

- **Capacitación y Soporte al Personal**: Proporcionar formación integral en el uso de las herramientas implementadas, incluyendo Odoo y tecnologías de inteligencia artificial como ChatGPT y DALL-E, para fomentar una cultura de decisiones basadas en datos.

- **Optimización de Canales de Comunicación**: Unificar las líneas telefónicas en una única línea centralizada para mejorar la gestión de pedidos y la atención al cliente, reduciendo la confusión y aumentando la eficiencia operativa.

Tabla de Actividades y sus Impactos para Tres Cielos

Tabla 1.1. Actividades Propuestas y sus Impactos Positivos para Tres Cielos

Actividad	**Descripción**	**Impactos Positivos**
Desarrollo de un *Data Warehouse*	Crear una infraestructura que centralice y automatice la integración de datos provenientes de diversas fuentes empresariales.	Permite la automatización de modelos de inteligencia de negocios y ciencia de datos, mejorando la toma de decisiones basada en datos.
Implementación de Modelos de Ciencia de Datos	Desarrollar modelos de regresión lineal múltiple, regresión logística y clusterización K-medias para analizar tendencias de ventas, predecir comportamientos de clientes y segmentar productos.	Proporciona *insights* valiosos sobre el comportamiento de ventas y clientes, facilitando estrategias de marketing y optimización de inventarios.
Integración de un Sistema ERP/CRM (Odoo)	Configurar e implementar Odoo como plataforma unificada para la gestión de procesos empresariales y relaciones con clientes.	Mejora la eficiencia operativa, la colaboración interna y la gestión de relaciones con clientes mediante una herramienta centralizada.
Capacitación y Soporte al Personal	Proporcionar formación en el uso de Odoo y herramientas de inteligencia artificial como ChatGPT y DALL-E.	Aumenta la eficiencia y creatividad del equipo, facilita la generación de contenido de alta calidad y mejora la atención al cliente.
Optimización de Canales de Comunicación	Unificar las líneas telefónicas en una única línea centralizada para la gestión de pedidos y atención al cliente.	Reduce la confusión en la gestión de pedidos, mejora la experiencia del cliente y optimiza el proceso de atención al cliente.

2. EXTRACCIÓN Y MINERÍA DE DATOS

En esta sección, se describe el proceso de creación de las tablas en la base de datos que servirán como el *Data Warehouse* para la empresa. El objetivo fue diseñar una estructura de base de datos relacional que permitiera almacenar de manera eficiente los datos extraídos del sistema Odoo. A continuación, se detalla el código SQL empleado y las consideraciones tomadas durante su implementación, además del diagrama de entidad relación de la base de datos.

2.1. Diseño de la Base de Datos

Para este caso, se selecciono el sistema de base de datos MySQL. A continuación se detalle las sentencias SQL que se utilizaron para la creación del esquema de la base de datos.

Antes de proceder con la creación de las tablas, fue necesario desactivar temporalmente las restricciones de clave foránea utilizando el comando

```
1 SET FOREIGN_KEY_CHECKS = 0;
```

Esta práctica es común cuando se necesita realizar operaciones masivas en la base de datos, como la creación o eliminación de múltiples tablas, sin que las restricciones de integridad referencial interfieran en el proceso. Al desactivar las verificaciones de claves foráneas, se evita que el motor de la base de datos realice comprobaciones de integridad referencial en cada operación, lo que agiliza significativamente el proceso.

El diseño de la base de datos se estructuró en torno a tablas de dimensiones y hechos. Las tablas de dimensiones almacenan atributos descriptivos, mientras que las tablas de hechos contienen datos transaccionales y métricas cuantitativas.

- **Tablas de Dimensiones**: Se crearon las siguientes tablas de dimensiones para almacenar información estática y categorizada que se asocia a los hechos:

- `dim_empresa`: Contiene información sobre las empresas.

```sql
CREATE TABLE IF NOT EXISTS dim_empresa (
    empresa_id INT PRIMARY KEY,
    nombre_empresa VARCHAR(255)
);
```

- `dim_calendario_recursos`: Almacena información sobre los calendarios de recursos.

```sql
CREATE TABLE IF NOT EXISTS dim_calendario_recursos (
    calendario_recursos_id INT PRIMARY KEY,
    nombre_calendario VARCHAR(255)
);
```

- `dim_empleado`: Guarda información detallada sobre los empleados.

```sql
CREATE TABLE IF NOT EXISTS dim_empleado (
    empleado_id INT PRIMARY KEY,
    nombre_completo VARCHAR(255),
    imagen LONGBLOB,
    empresa_id INT,
    calendario_recursos_id INT,
    titulo_trabajo VARCHAR(255),
    telefono_trabajo VARCHAR(50),
    telefono_movil VARCHAR(50),
    email_trabajo VARCHAR(255),
    genero VARCHAR(10),
    estado_civil VARCHAR(50),
    fecha_creacion DATETIME,
    id_identificacion VARCHAR(50),
    fecha_nacimiento DATE,
    codigo_barras VARCHAR(50),
    direccion_privada VARCHAR(255),
    FOREIGN KEY (empresa_id) REFERENCES dim_empresa(empresa_id),
    FOREIGN KEY (calendario_recursos_id) REFERENCES
    dim_calendario_recursos(calendario_recursos_id)
);
```

- `dim_producto`: Contiene información sobre los productos.

```sql
CREATE TABLE IF NOT EXISTS dim_producto (
    producto_id INT PRIMARY KEY,
    nombre VARCHAR(255),
    descripcion TEXT,
```

```
    imagen LONGBLOB,
    precio_lista FLOAT,
    precio_estandar FLOAT,
    tipo_detallado VARCHAR(50),
    categoria_id INT,
    cantidad_disponible FLOAT,
    venta_permitida BOOLEAN,
    compra_permitida BOOLEAN,
    codigo_barras VARCHAR(50),
    codigo_default VARCHAR(50),
    unidad_medida_id INT,
    activo BOOLEAN,
    disponible_en_pos BOOLEAN,
    FOREIGN KEY (categoria_id) REFERENCES dim_categoria(categoria_id),
    FOREIGN KEY (unidad_medida_id) REFERENCES
    dim_unidad_medida(unidad_medida_id)
);
```

- `dim_cliente`: Almacena datos sobre los clientes.

```
CREATE TABLE IF NOT EXISTS dim_cliente (
    cliente_id INT PRIMARY KEY,
    nombre VARCHAR(255),
    empresa_id INT,
    tipo VARCHAR(50),
    direccion VARCHAR(255),
    ciudad VARCHAR(255),
    estado_id INT,
    pais_id INT,
    email VARCHAR(255),
    telefono VARCHAR(50),
    movil VARCHAR(50)
);
```

- `dim_fecha`: Contiene información de calendario y fechas.

```
CREATE TABLE IF NOT EXISTS dim_fecha (
    fecha_id INT PRIMARY KEY,
    fecha DATE,
    dia INT,
    mes INT,
    anio INT,
    dia_semana INT,
    nombre_mes VARCHAR(20),
    anio_mes VARCHAR(20),
```

```
    trimestre VARCHAR(10)
);
```

- `dim_categoria`: Almacena las categorías de productos.

```
CREATE TABLE IF NOT EXISTS dim_categoria (
    categoria_id INT PRIMARY KEY,
    nombre_categoria VARCHAR(255)
);
```

- `dim_unidad_medida`: Guarda la información sobre las unidades de medida de los productos.

```
CREATE TABLE IF NOT EXISTS dim_unidad_medida (
    unidad_medida_id INT PRIMARY KEY,
    nombre_unidad_medida VARCHAR(255)
);
```

- **Tablas de Hechos**: Se crearon las siguientes tablas de hechos para almacenar transacciones y datos cuantitativos:

  - `fact_venta`: Registra las transacciones de ventas.

```
CREATE TABLE IF NOT EXISTS fact_venta (
    venta_id INT PRIMARY KEY,
    fecha_venta DATETIME,
    monto_total FLOAT,
    empresa_id INT,
    cliente_id INT,
    FOREIGN KEY (empresa_id) REFERENCES dim_empresa(empresa_id),
    FOREIGN KEY (cliente_id) REFERENCES dim_cliente(cliente_id)
);
```

  - `fact_detalle_venta`: Almacena los detalles de cada venta, incluyendo los productos vendidos.

```
CREATE TABLE IF NOT EXISTS fact_detalle_venta (
    venta_id INT,
    detalle_id INT PRIMARY KEY,
    producto_id INT,
    precio_unitario FLOAT,
    cantidad FLOAT,
    subtotal FLOAT,
    nombre_completo_producto VARCHAR(255),
```

```
    FOREIGN KEY (venta_id) REFERENCES fact_venta(venta_id),
    FOREIGN KEY (producto_id) REFERENCES dim_producto(producto_id)
);
```

Una vez creadas todas las tablas necesarias, se reactivaron las restricciones de clave foránea utilizando

```
SET FOREIGN_KEY_CHECKS = 1;
```

Esto garantiza que, a partir de ese momento, cualquier inserción o actualización de datos que viole las reglas de integridad referencial sea rechazada por el sistema de base de datos.

Este enfoque permitió crear una base de datos robusta y eficiente, preparada para almacenar y analizar grandes volúmenes de datos extraídos del sistema Odoo. Por ultimo, lo siguiente es el diagrama de entidad relación de la base de datos resultante:

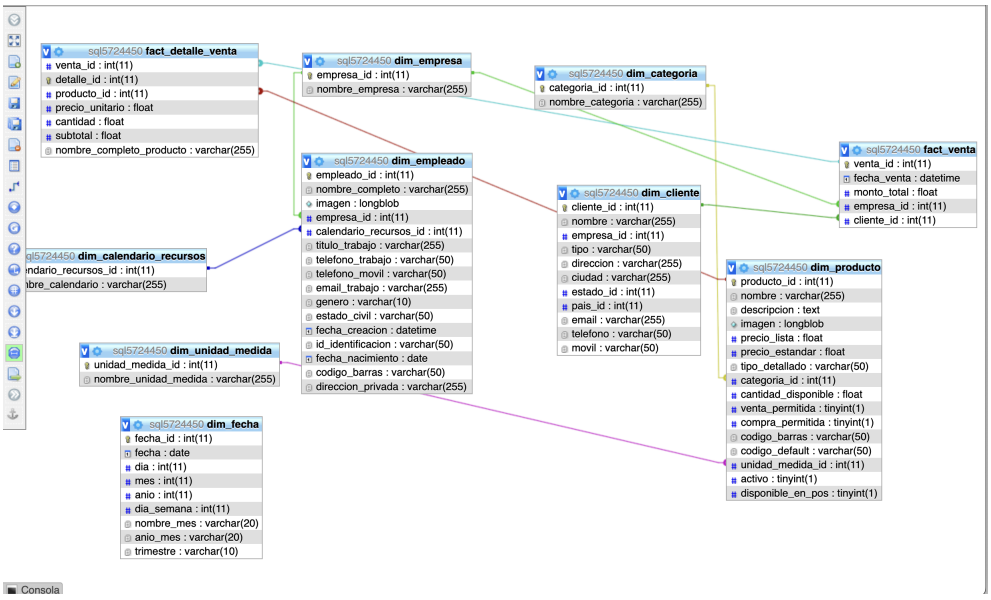


Figura 2.1. Diagrama de entidad relación de la base de datos del *Data Warehouse*

2.2. Extracción de Datos

La extracción de datos es una parte fundamental del proceso ETL, ya que implica obtener la información desde el sistema fuente, Odoo, para posteriormente

transformarla y cargarla en el *Data Warehouse*, el proceso de extracción de datos desde el sistema, utilizando la API XML-RPC.

Para interactuar con la base de datos de Odoo, se configuró una conexión utilizando el protocolo XML-RPC. Este protocolo permite la comunicación remota a través de XML, posibilitando la ejecución de comandos en Odoo desde un entorno Python.

Se configuraron los siguientes parámetros de conexión:

- **URL del Servidor Odoo**: https://trescielos.odoo.com/
- **Base de Datos**: trescielos-main-12427981
- **Credenciales de Usuario**: Se utilizaron las credenciales necesarias (username y password) para autenticarse en la API de Odoo.

La autenticación se realizó mediante el método authenticate de la API común de Odoo, que devuelve un uid (User ID) necesario para realizar operaciones sobre los objetos de Odoo.

```
common = xmlrpc.client.ServerProxy('{}/xmlrpc/2/common'.format(self.url))
self.uid = common.authenticate(self.db, self.username, self.password, {})
self.models = xmlrpc.client.ServerProxy('{}/xmlrpc/2/object'.format(self.url))
```

Se implementaron varios métodos para extraer diferentes conjuntos de datos desde Odoo, correspondientes a empleados, productos, ventas, categorías, unidades de medida, empresas, calendarios de recursos, clientes, y fechas. Cada método utiliza la función `search_read` de la API de Odoo, que permite buscar y leer registros en una única operación. A continuación, se describen los métodos implementados y los campos específicos extraídos en cada caso.

El método `empleados` fue diseñado para extraer información de los empleados registrados en el sistema Odoo, la extracción de estos datos es crucial para alimentar la tabla de `dim_empleado` en la base de datos. Se especificaron varios campos relevantes que permiten describir a cada empleado de manera completa:

- **id**: Identificador único del empleado.
- **display_name**: Nombre completo del empleado.
- **image_1920**: Imagen de perfil del empleado.

- **company_id**: Identificador de la empresa a la que pertenece el empleado.
- **resource_calendar_id**: Identificador del calendario de recursos asociado al empleado.
- **job_title**: Título del trabajo o cargo del empleado.
- **work_phone**: Número de teléfono del trabajo del empleado.
- **mobile_phone**: Número de teléfono móvil del empleado.
- **work_email**: Correo electrónico del trabajo del empleado.
- **gender**: Género del empleado.
- **marital**: Estado civil del empleado.
- **create_date**: Fecha de creación del registro del empleado.
- **identification_id**: Número de identificación del empleado.
- **birthday**: Fecha de nacimiento del empleado.
- **barcode**: Código de barras asignado al empleado.
- **private_street**: Dirección privada del empleado.

```
fields = [
    'id', 'display_name', 'image_1920', 'company_id', 'resource_calendar_id',
    'job_title', 'work_phone', 'mobile_phone', 'work_email',
    'gender', 'marital', 'create_date', 'identification_id',
    'birthday', 'barcode', 'private_street'
]

empleados = self.models.execute_kw(
    self.db,
    self.uid,
    self.password,
    'hr.employee',
    'search_read',
    [],
    {'fields': fields}
)
```

El método `productos` se encarga de extraer información detallada sobre los productos disponibles en Odoo. La extracción de estos datos es esencial para alimentar la tabla de `dim_producto` en la base de datos. Se seleccionaron campos que cubren tanto la descripción básica del producto como su categorización y disponibilidad:

- **id**: Identificador único del producto.
- **name**: Nombre del producto.
- **description**: Descripción detallada del producto.
- **image_1920**: Imagen del producto.
- **list_price**: Precio de lista del producto.
- **standard_price**: Precio estándar del producto.
- **detailed_type**: Tipo detallado del producto.
- **categ_id**: Identificador de la categoría a la que pertenece el producto.
- **qty_available**: Cantidad disponible del producto.
- **sale_ok**: Indicador de si el producto está disponible para la venta.
- **purchase_ok**: Indicador de si el producto está disponible para la compra.
- **barcode**: Código de barras del producto.
- **default_code**: Código interno o SKU del producto.
- **uom_id**: Identificador de la unidad de medida del producto.
- **active**: Indicador de si el producto está activo.
- **available_in_pos**: Indicador de si el producto está disponible en el punto de venta.

```
fields = [
'id', 'name', 'description', 'image_1920', 'list_price', 'standard_price',
'detailed_type', 'categ_id', 'qty_available', 'sale_ok', 'purchase_ok',
'barcode', 'default_code', 'uom_id', 'active', 'available_in_pos'
]

productos = self.models.execute_kw(
```

```
self.db,
self.uid,
self.password,
'product.product',
'search_read',
[[]],
{'fields': fields}
)
```

El método `ventas` fue implementado para extraer datos relacionados con las transacciones de ventas en Odoo. La información extraída es fundamental para poblar la tabla de `fact_venta` en la base de datos. Se especificaron varios campos clave que permiten registrar los detalles de cada venta:

- **id**: Identificador único de la venta.
- **date_order**: Fecha en la que se realizó la venta.
- **amount_total**: Monto total de la venta.
- **company_id**: Identificador de la empresa que realizó la venta.
- **partner_id**: Identificador del cliente que realizó la compra.
- **lines**: Identificadores de las líneas de detalle de la venta.

```
fields = [
'id', 'date_order', 'amount_total', 'company_id', 'partner_id', 'lines'
]

ventas = self.models.execute_kw(
self.db,
self.uid,
self.password,
'pos.order',
'search_read',
[[]],
{'fields': fields}
)
```

El método `detalles_venta` extrae información detallada sobre los productos vendidos en cada transacción de venta. Estos datos son cruciales para poblar la tabla de `fact_detalle_venta` en la base de datos. A continuación, se describen los campos extraídos:

- **venta_id**: Identificador de la venta a la que pertenece el detalle.
- **detalle_id**: Identificador único del detalle de la venta.
- **producto_id**: Identificador del producto vendido.
- **precio_unitario**: Precio unitario del producto.
- **cantidad**: Cantidad vendida del producto.
- **subtotal**: Subtotal de la línea de detalle (precio unitario * cantidad).
- **nombre_completo_producto**: Nombre completo del producto vendido.

```
lines = self.models.execute_kw(
self.db,
self.uid,
self.password,
'pos.order.line',
'search_read',
[[['id', 'in', line_ids]]],
{}
)

for line in lines:
detalles_venta.append({
'venta_id': venta['id'],
'detalle_id': line['id'],
'producto_id': line['product_id'][0],
'precio_unitario': line['price_unit'],
'cantidad': line['qty'],
'subtotal': line['price_subtotal'],
'nombre_completo_producto': line['full_product_name']
})
```

El método `categorias` se utiliza para extraer las categorías de productos desde Odoo, las cuales son fundamentales para organizar y categorizar los productos en la base de datos. Los campos extraídos son esenciales para poblar la tabla de `dim_categoria`:

- **id**: Identificador único de la categoría.
- **name**: Nombre de la categoría.

```
fields = ['id', 'name']

categorias = self.models.execute_kw(
self.db,
self.uid,
self.password,
'product.category',
'search_read',
[[]],
{'fields': fields}
)
```

El método `unidades_medida` se encarga de extraer las unidades de medida de los productos, que son esenciales para gestionar y estandarizar las cantidades en las transacciones. Estos datos son utilizados para poblar la tabla de `dim_unidad_medida`:

- **id**: Identificador único de la unidad de medida.
- **name**: Nombre de la unidad de medida.

```
fields = ['id', 'name']

unidades_medida = self.models.execute_kw(
self.db,
self.uid,
self.password,
'uom.uom',
'search_read',
[[]],
{'fields': fields}
)
```

El método `empresas` se implementó para extraer la información de las empresas registradas en el sistema Odoo. Estos datos son necesarios para alimentar la tabla de `dim_empresa` en la base de datos. Se especificaron los siguientes campos clave:

- **id**: Identificador único de la empresa.
- **name**: Nombre de la empresa.

```
fields = ['id', 'name']

empresas = self.models.execute_kw(
```

```
self.db,
self.uid,
self.password,
'res.company',
'search_read',
[[]],
{'fields': fields}
)
```

El método `calendarios_recursos` extrae la información relacionada con los calendarios de recursos, que son utilizados para gestionar la disponibilidad de empleados y recursos dentro de la empresa. Estos datos se utilizan para poblar la tabla de `dim_calendario_recursos`:

- **id**: Identificador único del calendario de recursos.
- **name**: Nombre del calendario de recursos.

```
fields = ['id', 'name']

calendarios_recursos = self.models.execute_kw(
self.db,
self.uid,
self.password,
'resource.calendar',
'search_read',
[[]],
{'fields': fields}
)
```

El método `clientes` se utiliza para extraer la información de los clientes registrados en el sistema Odoo, los cuales son esenciales para las transacciones de venta. Los campos extraídos son cruciales para poblar la tabla de `dim_cliente`:

- **id**: Identificador único del cliente.
- **name**: Nombre del cliente.
- **company_id**: Identificador de la empresa asociada al cliente.
- **type**: Tipo de cliente (individual o empresa).
- **street**: Calle de la dirección del cliente.

- **city**: Ciudad de la dirección del cliente.
- **state_id**: Identificador del estado de la dirección del cliente.
- **country_id**: Identificador del país de la dirección del cliente.
- **email**: Correo electrónico del cliente.
- **phone**: Número de teléfono del cliente.
- **mobile**: Número de teléfono móvil del cliente.

```
fields = [
'id', 'name', 'company_id', 'type', 'street', 'city',
'state_id', 'country_id', 'email', 'phone', 'mobile'
]

clientes = self.models.execute_kw(
self.db,
self.uid,
self.password,
'res.partner',
'search_read',
[['customer_rank', '>', 0]],
{'fields': fields}
)
```

El proceso de generación de fechas es clave para cualquier análisis temporal de los datos en el *Data Warehouse*. Este método genera automáticamente las fechas necesarias, que son utilizadas para poblar la tabla de `dim_fecha`. Se generaron los siguientes campos:

- **fecha**: Fecha en formato YYYY-MM-DD.
- **dia**: Día del mes (1-31).
- **mes**: Mes del año (1-12).
- **anio**: Año.
- **dia_semana**: Día de la semana (1=Lunes, 7=Domingo).
- **nombre_mes**: Nombre del mes.
- **anio_mes**: Combinación del año y el mes en formato YYYY-MM.

- **trimestre**: Trimestre del año (1-4).

```python
import datetime

def generar_fechas(inicio, fin):
    fechas = []
    inicio = datetime.datetime.strptime(inicio, '%Y-%m-%d')
    fin = datetime.datetime.strptime(fin, '%Y-%m-%d')
    delta = datetime.timedelta(days=1)
    while inicio <= fin:
        fechas.append({
            'fecha': inicio.date(),
            'dia': inicio.day,
            'mes': inicio.month,
            'anio': inicio.year,
            'dia_semana': inicio.isoweekday(),
            'nombre_mes': inicio.strftime('%B'),
            'anio_mes': f"{inicio.year}-{inicio.month}",
            'trimestre': (inicio.month-1)//3 + 1
        })
        inicio += delta
    return fechas
```

2.3. Transformación y Carga de Datos en MySQL

El objetivo principal de esta etapa es asegurar que los datos sean transformados de manera adecuada y que se inserten correctamente en las tablas correspondientes, manteniendo la integridad y coherencia de la información. Se detalla el proceso de transformación de los datos extraídos de Odoo y su carga en el *Data Warehouse*, utilizando MySQL como sistema de gestión de bases de datos.

2.3.1. Inserción de Datos en Tablas de Dimensiones

El proceso de inserción de datos en las tablas de dimensiones fue una de las primeras tareas realizadas durante la carga de datos. Las tablas de dimensiones (`dim_empresa`, `dim_calendario_recursos`, `dim_empleado`, `dim_producto`, `dim_cliente`, `dim_fecha`, `dim_categoria`, `dim_unidad_medida`) almacenan información descriptiva que se utiliza para categorizar y relacionar los datos transaccionales almacenados en las

tablas de hechos.

Antes de insertar los datos en las tablas de dimensiones, se realizó una transformación para asegurar que los datos estuvieran en el formato adecuado y que se cumpliera con las reglas de negocio definidas. Esta transformación incluyó:

- **Normalización de Datos**: Asegurar que los datos fueran consistentes, eliminando duplicados y garantizando que las claves primarias y foráneas estuvieran correctamente asignadas.
- **Manejo de Nulos y Vacíos**: Se implementaron estrategias para manejar valores nulos o vacíos, como el uso de valores por defecto o la omisión de registros incompletos cuando fuera necesario.

Durante la inserción de datos en las tablas de dimensiones, se implementaron mecanismos para manejar posibles entradas duplicadas. Esto fue especialmente importante en tablas como `dim_empresa` y `dim_cliente`, donde es común que múltiples registros puedan referirse a la misma entidad en Odoo.

Para evitar la inserción de datos duplicados, se utilizó un control de duplicados basado en las claves primarias. Si se detectaba un duplicado, el registro se omitía y se registraba un mensaje de advertencia en la consola:

```
def insert_dim(self, table, data):
    cursor = self.connection.cursor()
    for row in data:
        placeholders = ', '.join(['%s'] * len(row))
        columns = ', '.join(row.keys())
        sql = f"INSERT INTO {table} ({columns}) VALUES ({placeholders})"
        try:
            cursor.execute(sql, tuple(row.values()))
        except mysql.connector.errors.IntegrityError as e:
            if "Duplicate entry" in str(e):
                print(f"Skipping duplicate entry in {table}: {row}")
            else:
                raise e
    self.connection.commit()
    cursor.close()
```

Este código asegura que se mantenga la integridad de los datos y evita la inserción de registros redundantes.

2.3.2. Inserción de Datos en Tablas de Hechos

Después de poblar las tablas de dimensiones, se procedió a la inserción de datos en las tablas de hechos (`fact_venta` y `fact_detalle_venta`). Las tablas de hechos contienen datos transaccionales que son clave para el análisis de las operaciones comerciales realizadas.

La inserción en las tablas de hechos requiere una cuidadosa consideración de las relaciones entre las tablas de hechos y las dimensiones. Cada registro en una tabla de hechos está relacionado con uno o más registros en las tablas de dimensiones a través de claves foráneas. Por ejemplo, la tabla `fact_venta` se relaciona con `dim_empresa` y `dim_cliente` para identificar la empresa que realizó la venta y el cliente que la recibió.

Durante la inserción, se aseguró que todas las claves foráneas estuvieran correctamente alineadas con los datos en las tablas de dimensiones. Esto se logró mediante un proceso de validación previo, donde se verificaba la existencia de los registros en las tablas de dimensiones antes de realizar la inserción en las tablas de hechos.

```
def insert_ventas(self, ventas, detalles_venta, dictionaries):
    cursor = self.connection.cursor()
    empresa_dict = dictionaries['company_id']
    cliente_dict = dictionaries['partner_id']

    # Insertar empresa_id en dim_empresa
    empresas = [{'empresa_id': k, 'nombre_empresa': v} for k, v in
    empresa_dict.items()]
    self.insert_dim('dim_empresa', empresas)

    # Insertar cliente_id en dim_cliente
    clientes = [{'cliente_id': k, 'nombre': v} for k, v in cliente_dict.items()]
    self.insert_dim('dim_cliente', clientes)

    for venta in ventas:
        sql = """
        INSERT INTO fact_venta (venta_id, fecha_venta, monto_total, empresa_id,
    cliente_id)
        VALUES (%s, %s, %s, %s, %s)
        """
        values = (
            venta['id'], venta['date_order'], venta['amount_total'],
```

```
            venta['company_id'][0] if venta['company_id'] else None,
            venta['partner_id'][0] if venta['partner_id'] else None
        )
        try:
            cursor.execute(sql, values)
        except mysql.connector.errors.IntegrityError as e:
            if "Duplicate entry" in str(e):
                print(f"Skipping duplicate entry in fact_venta: {venta}")
            else:
                raise e
    self.connection.commit()
    cursor.close()
```

El código anterior ilustra cómo se manejó la inserción en la tabla `fact_venta`, asegurando que las relaciones entre las tablas estuvieran correctamente establecidas y que no se introdujeran datos inconsistentes. Al igual que con las dimensiones, se implementaron mecanismos para manejar duplicados y errores de integridad referencial.

Para los detalles de venta, se siguió un proceso similar, validando que cada línea de detalle estuviera vinculada a un producto y a la venta correspondiente:

```
for detalle in detalles_venta:
    sql = """
    INSERT INTO fact_detalle_venta (venta_id, detalle_id, producto_id,
    precio_unitario, cantidad, subtotal, nombre_completo_producto)
    VALUES (%s, %s, %s, %s, %s, %s, %s)
    """
    values = (
        detalle['venta_id'], detalle['detalle_id'],
        detalle['producto_id'],
        detalle['precio_unitario'], detalle['cantidad'],
        detalle['subtotal'], detalle['nombre_completo_producto']
    )
    try:
        cursor.execute(sql, values)
    except mysql.connector.errors.IntegrityError as e:
        if "Duplicate entry" in str(e):
            print(f"Skipping duplicate entry in fact_detalle_venta: {detalle}")
        else:
            print(f"{Fore.RED}{detalle}")
            raise e
```

Este enfoque permitió una inserción robusta y confiable de datos en el *Data Warehouse*, asegurando que la información fuera precisa y estuviera lista para análisis posteriores.

2.4. Ejecución del Script ETL

Se presenta un análisis del flujo del script y un resumen de los resultados obtenidos, incluyendo tiempos de ejecución, cantidad de datos procesados y la gestión de cualquier incidencia relevante. Se detalla la ejecución del script ETL, que abarca desde la creación inicial de la base de datos hasta la inserción final de los datos en las tablas de dimensiones y hechos.

El script ETL se diseñó para ejecutarse de manera secuencial, siguiendo un flujo lógico que garantiza la correcta extracción, transformación y carga de los datos en el *Data Warehouse*. A continuación se describen los pasos principales de la ejecución del script:

1. **Creación de la Base de Datos**: El primer paso del script consiste en crear las tablas de la base de datos, tanto de dimensiones como de hechos, utilizando el código SQL previamente definido. Durante esta etapa, se desactivan temporalmente las restricciones de clave foránea para facilitar la creación de las tablas sin conflictos.

2. **Extracción de Datos**: Una vez que las tablas están creadas, el script procede a conectar con la API de Odoo utilizando XML-RPC para extraer los datos necesarios. Se ejecutan los métodos de extracción para empleados, productos, ventas, categorías, unidades de medida, empresas, calendarios de recursos, y clientes. Los datos extraídos se almacenan temporalmente en estructuras de datos en Python.

3. **Transformación de los Datos**: Los datos extraídos son transformados para asegurar que se ajusten al esquema de la base de datos. Esta transformación incluye la normalización de los datos, el manejo de valores nulos o vacíos, y la preparación de las claves foráneas para su inserción en las tablas correspondientes.

4. **Carga de Datos en Tablas de Dimensiones**: El script inserta primero los datos en las tablas de dimensiones. Durante esta fase, se manejan posibles

duplicados y se asegura que todas las relaciones con las tablas de hechos estén preparadas.

5. **Carga de Datos en Tablas de Hechos**: Una vez que las tablas de dimensiones están pobladas, el script carga los datos transaccionales en las tablas de hechos. Este paso incluye la validación de las claves foráneas y la inserción de detalles específicos de las ventas.

6. **Finalización y Cierre de la Conexión**: Al completar la carga de datos, el script finaliza cerrando la conexión con la base de datos y mostrando un resumen de la ejecución.

2.4.1. Resultados de la Ejecución

Al ejecutar el script ETL, se obtuvieron los siguientes resultados:

- **Cantidad de Datos Procesados**: Se procesaron los siguientes volúmenes de datos:
  - **Empleados**: Se insertaron *22* registros en la tabla `dim_empleado`.
  - **Productos**: Se insertaron *274* registros en la tabla `dim_producto`.
  - **Ventas**: Se insertaron *6,138* registros en la tabla `fact_venta`.
  - **Detalles de Venta**: Se insertaron *12,167* registros en la tabla `fact_detalle_venta`.
  - **Categorías, Unidades de Medida, Empresas, Calendarios de Recursos y Clientes**: Se insertaron los correspondientes registros en sus respectivas tablas de dimensiones.

- **Manejo de Incidencias**: Durante la ejecución, se manejaron correctamente las incidencias relacionadas con la inserción de datos duplicados. En cada caso de duplicado, el script registró el incidente y omitió la inserción del registro redundante, asegurando la integridad de los datos en la base de datos.

- **Errores Manejados**: No se presentaron errores críticos durante la ejecución del script. Cualquier error menor, como problemas de integridad referencial, fue manejado adecuadamente, permitiendo que el proceso ETL se completara con éxito.

- **Tiempos de Ejecución**: El tiempo total de ejecución del script se registró para evaluar la eficiencia del proceso. Este tiempo incluye todas las etapas, desde la creación de la base de datos hasta la inserción final de los datos. El script mostró un rendimiento adecuado, completando la ejecución en un tiempo razonable de 34 minutos para la cantidad de datos procesados. La orquestación del proceso ETL se llevó a cabo en un servidor Ubuntu 22.04 montado en una Raspberry Pi 3 Modelo B, equipada con 1 GB de RAM y un procesador Quad-core ARM Cortex-A53 a 1.2 GHz.

El script ETL cumplió con su objetivo de extraer, transformar y cargar los datos en el *Data Warehouse* de manera eficiente y con un manejo adecuado de posibles incidencias. El proceso quedó registrado para futuras referencias y análisis de rendimiento.

2.5. Problemas Encontrados y Soluciones Implementadas

Durante la ejecución del proyecto ETL, surgieron varios desafíos técnicos que requirieron soluciones específicas para garantizar la integridad y la eficacia del proceso. En esta sección, se describen los problemas más significativos encontrados y las soluciones implementadas para resolverlos.

Uno de los problemas más recurrentes fue el manejo de entradas duplicadas durante la inserción de datos en las tablas de la base de datos. Los duplicados son comunes cuando se extraen datos de un sistema de gestión como Odoo, donde múltiples registros pueden referirse a la misma entidad debido a la naturaleza repetitiva de las transacciones o errores en la entrada de datos.

Durante la inserción de datos en las tablas de dimensiones y hechos, se identificaron varios casos en los que el mismo registro intentaba insertarse más de una vez, lo que provocaba errores de integridad al intentar violar las restricciones de clave primaria. Esto ocurrió, por ejemplo, al intentar insertar empresas, clientes o productos que ya existían en la base de datos.

Para abordar este problema, se implementó un control de duplicados en el código de inserción. La solución consistió en capturar la excepción de integridad que se

produce cuando se intenta insertar un registro duplicado. En lugar de detener la ejecución del script, el código captura la excepción, registra un mensaje de advertencia y omite la inserción del registro duplicado. Esto asegura que el proceso continúe sin interrupciones y que la base de datos mantenga su integridad.

El siguiente fragmento de código muestra cómo se manejó este problema en la función de inserción de datos:

```
def insert_dim(self, table, data):
    cursor = self.connection.cursor()
    for row in data:
        placeholders = ', '.join(['%s'] * len(row))
        columns = ', '.join(row.keys())
        sql = f"INSERT INTO {table} ({columns}) VALUES ({placeholders})"
        try:
            cursor.execute(sql, tuple(row.values()))
        except mysql.connector.errors.IntegrityError as e:
            if "Duplicate entry" in str(e):
                print(f"Skipping duplicate entry in {table}: {row}")
            else:
                raise e
    self.connection.commit()
    cursor.close()
```

Este enfoque permitió manejar los duplicados de manera eficiente, evitando interrupciones en el flujo del proceso ETL y garantizando que solo los registros únicos fueran insertados en las tablas de la base de datos.

2.5.1. Otros Errores y Resoluciones

Además de los problemas relacionados con los duplicados, surgieron otros desafíos durante el desarrollo del proyecto. A continuación, se detallan algunos de los más relevantes y las soluciones implementadas:

1. **Errores de Integridad Referencial**: Durante la inserción de datos en las tablas de hechos, surgieron problemas de integridad referencial cuando los registros en las tablas de dimensiones no se encontraban completamente sincronizados con los datos que se intentaban insertar. Esto ocurría, por ejemplo, cuando una venta hacía referencia a un cliente o producto que no estaba previamente registrado en la base de datos.

Para solucionar este problema, se implementó un proceso de validación previo a la inserción en las tablas de hechos. Antes de insertar los registros, el script verifica que todas las claves foráneas estén correctamente alineadas con las tablas de dimensiones. Si una clave foránea no coincide, se omite la inserción del registro en la tabla de hechos y se registra un mensaje de advertencia.

```
        for venta in ventas:
            sql = """
            INSERT INTO fact_venta (venta_id, fecha_venta, monto_total,
    empresa_id, cliente_id)
            VALUES (%s, %s, %s, %s, %s)
            """
            values = (
                venta['id'], venta['date_order'], venta['amount_total'],
                venta['company_id'][0] if venta['company_id'] else None,
                venta['partner_id'][0] if venta['partner_id'] else None
            )
            try:
                cursor.execute(sql, values)
            except mysql.connector.errors.IntegrityError as e:
                if "Duplicate entry" in str(e):
                    print(f"Skipping duplicate entry in fact_venta:
    {venta}")
                else:
                    raise e

```

2. **Manejo de Campos Nulos o Vacíos**: Otro problema común fue el manejo de campos nulos o vacíos en los datos extraídos de Odoo. Algunos campos, como números de teléfono o correos electrónicos, a veces estaban vacíos o nulos, lo que podría causar problemas durante la inserción en la base de datos.

   Para abordar este problema, se implementaron valores por defecto y se realizaron verificaciones adicionales antes de insertar los datos. Por ejemplo, si un campo crítico como un identificador de empresa estaba vacío, el registro se omitía, o se asignaba un valor por defecto si el campo vacío no afectaba la integridad de los datos.

```
        values = (
            emp['id'], emp['display_name'], emp['image_1920'],
            emp['company_id'][0] if emp['company_id'] else None,
            emp['resource_calendar_id'][0] if emp['resource_calendar_id']
    else None,
```

```
            emp['job_title'], emp['work_phone'] if emp['work_phone'] else
    None,
            emp['mobile_phone'] if emp['mobile_phone'] else None,
    emp['work_email'] if emp['work_email'] else None,
            emp['gender'] if emp['gender'] else None, emp['marital'],
    emp['create_date'],
            emp['identification_id'] if emp['identification_id'] else None,
            emp['birthday'] if emp['birthday'] else None,
            emp['barcode'] if emp['barcode'] else None,
            emp['private_street'] if emp['private_street'] else None
        )

```

El manejo eficaz de los duplicados, la validación de la integridad referencial y el tratamiento adecuado de los campos nulos fueron clave para el éxito del proyecto ETL, asegurando la calidad y fiabilidad de los datos cargados en el *Data Warehouse*.

3. FUNDAMENTOS MATEMÁTICOS

3.1. Regresión lineal

Definición 3.1. Modelo estadístico lineal. Un modelo estadístico lineal que relaciona una respuesta aleatoria Y con un conjunto de variables $x_1, x_2, ..., x_k$ es de la forma

$$Y = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + \epsilon,$$

donde $\beta_0, ..., \beta_k$ son parámetros desconocidos, ϵ es una variable aleatoria y $x_1, ..., x_k$ toman valores conocidos. Supondremos que $E(\epsilon) = 0$ y que $V(Y) = V(\epsilon) = \sigma^2$, y por tanto, que

$$E(Y) = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k.$$

Una forma de poder ajustar un modelo estadístico lineal es con el **Método de Mínimos Cuadrados.** Este método se basa en la idea de que si queremos ajustar una recta a un conjunto de puntos, y suponiendo que el modelo que queremos ajustar es $E(Y) = \beta_0 + \beta_1 x_1$, es decir, postulamos que $Y = \beta_0 + \beta_1 x_1 + \epsilon$, donde ϵ tiene una distribución de probabilidad con $E(\epsilon) = 0$, y si $\hat{\beta}_0$ y $\hat{\beta}_1$ son estimadores de β_0 y β_1 respectivamente, entonces $\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 x_1$ es un estimador de Y y para ajustar una recta que pase por un conjunto de n puntos, deseamos que la diferencia entre los valores observador y los puntos de la recta sea pequeña. Matemáticamente, podríamos plantear esto como el buscar minimizar

$$\begin{aligned} \text{SSE} &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{n} (y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_1))^2, \end{aligned}$$

donde y_i es el i-ésimo valor observado, y $\hat{y}_i$ el i-ésimo valor pronosticado.

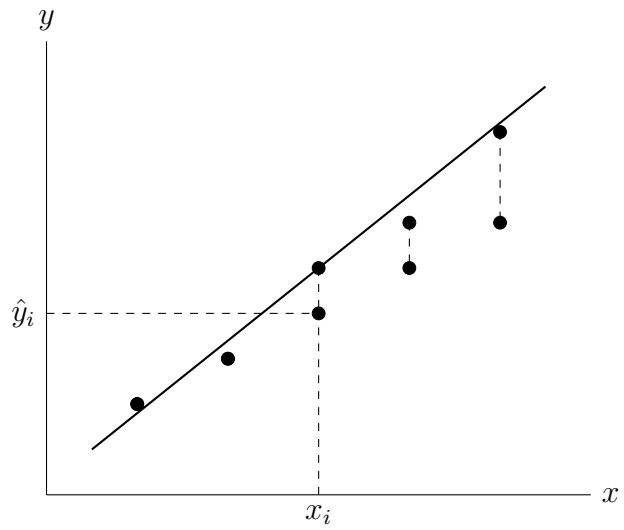


Figura 3.1. Ajuste de una recta que pasa por un conjunto de puntos

Si SSE tiene un mínimo, ocurrirá para valores β_0 y β_1 que satisfagan

$$\begin{aligned}\frac{\partial \text{SSE}}{\partial \hat{\beta}_0} &= \frac{\partial \left(\sum_{i=1}^{n}(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_1))^2\right)}{\partial \hat{\beta}_0} \\ &= -\sum_{i=1}^{n} 2\left(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_1)\right) \\ &= -2\left(\sum_{i=1}^{n} y_i - n\hat{\beta}_0 - \hat{\beta}_1 \sum_{i=1}^{n} x_i\right) \\ &= 0,\end{aligned}$$

y

$$\begin{aligned}\frac{\partial \text{SSE}}{\partial \hat{\beta}_1} &= \frac{\partial \left(\sum_{i=1}^{n}(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_1))^2\right)}{\partial \hat{\beta}_1} \\ &= -2\sum_{i=1}^{n}(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))x_i \\ &= -2\left(\sum_{i=1}^{n} x_i y_i - \hat{\beta}_0 \sum_{i=1}^{n} x_i - \hat{\beta}_1 \sum_{i=1}^{n} x_i^2\right) \\ &= 0.\end{aligned}$$

Las ecuaciones anteriores reciben el nombre de ecuaciones de mínimos cuadrados para estimar los parámetros de una recta. Tras manipular ambas ecuaciones, y por

ser estas lineales en $\hat{\beta}_0$ y $\hat{\beta}_1$, llegamos a

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \frac{\sum_{i=1}^{n} x_i y_i - \frac{1}{n}\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2}, \tag{3.1}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}. \tag{3.2}$$

Teorema 3.1.1. *Los estimadores de mínimos cuadrados de un modelo lineal son insesgados.*

Demostración. Sabemos que

$$\begin{aligned}\hat{\beta}_1 &= \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \\ &= \frac{\sum_{i=1}^{n}(x_i - \bar{x})y_i - \bar{y}\sum_{i=1}^{n}(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}.\end{aligned}$$

Dado que $\sum_{i=1}^{n} x_i - \bar{x} = 0$, podemos simplificar lo anterior en

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})y_i}{\sum_{i=1}^{n}(x_i - \bar{x})^2}.$$

Entonces

$$\begin{aligned}E(\hat{\beta}_1) = E\left[\frac{\sum(x_i - \bar{x})y_i}{\sum_{i=1}^{n}(x_i - \bar{x})^2}\right] &= \frac{\sum(x_i - \bar{x})E(y_i)}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \\ &= \frac{\sum(x_i - \bar{x})(\beta_0 + \beta_1 x_i)}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \\ &= \beta_0 \frac{\sum(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} + \beta_1 \frac{\sum(x_i - \bar{x})x_i}{\sum_{i=1}^{n}(x_i - \bar{x})^2}\end{aligned}$$

Como $\sum(x_i - \bar{x}) = 0$ y $\sum_{i=1}^{n}(x_i - \bar{x})^2 = \sum(x_i - \bar{x})x_i$, tenemos

$$E(\hat{\beta}_1) = 0 + \beta_1 \frac{\sum_{i=1}^{n}(x_i - \bar{x})x_i}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \beta_1$$

Entonces, $\hat{\beta}_1$ es un estimador insesgado de β_1.

Además

$$\begin{aligned} V(\hat{\beta}_1) &= V\left[\frac{\sum(x_i-\bar{x})y_i}{\sum_{i=1}^n(x_1-\bar{x})^2}\right] \\ &= \left[\frac{1}{\sum_{i=1}^n(x_i-\bar{x})^2}\right]^2 \sum V[(x_i-\bar{x})y_i] \\ &= \left[\frac{1}{\sum_{i=1}^n(x_i-\bar{x})^2}\right]^2 \sum (x_i-\bar{x})^2 V(y_i). \end{aligned}$$

Debido a que $V(y_i) = \sigma^2$, para $i = 1, 2, \ldots, n$,

$$V(\hat{\beta}_1) = \frac{\sigma^2}{\sum_{i=1}^n(x_i-\bar{x})^2}.$$

Para encontrar el valor esperado y la varianza de $\hat{\beta}_0$, donde $\hat{\beta}_0 = \overline{Y} - \hat{\beta}_1\overline{x}$ notamos que

$$\begin{aligned} V(\hat{\beta}_0) &= V(\overline{Y} - \hat{\beta}_1\overline{x}) && (3.3) \\ &= V(\overline{Y}) + \overline{x}^2 V(\hat{\beta}_1) - 2\,\mathrm{Cov}(\overline{Y}, \hat{\beta}_1). && (3.4) \end{aligned}$$

Entonces hay que hallar $V(\hat{\beta}_1)$ y $\mathrm{Cov}(\overline{Y}, \hat{\beta}_1)$. Para eso:

$$\begin{aligned} \overline{Y} &= \frac{1}{n}\sum Y_i = \beta_0 + \beta_1\overline{x} + \overline{\varepsilon}. \\ \Rightarrow E(\overline{Y}) &= \beta_0 + \beta_1\overline{x} + E(\overline{\varepsilon}) = \beta_0 + \beta_1\overline{x}, \\ V(\overline{Y}) &= V(\overline{\varepsilon}) = \left(\frac{1}{n}\right) V(\varepsilon_1) = \frac{\sigma^2}{n}. \end{aligned}$$

Para encontrar $\mathrm{Cov}(\overline{Y}, \hat{\beta}_1)$ reescribimos $\hat{\beta}_1 = \sum_{i=1}^n \frac{(x_i-\overline{x})Y_i}{(x_i-\overline{x})^2}$ y como $\sum_{i=1}^n \frac{x_i-\overline{x}}{(x_i-\overline{x})^2} = 0$ entonces

$$\begin{aligned} \mathrm{Cov}(\overline{Y}, \hat{\beta}_1) &= \mathrm{Cov}\left(\frac{1}{n}\sum_{i=1}^n Y_i, \sum_{i=1}^n \frac{(x_i-\overline{x})Y_i}{(x_i-\overline{x})^2}\right) \\ &= \sum_{i\neq j} \frac{1}{n}\cdot\frac{x_i-\overline{x}}{(x_i-\overline{x})^2}\cdot \mathrm{Cov}(Y_i, Y_j) + \sum_{i=1}^n \frac{1}{n}\cdot\frac{x_i-\overline{x}}{(x_i-\overline{x})^2}\cdot V(Y_i), \end{aligned}$$

pero como para $i \neq j$ se cumple que Y_i y Y_j son independientes y además porque $V(Y_i) = \sigma^2$ entonces esto se simplifica a

$$\mathrm{Cov}(\overline{Y}, \hat{\beta}_1) = 0.$$

Retomando $E(\hat{\beta}_0)$ y 3.3 tenemos que

$$\begin{aligned} E(\hat{\beta}_0) &= \beta_0, \\ V(\hat{\beta}_0) &= \frac{\sigma^2}{n} + \overline{x}^2 V(\hat{\beta}_1) \\ &= \frac{\sigma^2}{n} + \frac{\sigma^2 \overline{x}^2}{\sum (x_i - \overline{x})^2}, \end{aligned}$$

mostrando que $\hat{\beta}_0$ y $\hat{\beta}_1$ son estimadores insesgados.

3.1.1. Pruebas de hipótesis para β_i

En muchas ocasiones tendremos que ϵ $N(0, \sigma^2)$, por lo que Y $N(0, \sigma^2)$ y por tanto, los estimadores de mínimos cuadrados, tras un muestreo aleatorio repetido, también tendrán una distribución normal. Algo que nos podríamos preguntar considerando esto es el cómo dar un nivel de confianza para estos estimadores y cómo construir intervalos de confianza sobre sus valores. Usando el valor

$$Z = \frac{\hat{\beta}_i - \beta_{i0}}{\sigma \sqrt{c_{ii}}},$$

donde

$$\begin{aligned} c_{00} &= \frac{\sum_{i=1}^{n} x_i^2}{n \sum_{i=1}^{n} (x_i - \overline{x})^2} \\ c_{11} &= \frac{1}{\sum_{i=1}^{n} (x_i - \overline{x})^2} \end{aligned}$$

podemos hacer una prueba de hipótesis $H_0 : \beta_i = \beta_{i0}$. Con una prueba de dos colas tendríamos una región de rechazo dada por $|z| \geq z_{\alpha/2}$. Generalmente, no conoceremos σ^2, por lo que podemos utilizar $S = \sqrt{SSE/n-2}$ para usar como estadístico de prueba

$$T = \frac{\hat{\beta}_i - \beta_{i0}}{S \sqrt{c_{ii}}}$$

que tiene una distribución t de Student con $n-2$ grados de libertad.

Prueba de hipótesis para β_i

$$H_0 : \beta_i = \beta_{i0}.$$

$$H_a : \begin{cases} \beta_i > \beta_{i0} & \text{(región de rechazo de cola superior),} \\ \beta_i < \beta_{i0} & \text{(región de rechazo de cola inferior),} \\ \beta_i \neq \beta_{i0} & \text{(región de rechazo de dos colas).} \end{cases}$$

Estadístico de prueba: $T = \frac{\hat{\beta}_i - \beta_{i0}}{S\sqrt{c_{ii}}}$.

$$\text{Región de rechazo: } \begin{cases} t > t_\alpha & \text{(alternativa de cola superior),} \\ t < -t_\alpha & \text{(alternativa de cola inferior),} \\ |t| > t_{\alpha/2} & \text{(alternativa de dos colas).} \end{cases}$$

donde

$$c_{00} = \frac{\sum x_i^2}{n\sum_{i=1}^n (x_i - \overline{x})^2}, \quad c_{11} = \frac{1}{\sum_{i=1}^n (x_i - \overline{x})^2}.$$

Con base en esto podríamos construir un intervalo de confianza para β_i así:

$$\beta_i \pm t_{\alpha/2} S\sqrt{c_{ii}}$$

3.1.2. Predicción de valores particulares de Y mediante una regresión lineal simple

Si queremos predecir valores particulares de Y usando la regresión lineal, la situación cambia un poco porque anteriormente, cuando estimamos los parámetros de la regresión, lo que estimábamos eran parámetros, pero ahora queremos estimar una variable aleatoria. Claramente buscaríamos predecir un valor de Y cercano al valor esperado de la variable. Si estamos interesados en el valor de Y cuando $x = x^*$, podemos usar el pronosticador $\hat{Y}^* = \hat{\beta}_0 + \hat{\beta}_1 x^*$ de $\overline{Y}$ y del valor particular Y^* que ocurre cuando $x = x^*$.

Calculamos el error r de la predicción como la diferencia entre el valor observado y

el predicho. Y además

$$\begin{aligned} E(r) &= E(Y^*) - E(\hat{Y}^*) \\ &= \beta_0 + \beta_1 x - (\beta_0 + \beta_1 x) \\ &= 0. \\ V(r) &= V(Y^*) + V(\hat{Y}^*) - 2\,\mathrm{Cov}(Y^*, \hat{Y}^*) \\ &= \sigma^2 + \left(\frac{1}{n} + \frac{(x^* - \overline{x})^2}{\sum (x_i - \overline{x})^2}\right)\sigma^2, \end{aligned}$$

por lo que el error de las predicciones está distribuido normalmente (pues es combinación lineal de variables aleatorias distribuidas normalmente) con media cero y varianza dada en la ecuación anterior. Por tanto,

$$Z = \frac{Y^* - \hat{Y}^*}{\sigma\sqrt{1 + \frac{1}{n} + \frac{(x^* - \bar{x})^2}{S_{xx}}}}$$

tiene una distribución normal estándar. Además, si σ es sustituida por S, se puede demostrar que

$$T = \frac{Y^* - \hat{Y}^*}{S\sqrt{1 + \frac{1}{n} + \frac{(x^* - \bar{x})^2}{S_{xx}}}}$$

tiene una distribución t de Student con $n - 2$ grados de libertad. Usamos este resultado para calcular el límite en el error de predicción. De esta forma, el intervalo de predicción de $100(1 - \alpha)\,\%$ para Y cuando $x = x^*$ es

$$\hat{\beta}_0 + \hat{\beta}_1 x^* \pm t_{\alpha/2} S \sqrt{1 + \frac{1}{n} + \frac{(x^* - \bar{x})^2}{S_{xx}}}.$$

Al hacer las predicciones podemos hacer bandas de confianza para $E(Y)$ usando estos intervalos de confianza, podemos notar cómo las bandas son más estrechas cuando x se acerca a $\overline{x}$

3.1.3. Ajuste de modelo lineal mediante matrices

Supongamos que tenemos el modelo lineal

$$Y = \beta_0 + \beta_1 x_1 + \cdots \beta_n x_n + \epsilon$$

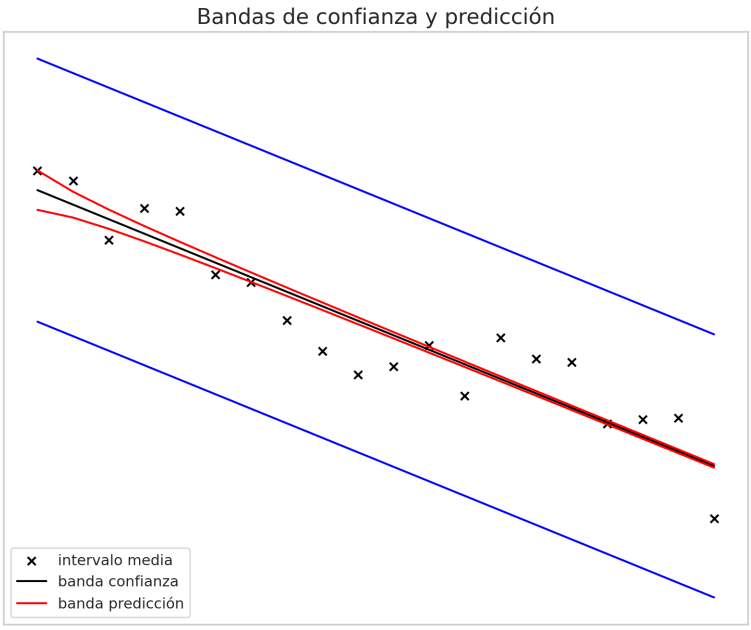


Figura 3.2. Bandas de confianza

y que hacemos n observaciones independientes escribiendo la iésima de estas por

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_n x_{in} + \epsilon_i,$$

donde x_{ij} es la iésima observación para la jésima variable. Definimos las siguientes matrices con $x_0 = 1$:

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} x_{01} & x_{11} & x_{12} & \cdots & x_{1k} \\ x_{02} & x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{0n} & x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}, \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}.$$

Entonces la n ecuaciones de las observaciones se pueden escribir simultáneamente como

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon$$

Particularmente para n observaciones desde un modelo lineal simple de la forma

$$Y = \beta_0 + \beta_1 x + \epsilon,$$

tenemos

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}.$$

Dado que

$$\mathbf{X}'\mathbf{X} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} = \begin{bmatrix} n & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{bmatrix}$$

$$\mathbf{X}'\mathbf{Y} = \begin{bmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \end{bmatrix}, \quad \text{si} \quad \hat{\boldsymbol{\beta}} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{bmatrix}$$

vemos que las ecuaciones de mínimos cuadrados están dadas por $(\mathbf{X}'\mathbf{X})\hat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{Y}$.

$$\text{Por tanto,} \quad \hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}.$$

En resumen,

- *Ecuaciones*: $(\mathbf{X}'\mathbf{X})\hat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{Y}$.
- *Soluciones*: $\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$.

Las propiedades demostradas para el caso lineal simple se extienden para un modelo de regresión lineal múltiple. Esto es,

- $E(\hat{\beta}_i) = \beta_i$ para $i0 \leq i \leq k$
- $V(\hat{\beta}_i) = c_{ii}\sigma^2$, donde c_{ii} es la entrada ii de $(\mathbf{X}'\mathbf{X})^{-1}$.
- $\text{Cov}(\hat{\beta}_i, \hat{\beta}_j) = c_{ij}\sigma^2$, donde c_{ij} es la entrada ij de $(\mathbf{X}'\mathbf{X})^{-1}$.
- Un estimador insesgado de σ^2 es

$$S^2 = \frac{SSE}{n - (k+1)},$$

donde $SSE = \mathbf{Y}'\mathbf{Y} - \hat{\beta}\mathbf{X}\mathbf{Y}$.

Por otro lado, algo importante es que si ϵ_i están distribuidos normalmente, entonces cada β_i también lo estará.

3.1.3.1. Inferencias mediante regresión lineal múltiple

Supongamos que $\beta_0, \beta_1, ..., \beta_n$ son los parámetros del modelo y que queremos hacer una inferencia acerca de

$$a_0\beta_0 + a_1\beta_1 + \cdots + a_k\beta_k,$$

para a_i constantes no necesariamente todas distintas de cero. Definiendo la matriz $(k+1) \times 1$,

$$\mathbf{a} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix},$$

Entonces, como la combinación lineal puede expresarse como $\mathbf{a}\beta$ y

$$a\hat{\boldsymbol{\beta}} = a_0\hat{\beta}_0 + a_1\hat{\beta}_1 + a_2\hat{\beta}_2 + \cdots + a_k\hat{\beta}_k = a\hat{\boldsymbol{\beta}},$$

entonces

$$E(a\hat{\boldsymbol{\beta}}) = E(a_0\hat{\beta}_0 + a_1\hat{\beta}_1 + a_2\hat{\beta}_2 + \cdots + a_k\hat{\beta}_k) = a_0\beta_0 + a_1\beta_1 + a_2\beta_2 + \cdots + a_k\beta_k = a\boldsymbol{\beta}.$$

Análogamente, encontramos la varianza de $a\hat{\boldsymbol{\beta}}$:

$$V(a\hat{\boldsymbol{\beta}}) = [a'(\mathbf{X}'\mathbf{X})^{-1}a]\sigma^2.$$

Además, como $\hat{\beta}_0, ..., \hat{\beta}_k$ están distribuidas normalmente con muestreo repetido y $\mathbf{a}\beta$ es combinación lineal de esto, las combinaciones también están distribuidas normalmente y dado que $E(\mathbf{a}\hat{\beta}) = \mathbf{a}\beta$ y $V(\mathbf{a}\hat{\beta}) = [\mathbf{a}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{a}]\sigma^2$ tenemos que

$$Z = \frac{a\hat{\boldsymbol{\beta}} - a\boldsymbol{\beta}}{\sqrt{V(a\hat{\boldsymbol{\beta}})}} = \frac{a\hat{\boldsymbol{\beta}} - a\boldsymbol{\beta}}{\sigma\sqrt{a'(\mathbf{X}'\mathbf{X})^{-1}a}}$$

tiene una distribución normal estándar y podría emplearse para probar una hipótesis

$$H_0 : a\boldsymbol{\beta} = (a\boldsymbol{\beta})_0$$

cuando $(a\boldsymbol{\beta})_0$ es algún valor especificado. Del mismo modo, un intervalo de confianza de $100(1-\alpha)\,\%$ para $a\boldsymbol{\beta}$ es

$$a\hat{\boldsymbol{\beta}} \pm z_{\alpha/2}\sigma\sqrt{a'(\mathbf{X}'\mathbf{X})^{-1}a}.$$

Además, si sustituimos S por σ, la cantidad

$$T = \frac{a\hat{\boldsymbol{\beta}} - a\boldsymbol{\beta}}{S\sqrt{a'(\mathbf{X}'\mathbf{X})^{-1}a}}$$

tiene una distribución t de Student en muestreo repetitivo, con $[n-(k+1)]$ grados de libertad.

3.1.4. Predicción de un valor particular de Y mediante regresión lineal múltiple

Supongamos que tenemos un modelo de regresión lineal múltiple

$$Y = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + \epsilon$$

y que queremos hallar Y cuando $x_1 = x_1^*, x_2 = x_2^*, ..., x_k = x_k^*$. Entonces vamos a predecir Y con $\beta_0 + \beta_1 x_1^* + \cdots + \beta_k x_k^* = \mathbf{a}\hat{\beta}$, donde

$$\mathbf{a} = \begin{bmatrix} 1 \\ x_1^* \\ x_2^* \\ \vdots \\ x_k^* \end{bmatrix}.$$

Como Y^* y $\hat{Y}^*$ están distribuidas normalmente, el error está distribuido normalmente y

$$E(Y^* - \hat{Y}^*) = 0 \text{ y } V(Y^* - \hat{Y}^*) = \sigma^2[1 + a'(\mathbf{X}'\mathbf{X})^{-1}a]$$

y

$$Z = \frac{Y^* - \hat{Y}^*}{\sigma\sqrt{1 + a'(\mathbf{X}'\mathbf{X})^{-1}a}}$$

tiene una distribución normal estándar. Además, si S es sustituido por σ, se puede demostrar que

$$T = \frac{Y^* - \hat{Y}^*}{S\sqrt{1 + a'(\mathbf{X}'\mathbf{X})^{-1}a}}$$

tiene una distribución t de Student con $[n - (k + 1)]$ grados de libertad. Con lo anterior llegamos a que un intervalo de predicción $100(1 - \alpha)\,\%$ para Y cuando $\mathbf{x}_1 = \mathbf{x}_1^*$, $\mathbf{x}_2 = \mathbf{x}_2^*, \ldots, \mathbf{x}_k = \mathbf{x}_k^*$ está dado por

$$a'\hat{\boldsymbol{\beta}} \pm t_{\alpha/2} S\sqrt{1 + a'(\mathbf{X}'\mathbf{X})^{-1}a},$$

donde $a' = [1, x_1^*, x_2^*, \ldots, x_k^*]$.

3.2. Regresión logística

La regresión logística modela la probabilidad de que un hecho ocurra. Se basa en suponer que las probabilidades logarítmicas o el logaritmo natural de las probabilidades, que es aplicar logaritmo natural al cociente entre la probabilidad de éxito y la de fracaso, puede modelarse mediante una regresión lineal. Es decir, suponemos que, si p es la probabilidad de éxito de un evento, entonces

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k. \tag{3.5}$$

De esta forma, despejando 3.5 llegamos a

$$p = \frac{e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k}}{1 + e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k}}$$

3.2.1. Función de Verosimilitud

La función de verosimilitud en el contexto de la regresión logística expresa la probabilidad de observar los datos dados los parámetros del modelo $\boldsymbol{\beta}$. Para un conjunto de datos compuesto por n observaciones independientes, cada una con una variable de respuesta binaria Y_i y un vector de predictores $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ik})'$, la función

de verosimilitud se define como:

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} P(Y_i = y_i \mid \mathbf{x}_i; \boldsymbol{\beta})$$

Dado que en la regresión logística modelamos la probabilidad $P(Y_i = 1 \mid \mathbf{x}_i) = \pi(\mathbf{x}_i)$, donde:

$$\pi(\mathbf{x}_i) = \frac{1}{1 + \exp(-\mathbf{x}_i'\boldsymbol{\beta})}$$

y $P(Y_i = 0 \mid \mathbf{x}_i) = 1 - \pi(\mathbf{x}_i)$, podemos reescribir la función de verosimilitud como:

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} [\pi(\mathbf{x}_i)]^{y_i} [1 - \pi(\mathbf{x}_i)]^{1-y_i}$$

Esta función de verosimilitud combina las probabilidades de observar $Y_i = 1$ o $Y_i = 0$ en cada observación i, ponderadas por los valores observados y_i.

Para simplificar el cálculo y facilitar la maximización, es común trabajar con la log-verosimilitud, que es el logaritmo natural de la función de verosimilitud. La función de log-verosimilitud se define como:

$$\ell(\boldsymbol{\beta}) = \log L(\boldsymbol{\beta}) = \sum_{i=1}^{n} [y_i \log(\pi(\mathbf{x}_i)) + (1 - y_i) \log(1 - \pi(\mathbf{x}_i))]$$

Sustituyendo la expresión para $\pi(\mathbf{x}_i)$, obtenemos:

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} [y_i \mathbf{x}_i'\boldsymbol{\beta} - \log(1 + \exp(\mathbf{x}_i'\boldsymbol{\beta}))]$$

Maximizar esta función de log-verosimilitud respecto a $\boldsymbol{\beta}$ nos proporciona los estimadores de máxima verosimilitud $\hat{\boldsymbol{\beta}}$, que son los valores de $\boldsymbol{\beta}$ que hacen que la probabilidad de observar los datos dados estos parámetros sea máxima. En la regresión logística, los coeficientes $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_k)'$ se estiman utilizando el método de máxima verosimilitud. Como se mencionó en la sección anterior, este método busca maximizar la función de log-verosimilitud:

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} [y_i \log(\pi(\mathbf{x}_i)) + (1 - y_i) \log(1 - \pi(\mathbf{x}_i))] ,$$

donde $\pi(\mathbf{x}_i)$ es la probabilidad predicha de que $Y_i = 1$ dada la observación $\mathbf{x}_i$:

$$\pi(\mathbf{x}_i) = \frac{1}{1 + \exp(-\mathbf{x}_i'\boldsymbol{\beta})}.$$

Una vez que los coeficientes $\hat{\boldsymbol{\beta}}$ han sido estimados, podemos interpretarlos en términos del logit de la probabilidad:

$$\log\left(\frac{\pi(\mathbf{x})}{1 - \pi(\mathbf{x})}\right) = \mathbf{x}'\hat{\boldsymbol{\beta}}.$$

Cada coeficiente $\hat{\beta}_j$ representa el cambio en el logit de la probabilidad de $Y = 1$ asociado con un aumento unitario en X_j, manteniendo constantes todas las demás variables.

3.2.2. Intervalos de Confianza y Pruebas de Hipótesis

Los intervalos de confianza para los coeficientes $\hat{\beta}_j$ se calculan como:

$$\hat{\beta}_j \pm z_{\alpha/2} \cdot \mathrm{SE}(\hat{\beta}_j),$$

donde $\mathrm{SE}(\hat{\beta}_j)$ es el error estándar de $\hat{\beta}_j$, que puede estimarse a partir de la matriz de covarianza, la cual es la inversa de la matriz Hessiana evaluada en $\hat{\boldsymbol{\beta}}$:

$$\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \mathbf{H}^{-1}(\hat{\boldsymbol{\beta}}).$$

Además, se pueden realizar pruebas de hipótesis para determinar si un coeficiente β_j es significativamente diferente de cero (es decir, si la variable X_j tiene un efecto significativo en la probabilidad de que $Y = 1$). Esto se hace utilizando la estadística de prueba z:

$$z_j = \frac{\hat{\beta}_j}{\mathrm{SE}(\hat{\beta}_j)},$$

que se distribuye aproximadamente como una normal estándar bajo la hipótesis nula $H_0 : \beta_j = 0$.

Si $|z_j| > z_{\alpha/2}$, rechazamos la hipótesis nula y concluimos que el coeficiente β_j es significativamente diferente de cero al nivel de significancia α.

3.3. K-medias

El método de K-medias es un método de agrupamiento en donde, dadas n observaciones, buscamos dividirlas en k grupos en el que cada observación pertenezca al grupo cuya media sea más cercana.

3.3.1. Funcionamiento del método de K-medias

Supongamos que tenemos n observaciones $(x_1, x_2, ..., x_n)$, donde cada observación es un vector en $\mathbf{R^m}$. El objetivo del método es dividir estas observaciones en $k \leq n$ observaciones mediante la minimización de la suma de los cuadrados dentro del clúster. Es decir, queremos encontrar

$$\arg\min_S \sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2 = \arg\min_S \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i,$$

donde μ_i es la media de puntos de S_i.

3.3.2. Algoritmo

La técnica más común es mediante iteraciones. Dado un conjunto de k medias, digamos $m_1^{(1)}, m_2^{(1)}, ..., m_k^{(1)}$, el algoritmo se basa en ir alternando en los siguientes pasos:

- **Asignación:** asignar a cada observación al grupo con media más parecida, lo que minimizaría la distancia entre la observación y la media. Matemáticamente,

  $$S_i^{(t)} = \left\{ x_p : \left\| x_p - m_i^{(t)} \right\|^2 \leq \left\| x_p - m_j^{(t)} \right\|^2 \ \forall j, 1 \leq j \leq k \right\}$$

  y cada x_p es asignada a exactamente un $S^{(t)}$ incluso si podría ser asignada a más de uno.

- **Paso de actualización:** recalcular la media (centroide) para las observaciones asignadas en cada grupo.

Dado que la función objetivo de k-medias es la suma de los cuadrados intragrupo, tenemos una sucesión decreciente de números no negativos, por lo que sabemos que el algoritmo va a converger, aunque no necesariamente el mínimo global.

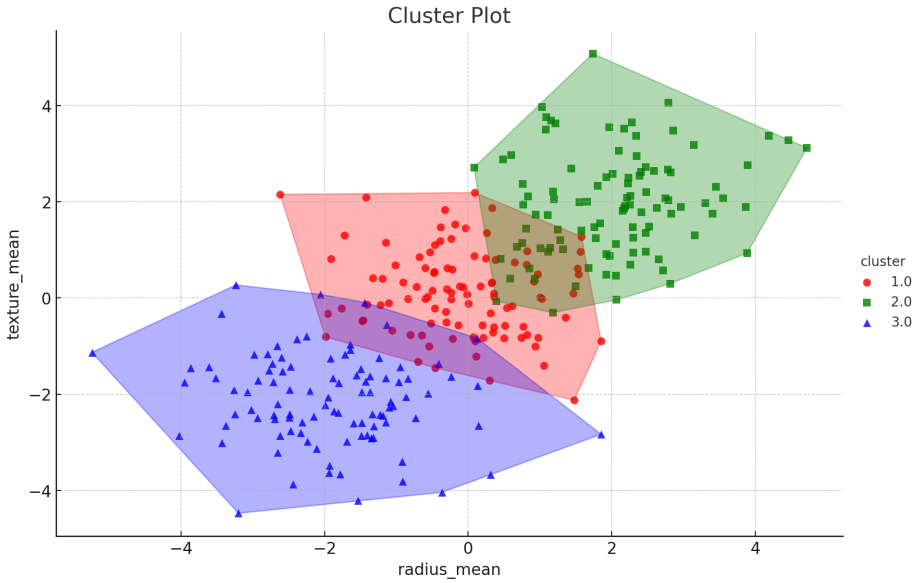


Figura 3.3. Ejemplo de K-medias

4. DESARROLLO DE MODELOS E INTELIGENCIA DE NEGOCIO

4.1. Modelo de regresión lineal múltiple

Se importan las librerías necesarias para la manipulación de datos, conexión a la base de datos, modelado estadístico y visualización.

```
# Importar las libreras necesarias
import pandas as pd
import numpy as np
from sqlalchemy import create_engine
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

Este bloque establece la conexión con una base de datos MySQL utilizando SQLAlchemy y carga varias tablas en DataFrames de pandas para su posterior análisis. Se definen las credenciales de acceso y las tablas específicas que se desean extraer.

```
# Configurar la conexin a la base de datos
host = <HOST>
database = <DATABASE>
user = <USER>
password = <PASSSWORD>
port = <PORT>

# Crear la cadena de conexin
connection_string =
    f'mysql+mysqlconnector://{user}:{password}@{host}:{port}/{database}'

# Crear la conexin
engine = create_engine(connection_string)

# Definir las tablas a cargar
```

```
tablas = ['dim_empresa', 'dim_producto', 'dim_cliente', 'dim_fecha',
    'dim_categoria',
          'fact_venta', 'fact_detalle_venta']

# Cargar las tablas en DataFrames
dataframes = {}
for tabla in tablas:
    query = f'SELECT * FROM {tabla}'
    dataframes[tabla] = pd.read_sql(query, engine)
```

En esta etapa, se realiza la conversión de columnas de fecha al tipo `datetime` para facilitar su manejo. Además, se renombra una columna para evitar conflictos durante la unión de tablas. Posteriormente, se combinan varias tablas relacionadas para crear un DataFrame consolidado que contiene toda la información relevante para el análisis de ventas. Se seleccionan las variables pertinentes para el modelo de regresión y se transforman las variables categóricas en variables dummy. Finalmente, se eliminan las filas con valores faltantes para asegurar la integridad de los datos.

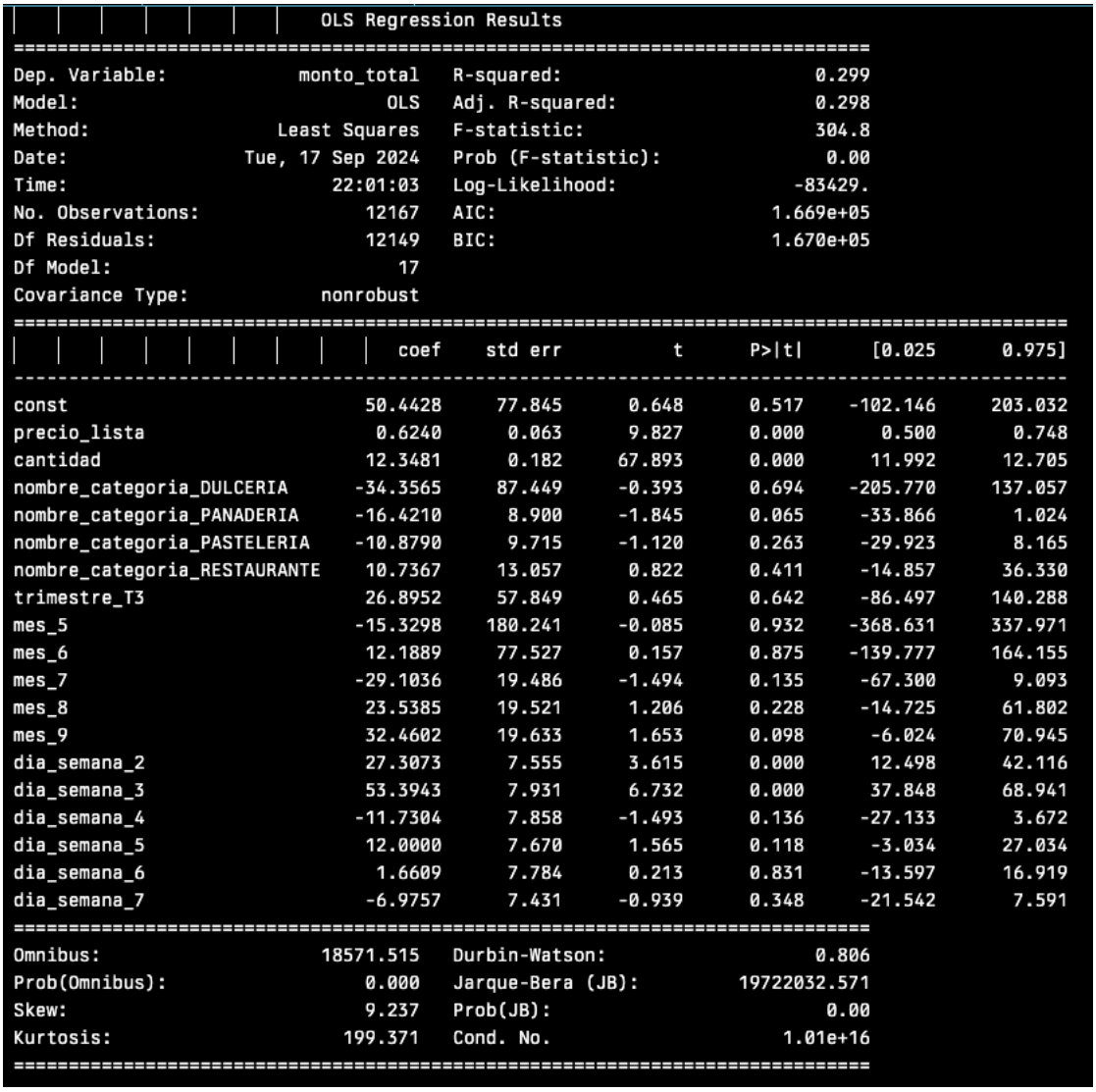
```
# Preprocesamiento inicial
# Convertir fechas a datetime
dataframes['fact_venta']['fecha_venta'] =
    pd.to_datetime(dataframes['fact_venta']['fecha_venta'])
dataframes['dim_fecha']['fecha'] =
    pd.to_datetime(dataframes['dim_fecha']['fecha'])

# Renombrar 'empresa_id' en dim_cliente para evitar conflictos
dataframes['dim_cliente'].rename(columns={'empresa_id': 'empresa_id_cliente'},
    inplace=True)

# Unir las tablas relevantes
df_ventas = pd.merge(dataframes['fact_venta'], dataframes['fact_detalle_venta'],
    on='venta_id', how='left')
df_ventas = pd.merge(df_ventas, dataframes['dim_producto'], on='producto_id',
    how='left', suffixes=('', '_producto'))
df_ventas = pd.merge(df_ventas, dataframes['dim_cliente'], on='cliente_id',
    how='left', suffixes=('', '_cliente'))
df_ventas = pd.merge(df_ventas, dataframes['dim_fecha'], on='fecha_id',
    how='left')
df_ventas = pd.merge(df_ventas, dataframes['dim_empresa'], on='empresa_id',
    how='left', suffixes=('', '_empresa'))
df_ventas = pd.merge(df_ventas, dataframes['dim_categoria'], on='categoria_id',
    how='left')

```

```
# Seleccionar variables para el modelo
df_modelo = df_ventas[['monto_total', 'precio_lista', 'cantidad',
    'nombre_categoria',
                       'tipo', 'ciudad', 'estado_id', 'pais_id', 'mes',
    'dia_semana', 'trimestre',]]

# Convertir variables categricas a dummies
categorical_vars = ['nombre_categoria', 'tipo', 'ciudad', 'estado_id',
    'pais_id', 'trimestre',
                    'mes', 'dia_semana']
df_modelo = pd.get_dummies(df_modelo, columns=categorical_vars, drop_first=True)

# Manejo de valores faltantes
df_modelo.dropna(inplace=True)
```

Aquí se definen las variables dependiente e independientes para el modelo de regresión. Se asegura que todas las columnas sean numéricas y se añaden constantes para el intercepto. Luego, se ajusta el modelo de regresión lineal múltiple utilizando `statsmodels` y se imprime un resumen del modelo, que incluye estadísticas clave como el R-cuadrado, coeficientes y valores p.

```
# Definir variable dependiente e independientes
y = df_modelo['monto_total']
X = df_modelo.drop('monto_total', axis=1)

# Convertir columnas booleanas a enteros
bool_cols = X.select_dtypes(include=['bool']).columns
X[bool_cols] = X[bool_cols].astype(int)

# Asegurarse de que todas las columnas sean numricas
X = X.apply(pd.to_numeric, errors='coerce')
y = pd.to_numeric(y, errors='coerce')

# Eliminar filas con NaN resultantes de la conversin
X.dropna(inplace=True)
y = y.loc[X.index]

#  Aadir  constante para el intercepto
X = sm.add_constant(X)

# Ajustar el modelo de regresin inicial
modelo_sm = sm.OLS(y, X).fit()
print(modelo_sm.summary())
```

A continuación, se presenta la tabla de resultados obtenida del primer ajuste del modelo de regresión lineal múltiple: El modelo presenta un R-cuadrado de 0.299, lo



OLS Regression Results

Dep. Variable:	monto_total	R-squared:	0.299
Model:	OLS	Adj. R-squared:	0.298
Method:	Least Squares	F-statistic:	304.8
Date:	Tue, 17 Sep 2024	Prob (F-statistic):	0.00
Time:	22:01:03	Log-Likelihood:	-83429.
No. Observations:	12167	AIC:	1.669e+05
Df Residuals:	12149	BIC:	1.670e+05
Df Model:	17		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	50.4428	77.845	0.648	0.517	-102.146	203.032
precio_lista	0.6240	0.063	9.827	0.000	0.500	0.748
cantidad	12.3481	0.182	67.893	0.000	11.992	12.705
nombre_categoria_DULCERIA	-34.3565	87.449	-0.393	0.694	-205.770	137.057
nombre_categoria_PANADERIA	-16.4210	8.900	-1.845	0.065	-33.866	1.024
nombre_categoria_PASTELERIA	-10.8790	9.715	-1.120	0.263	-29.923	8.165
nombre_categoria_RESTAURANTE	10.7367	13.057	0.822	0.411	-14.857	36.330
trimestre_T3	26.8952	57.849	0.465	0.642	-86.497	140.288
mes_5	-15.3298	180.241	-0.085	0.932	-368.631	337.971
mes_6	12.1889	77.527	0.157	0.875	-139.777	164.155
mes_7	-29.1036	19.486	-1.494	0.135	-67.300	9.093
mes_8	23.5385	19.521	1.206	0.228	-14.725	61.802
mes_9	32.4602	19.633	1.653	0.098	-6.024	70.945
dia_semana_2	27.3073	7.555	3.615	0.000	12.498	42.116
dia_semana_3	53.3943	7.931	6.732	0.000	37.848	68.941
dia_semana_4	-11.7304	7.858	-1.493	0.136	-27.133	3.672
dia_semana_5	12.0000	7.670	1.565	0.118	-3.034	27.034
dia_semana_6	1.6609	7.784	0.213	0.831	-13.597	16.919
dia_semana_7	-6.9757	7.431	-0.939	0.348	-21.542	7.591

Omnibus:	18571.515	Durbin-Watson:	0.806
Prob(Omnibus):	0.000	Jarque-Bera (JB):	19722032.571
Skew:	9.237	Prob(JB):	0.00
Kurtosis:	199.371	Cond. No.	1.01e+16

Figura 4.1. Tabla de Resultados del Primer Modelo Lineal Múltiple

que indica que aproximadamente el 29.9 % de la variabilidad en el monto total de las ventas es explicada por las variables incluidas en el modelo. A continuación, se analizan las variables más relevantes:

- precio_lista: Con un coeficiente de 0.6240 y un valor p <0.001, esta variable es altamente significativa. Esto sugiere que por cada unidad monetaria adicional en el precio de lista, el monto total de ventas aumenta en promedio en 0.624 unidades monetarias, manteniendo constantes las demás variables.

- cantidad: Presenta un coeficiente de 12.3481 y un valor p <0.001, indicando una alta significancia. Cada unidad adicional vendida incrementa el monto total de ventas en aproximadamente 12.35 unidades monetarias.

- dia_semana_2 y dia_semana_3: Ambas variables son significativas con valores $p < 0.001$ y 0.000, respectivamente. Esto indica que los días 2 y 3 de la semana (probablemente martes y miércoles) tienen un impacto positivo y significativo en las ventas.

Las demás variables, incluyendo las categorías de productos (nombre_categoria_*), los meses (mes_*) y otros días de la semana (dia_semana_*), no muestran una significancia estadística adecuada (valores $p > 0.05$), lo que sugiere que no aportan significativamente al modelo en su forma actual.

Basándose en los resultados obtenidos, se procederá a eliminar las variables que no son estadísticamente significativas para simplificar el modelo y mejorar su interpretabilidad. Las variables identificadas para ser descartadas incluyen:

- Categorías de Productos:
  - nombre_categoria_DULCERIA
  - nombre_categoria_PANADERIA
  - nombre_categoria_PASTELERIA
  - nombre_categoria_RESTAURANTE
- Trimestre:
  - trimestre_T3
- Meses:
  - mes_5
  - mes_6
  - mes_7
  - mes_8
  - mes_9
- Días de la Semana:
  - dia_semana_4

- dia_semana_5
- dia_semana_6
- dia_semana_7

La eliminación de estas variables se realizará en el siguiente paso del análisis para optimizar el modelo, reduciendo la complejidad y eliminando posibles fuentes de multicolinealidad.

En esta etapa, se optimiza el modelo eliminando variables que no son significativas o que presentan multicolinealidad. Se eliminan ciertos meses y categorías de productos que no aportan significativamente al modelo. Además, se agrupan los días de fin de semana en una sola variable para simplificar el modelo. Después de realizar estas modificaciones, se reajusta el modelo de regresión y se imprime un nuevo resumen.

```
# Eliminacin de variables no significativas y multicolineales
# Eliminar meses no significativos
meses_a_eliminar = ['mes_5', 'mes_6', 'mes_7', 'mes_8', 'mes_9']
X = X.drop(columns=meses_a_eliminar, errors='ignore')

# Eliminar categoras no significativas
X = X.drop([
    'nombre_categoria_DULCERIA', 'nombre_categoria_PASTELERIA',
    'nombre_categoria_PANADERIA',
    'nombre_categoria_RESTAURANTE', 'trimestre_T3'
], axis=1, errors='ignore')


# Agrupar das de fin de semana
X['Fin_de_Semana'] = X.get('dia_semana_6', 0) + X.get('dia_semana_7', 0)
dias_a_eliminar = ['dia_semana_4', 'dia_semana_6', 'dia_semana_7',
    'dia_semana_5']
X = X.drop(columns=dias_a_eliminar, errors='ignore')
```

Este código calcula el Factor de Inflación de la Varianza (VIF) para evaluar la multicolinealidad entre las variables independientes. Un VIF alto indica una fuerte correlación con otras variables, lo que puede afectar la estabilidad del modelo.

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Recalcular VIF
X_no_const = X.drop('const', axis=1)

```

```
vif_data = pd.DataFrame()
vif_data["Variable"] = X_no_const.columns
vif_data["VIF"] = [variance_inflation_factor(X_no_const.values, i) for i in
    range(X_no_const.shape[1])]

print(vif_data)
```

Después de eliminar las variables no significativas y ajustar por multicolinealidad, se reajusta el modelo de regresión lineal múltiple y se imprime el resumen, que incluye estadísticas clave como el R-cuadrado, coeficientes y valores p.

```
# Reajustar el modelo de regresin
modelo_sm = sm.OLS(y, X).fit()
print(modelo_sm.summary())
```

A continuación, se presenta la tabla de resultados obtenida del primer ajuste del modelo de regresión lineal múltiple:

OLS Regression Results

Dep. Variable:	monto_total	R-squared:	0.288
Model:	OLS	Adj. R-squared:	0.288
Method:	Least Squares	F-statistic:	983.7
Date:	Thu, 19 Sep 2024	Prob (F-statistic):	0.00
Time:	18:36:23	Log-Likelihood:	-83524.
No. Observations:	12167	AIC:	1.671e+05
Df Residuals:	12161	BIC:	1.671e+05
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	65.4365	3.537	18.501	0.000	58.503	72.370
precio_lista	0.6600	0.056	11.772	0.000	0.550	0.770
cantidad	12.4994	0.182	68.718	0.000	12.143	12.856
dia_semana_2	25.8318	6.354	4.065	0.000	13.377	38.287
dia_semana_3	47.7145	6.786	7.032	0.000	34.414	61.015
Fin_de_Semana	-4.7026	5.017	-0.937	0.349	-14.537	5.132

Omnibus:	18510.191	Durbin-Watson:	0.803
Prob(Omnibus):	0.000	Jarque-Bera (JB):	19135658.443
Skew:	9.184	Prob(JB):	0.00
Kurtosis:	196.413	Cond. No.	177.

Figura 4.2. Tabla de Resultados del Segundo Modelo Lineal Múltiple

Las variables `precio_lista`, `cantidad`, `dia_semana_2` y `dia_semana_3` son estadísticamente significativas al nivel del 1 % (p-valor <0.01), lo que indica que tienen un impacto significativo en el monto total de ventas.

La variable `Fin_de_Semana` no es estadísticamente significativa (p-valor = 0.349), lo que sugiere que no tiene un efecto determinante en las ventas.

El modelo tiene una **F-estadística** de 983.7 con un p-valor de 0.000, lo que indica que el modelo es globalmente significativo y que al menos una de las variables independientes tiene un efecto en la variable dependiente.

En este bloque:

- Se divide el conjunto de datos en entrenamiento y prueba.
- Se ajusta el modelo utilizando el conjunto de entrenamiento.
- Se realizan predicciones en el conjunto de prueba.
- Se calculan métricas de desempeño como el Error Cuadrático Medio (MSE), la Raíz del Error Cuadrático Medio (RMSE) y el R-cuadrado (R^2) para evaluar la precisión del modelo.

```
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

# Ajustar el modelo en el conjunto de entrenamiento
modelo_sm = sm.OLS(y_train, X_train).fit()

# Realizar predicciones en el conjunto de prueba
y_pred = modelo_sm.predict(X_test)

# Calcular mtricas de desempeo
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
print(f'RMSE: {rmse}')
print(f'R-squared: {r2}')
```

Los resultados de la validación cruzada del modelo muestran un RMSE de 251.12 y un R-cuadrado de 0.2695, lo que indica que el modelo tiene una capacidad predictiva moderada. Esto significa que el modelo explica aproximadamente el 26.95 % de la variabilidad en el monto total de ventas y que, en promedio, las predicciones se desvían del valor real en 251.12 unidades monetarias. Aunque el modelo captura

una parte significativa de la variabilidad en las ventas, existe margen para mejorar su precisión mediante la incorporación de variables adicionales, la exploración de modelos más complejos o el refinamiento de las técnicas de modelado utilizadas.

4.1.1. Gráfico de Residuos

Se genera un gráfico de residuos para visualizar la distribución de los errores del modelo. Esto ayuda a evaluar si los residuos están distribuidos aleatoriamente, lo cual es una suposición clave en la regresión lineal.

```
# Grfico de residuos
plt.scatter(y_pred, y_test - y_pred)
plt.xlabel('Predicciones')
plt.ylabel('Residuos')
plt.title('Grfico de Residuos')
plt.axhline(y=0, color='r', linestyle='--')
plt.show()
```

La gráfica de residuos muestra la relación entre las predicciones del modelo y los errores (residuos) en un modelo de regresión lineal múltiple. En un buen ajuste, los residuos deberían distribuirse aleatoriamente alrededor de la línea cero, pero en este caso se observan concentraciones cerca de cero para las predicciones pequeñas y una mayor dispersión para predicciones grandes, lo que sugiere posibles problemas de heterocedasticidad. Además, hay puntos atípicos que indican dificultades del modelo para ajustar ciertos valores extremos, lo que sugiere la necesidad de mejorar el modelo o explorar transformaciones en los datos.

4.1.2. Histograma de Residuos

Se crea un histograma de los residuos para analizar su distribución y verificar la normalidad de los errores, lo que es importante para la validez de las inferencias estadísticas del modelo.

```
# Histograma de residuos
plt.hist(y_test - y_pred, bins=30)
plt.xlabel('Residuos')
plt.ylabel('Frecuencia')
plt.title('Histograma de Residuos')
plt.show()
```

En este gráfico, se observa una fuerte concentración de residuos alrededor de cero, lo cual es esperable si el modelo está ajustado de manera adecuada. Sin embargo,

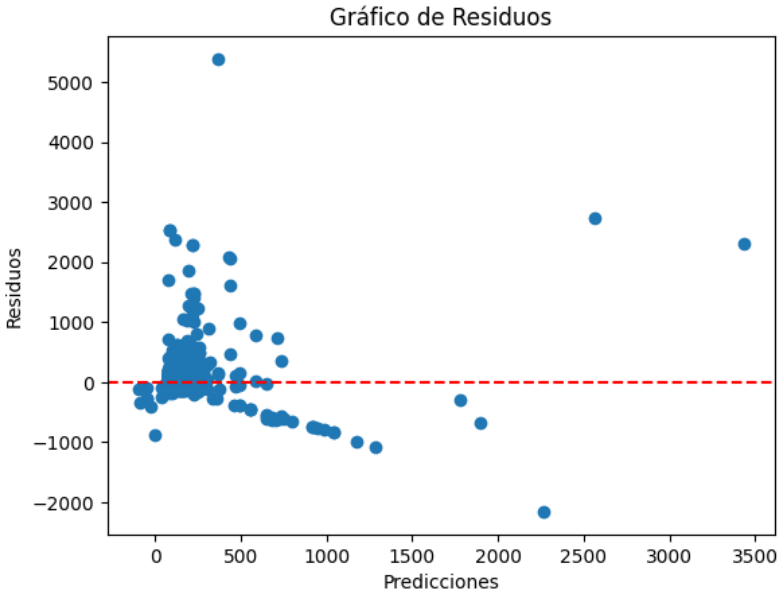


Figura 4.3. Gráfico de Residuos

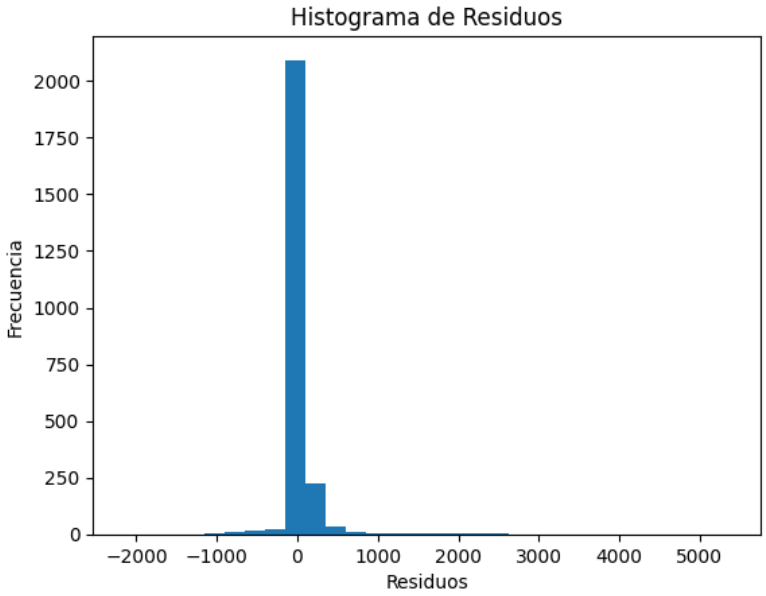


Figura 4.4. Histograma de Residuos

la gran asimetría y la presencia de algunos residuos más grandes, tanto positivos como negativos, indican que existen ciertos puntos donde el modelo tiene dificultades para predecir con precisión, lo que puede deberse a la presencia de valores atípicos o una variabilidad que el modelo no está capturando bien. La distribución sugiere que, si bien la mayoría de los errores son pequeños, existen algunos que se alejan significativamente de cero, lo que podría afectar la calidad general del ajuste.

4.2. Modelo de regresión logística

Para comenzar, se importan los paquetes y módulos necesarios para el análisis de datos, conexión con la base de datos, construcción del modelo y evaluación de su desempeño.

```
# Importar las libreras necesarias
from sqlalchemy import create_engine
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
    classification_report
import warnings
warnings.filterwarnings('ignore')
```

A continuación, se establece la conexión con la base de datos y se extraen las tablas relevantes para el análisis.

```
# Configurar la conexin a la base de datos
host = <HOST>
database = <DATABASE>
user = <USER>
password = <PASSSWORD>
port = <PORT>

# Crear la cadena de conexin
connection_string =
    f'mysql+mysqlconnector://{user}:{password}@{host}:{port}/{database}'

# Crear la conexin
engine = create_engine(connection_string)

# Definir las tablas a cargar
tablas = ['dim_producto',
          'fact_venta', 'fact_detalle_venta']

# Cargar las tablas en DataFrames
dataframes = {}
for tabla in tablas:
    query = f'SELECT * FROM {tabla}'
    dataframes[tabla] = pd.read_sql(query, engine)
```

En este paso, se extraen las columnas esenciales para el análisis. Se convierte la columna `fecha_id` al tipo de dato fecha, lo que permite ordenar y manipular temporalmente los registros. Además, se filtran las ventas y detalles con montos y cantidades positivas para garantizar la calidad de los datos.

```
# Preparacin y limpieza de datos
# Seleccionar las columnas relevantes
productos = dataframes["dim_producto"][["producto_id", "precio_lista",
    "disponible_en_pos"]]
fac_ventas = dataframes["fact_venta"][["fecha_id", "venta_id", "monto_total",
    "empresa_id", "cliente_id"]]
detalle_ventas = dataframes["fact_detalle_venta"]

# Convertir 'fecha_id' a formato datetime
fac_ventas["fecha_id"] = pd.to_datetime(fac_ventas["fecha_id"], format="%Y%m%d")

# Filtrar ventas con monto total positivo y detalles con cantidad positiva
fac_ventas = fac_ventas[fac_ventas["monto_total"] > 0]
detalle_ventas = detalle_ventas[detalle_ventas["cantidad"] > 0]

# Obtener las ltimas tres compras por cliente
fac_ventas = fac_ventas.sort_values(by=["cliente_id", "fecha_id"],
    ascending=[False, False])
fac_ventas = fac_ventas.groupby("cliente_id").head(3).reset_index(drop=True)

# Combinar ventas y detalles de ventas
ventas_totales = pd.merge(fac_ventas, detalle_ventas, on="venta_id", how="inner")

# Agrupar por cliente para obtener media de precios y productos comprados
ventas_totales_agrupado = ventas_totales.groupby("cliente_id").agg(
    media_precios=("precio_unitario", "mean"),
    prods_comprados=("producto_id", list)
).reset_index()
```

A continuación, se obtienen las tres compras más recientes de cada cliente, lo cual es relevante para analizar el comportamiento actual de los clientes.

```
# Construccin del conjunto de datos para el modelo
# Crear una lista de todos los productos
todos_los_productos = productos["producto_id"].unique().tolist()

# Crear un DataFrame vaco para almacenar los datos del modelo
datos = []

```

```
# Iterar sobre cada cliente
for _, row in ventas_totales_agrupado.iterrows():
    cliente = row["cliente_id"]
    media = row["media_precios"]
    prods_comprados = row["prods_comprados"]

    for prod in todos_los_productos:
        # Verificar si el cliente ha comprado el producto
        lo_ha_comprado = 1 if prod in prods_comprados else 0

        # Obtener informacin del producto
        producto_info = productos[productos["producto_id"] == prod].iloc[0]
        precio = producto_info["precio_lista"]
        disponible_pos = producto_info["disponible_en_pos"]

        # Verificar si el precio es menor o igual a la media de compras del
    cliente
        precio_menor_igual = 1 if precio <= media else 0

        # Agregar la fila al conjunto de datos
        datos.append([
            cliente, prod, lo_ha_comprado, precio_menor_igual,
            precio, media, disponible_pos
        ])

# Crear el DataFrame final para el modelo
df_modelo = pd.DataFrame(datos, columns=[
    "cliente_id", "producto_id", "lo_ha_comprado",
    "precio_menor_igual", "precio", "media_compras", "disponible_pos"
])
```

Se seleccionaron las columnas relevantes de las tablas de productos, ventas y detalles de ventas, convirtiendo las fechas al formato adecuado y filtrando registros con montos y cantidades positivas para asegurar la calidad de los datos. Se identificaron las tres últimas compras de cada cliente ordenando las ventas por fecha, se combinaron las ventas con sus detalles correspondientes y se agruparon los datos por cliente para calcular la media de precios unitarios y compilar una lista de productos adquiridos, proporcionando una visión general del comportamiento de compra de cada cliente.

```
# Preparacin y limpieza de datos
# Seleccionar las columnas relevantes
productos = dataframes["dim_producto"][["producto_id", "precio_lista",
    "disponible_en_pos"]]
```

```
fac_ventas = dataframes["fact_venta"][["fecha_id", "venta_id", "monto_total",
    "empresa_id", "cliente_id"]]
detalle_ventas = dataframes["fact_detalle_venta"]

# Convertir 'fecha_id' a formato datetime
fac_ventas["fecha_id"] = pd.to_datetime(fac_ventas["fecha_id"], format="%Y%m%d")

# Filtrar ventas con monto total positivo y detalles con cantidad positiva
fac_ventas = fac_ventas[fac_ventas["monto_total"] > 0]
detalle_ventas = detalle_ventas[detalle_ventas["cantidad"] > 0]

# Obtener las ltimas tres compras por cliente
fac_ventas = fac_ventas.sort_values(by=["cliente_id", "fecha_id"],
    ascending=[False, False])
fac_ventas = fac_ventas.groupby("cliente_id").head(3).reset_index(drop=True)

# Combinar ventas y detalles de ventas
ventas_totales = pd.merge(fac_ventas, detalle_ventas, on="venta_id", how="inner")

# Agrupar por cliente para obtener media de precios y productos comprados
ventas_totales_agrupado = ventas_totales.groupby("cliente_id").agg(
    media_precios=("precio_unitario", "mean"),
    prods_comprados=("producto_id", list)
).reset_index()
```

Se elaboró un conjunto de datos que relaciona a cada cliente con todos los productos disponibles, indicando si el cliente ha adquirido cada producto y recopilando características relevantes como el precio de lista y la disponibilidad en el punto de venta. Además, se calculó una variable indicadora que refleja si el precio del producto es menor o igual a la media de precios de las compras del cliente, proporcionando todas las variables necesarias para entrenar el modelo de regresión logística.

```
# Construccin del conjunto de datos para el modelo
# Crear una lista de todos los productos
todos_los_productos = productos["producto_id"].unique().tolist()

# Crear un DataFrame vaco para almacenar los datos del modelo
datos = []

# Iterar sobre cada cliente
for _, row in ventas_totales_agrupado.iterrows():
    cliente = row["cliente_id"]
    media = row["media_precios"]
```

```
    prods_comprados = row["prods_comprados"]

    for prod in todos_los_productos:
        # Verificar si el cliente ha comprado el producto
        lo_ha_comprado = 1 if prod in prods_comprados else 0

        # Obtener informacin del producto
        producto_info = productos[productos["producto_id"] == prod].iloc[0]
        precio = producto_info["precio_lista"]
        disponible_pos = producto_info["disponible_en_pos"]

        # Verificar si el precio es menor o igual a la media de compras del
    cliente
        precio_menor_igual = 1 if precio <= media else 0

        # Agregar la fila al conjunto de datos
        datos.append([
            cliente, prod, lo_ha_comprado, precio_menor_igual,
            precio, media, disponible_pos
        ])

# Crear el DataFrame final para el modelo
df_modelo = pd.DataFrame(datos, columns=[
    "cliente_id", "producto_id", "lo_ha_comprado",
    "precio_menor_igual", "precio", "media_compras", "disponible_pos"
])
```

Se definieron las variables predictoras, que incluyen indicadores sobre el precio en relación con la media de compras del cliente, el precio del producto, la media de precios de las compras del cliente y la disponibilidad del producto en el punto de venta, siendo la variable objetivo binaria si el cliente ha comprado el producto. El conjunto de datos se dividió en conjuntos de entrenamiento y prueba, se entrenó un modelo de regresión logística y se evaluó su desempeño utilizando métricas como exactitud, matriz de confusión y reporte de clasificación que incluye precisión, exhaustividad y puntuación F1.

```
# Modelo de Regresin Logstica
# Definir las variables predictoras y la variable objetivo
X = df_modelo[['precio_menor_igual', 'precio', 'media_compras',
    'disponible_pos']]
y = df_modelo['lo_ha_comprado']

# Dividir el conjunto de datos en entrenamiento y prueba
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Crear y entrenar el modelo
model = LogisticRegression()
model.fit(X_train, y_train)

# Realizar predicciones en el conjunto de prueba
y_pred = model.predict(X_test)

# Evaluar el modelo
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f'Accuracy del modelo: {accuracy:.2f}')
print('Matriz de confusin:')
print(conf_matrix)

# Obtener reporte de clasificacin
report = classification_report(y_test, y_pred)
print('Reporte de clasificacin:')
print(report)
```

Los resultados muestran que el modelo puede predecir con cierta precisión si un cliente comprará un producto, con métricas que indican un desempeño aceptable aunque identifican áreas de mejora. La matriz de confusión revela la distribución de aciertos y errores, evidenciando posibles desequilibrios en las clases y errores sistemáticos. Además, el análisis de las probabilidades de predicción ofrece una comprensión más profunda del comportamiento del modelo y su confianza en las predicciones, lo que permite identificar casos que requieren un análisis más detallado o la aplicación de reglas de negocio adicionales.

```
# Obtener las probabilidades de prediccin
y_prob = model.predict_proba(X_test)

# Mostrar las probabilidades junto con las predicciones y valores reales
prob_df = pd.DataFrame({
    'Prob_No_Comprado': y_prob[:, 0],
    'Prob_Comprado': y_prob[:, 1],
    'Prediccin': y_pred,
    'Real': y_test.values
})
```

```
11
12 prob_df.head()
```

4.3. Modelo de clusterización de K-medias

Este bloque de código importa bibliotecas como `pandas` y `numpy` para la manipulación de datos, `sqlalchemy` para la conexión a la base de datos, `matplotlib` para la generación de gráficos, y componentes de `scikit-learn` para el modelado y preprocesamiento necesarios en la implementación del algoritmo K-medias.

```
1  # Importar las libreras necesarias
2  from sqlalchemy import create_engine
3  import pandas as pd
4  import numpy as np
5  import matplotlib.pyplot as plt
6  from sklearn.cluster import KMeans
7  from sklearn.preprocessing import StandardScaler
8  from sklearn.decomposition import PCA
9  import warnings
10 warnings.filterwarnings("ignore")
```

A continuación, se establece la conexión con la base de datos MySQL utilizando SQLAlchemy y se cargan las tablas relevantes en *DataFrames* de pandas para su posterior análisis.

```
1  # Configurar la conexin a la base de datos
2  host = <HOST>
3  database = <DATABASE>
4  user = <USER>
5  password = <PASSSWORD>
6  port = <PORT>
7
8  # Crear la cadena de conexin
9  connection_string =
       f'mysql+mysqlconnector://{user}:{password}@{host}:{port}/{database}'
10
11 # Crear la conexin
12 engine = create_engine(connection_string)
13
14 # Definir las tablas a cargar
15 tablas = ['dim_producto', 'fact_venta', 'fact_detalle_venta']
16
```

```
# Cargar las tablas en DataFrames
dataframes = {}
for tabla in tablas:
    query = f'SELECT * FROM {tabla}'
    dataframes[tabla] = pd.read_sql(query, engine)
```

Posteriormente, se seleccionan las columnas de interés de cada tabla y se crean nuevos *DataFrames* que contienen únicamente la información necesaria para el modelo de clusterización. En este paso, se extraen las columnas esenciales para el análisis, como `precio_lista`, Se aplica el algoritmo K-medias para agrupar los productos en clústeres y se reduce la dimensionalidad de los datos para facilitar la visualización. Se inicializa el modelo K-medias especificando 4 clústeres y se ajusta a los datos escalados. Los productos se etiquetan con el clúster al que pertenecen. Para visualizar los clústeres en dos dimensiones, se utiliza `PCA` para reducir la dimensionalidad, obteniendo los dos principales componentes que capturan la mayor variabilidad en los datos.

```
# Aplicar K-medias para agrupar los productos
kmeans = KMeans(n_clusters=4, random_state=0)
kmeans.fit(df_escalado)
productos["Cluster"] = kmeans.labels_

# Reducir la dimensionalidad con PCA para visualizacin
pca = PCA(n_components=2)
componentes_principales = pca.fit_transform(df_escalado)
df_pca = pd.DataFrame(data=componentes_principales, columns=["Componente 1",
    "Componente 2"])
df_pca["Cluster"] = kmeans.labels_
```

Se visualizan los clústeres resultantes en un gráfico de dispersión para interpretar gráficamente la segmentación de los productos. El código genera una gráfica donde cada punto representa un producto en el espacio definido por los dos primeros componentes principales, y los colores indican el clúster al que pertenece cada producto. Esto permite identificar visualmente patrones y agrupaciones en los datos.

```
# Visualizar los clsteres en un grfico de dispersin
plt.figure(figsize=(10,6))
plt.scatter(df_pca["Componente 1"], df_pca["Componente 2"], c=df_pca["Cluster"],
    cmap="viridis")
plt.xlabel("Componente Principal 1")
plt.ylabel("Componente Principal 2")
plt.title("Clsteres basados en Componentes Principales")
plt.show()
```

Finalmente, se analizan las características de cada clúster mediante estadísticas descriptivas, lo que facilita la interpretación y comprensión de los grupos formados. En este bloque, se itera sobre cada clúster y se imprimen las estadísticas descriptivas de los productos pertenecientes a él. Esto incluye medidas como la media, desviación estándar, valores mínimos y máximos de las variables, permitiendo identificar las principales características que definen a cada grupo y entender mejor las diferencias

entre ellos.

```
# Mostrar estadsticas descriptivas de cada clster
for i in range(4):
    print(f"Estadsticas descriptivas para el Clster {i}:")
    print(productos[productos["Cluster"] == i].describe())
    print("-" * 67)
```

5. RESULTADOS DE LOS MODELOS

En el desarrollo de este proyecto se implementaron tres modelos distintos con el objetivo de analizar y extraer *insights* valiosos de los datos disponibles: un modelo de regresión lineal múltiple, un modelo de regresión logística y un modelo de clusterización mediante K-medias. A continuación, se presentan las conclusiones derivadas de cada uno de estos modelos.

5.1. Modelo de Regresión Lineal Múltiple

El modelo de regresión lineal múltiple se utilizó para identificar y cuantificar el impacto de diversas variables en el monto total de las ventas. Los resultados obtenidos indican que el modelo explica aproximadamente el 28.8 % de la variabilidad en el monto total de ventas, como lo demuestra el coeficiente de determinación R^2 de 0.288. Aunque este valor sugiere que existen otros factores no incluidos en el modelo que afectan significativamente las ventas, proporciona una base para entender algunas relaciones clave.

Las variables que resultaron ser estadísticamente significativas incluyen:

- **Precio de lista (`precio_lista`):** Con un coeficiente positivo y significativo ($\beta = 0.6600$, $p < 0.001$), indica que un aumento en el precio de lista se asocia con un incremento en el monto total de ventas. Esto puede deberse a que productos con precios más altos generan mayores ingresos por unidad vendida.

- **Cantidad vendida (`cantidad`):** Esta variable tiene el mayor impacto en el modelo ($\beta = 12.4994$, $p < 0.001$), lo que es coherente con la lógica de que vender más unidades aumenta el monto total de ventas.

- **Días de la semana (`dia_semana_2` y `dia_semana_3`):** Los coeficientes positivos y significativos para el martes y miércoles sugieren que las ventas tienden a ser mayores en estos días comparados con el día de referencia (lunes).

La variable `Fin_de_Semana` no resultó ser significativa ($p = 0.349$), indicando que, en este modelo, los fines de semana no tienen un impacto estadísticamente significativo en el monto total de ventas.

El análisis de los residuos reveló posibles problemas de heterocedasticidad y la presencia de valores atípicos, lo que sugiere que el modelo podría beneficiarse de refinamientos adicionales, como la inclusión de variables adicionales, transformaciones de variables existentes o el uso de modelos no lineales.

5.2. Modelo de Regresión Logística

El modelo de regresión logística se implementó con el objetivo de predecir la probabilidad de que un cliente compre un producto determinado, basándose en características como si el precio es menor o igual a su promedio de compra, el precio del producto, la media de sus compras y la disponibilidad del producto en el punto de venta.

Los resultados muestran una exactitud (accuracy) del 99 %; sin embargo, al analizar la matriz de confusión y el reporte de clasificación, se observa que el modelo tiene una capacidad predictiva limitada para la clase positiva (clientes que compran el producto). Específicamente:

- De los 198 casos reales donde el cliente compró el producto, el modelo no identificó correctamente ninguno (recall de 0.00 para la clase 1).
- La precisión (precision) y el puntaje F1 para la clase 1 son también 0.00, lo que indica que el modelo no está capturando adecuadamente esta clase.

Este desempeño se debe probablemente a un problema de desbalanceo de clases, donde la clase negativa (clientes que no compran el producto) es abrumadoramente mayoritaria. Esto provoca que el modelo aprenda a predecir siempre la clase mayoritaria para maximizar la exactitud global, pero a costa de no identificar correctamente los casos positivos.

Para mejorar el modelo, se podrían considerar las siguientes acciones:

- **Balancear el conjunto de datos:** Mediante técnicas como submuestreo de la clase mayoritaria, sobremuestreo de la clase minoritaria o el uso de algoritmos como SMOTE para generar ejemplos sintéticos.
- **Utilizar métricas adecuadas:** En lugar de centrarse en la exactitud, utilizar métricas como el área bajo la curva ROC (AUC-ROC), precisión, recall y puntaje F1 para evaluar mejor el desempeño en la clase minoritaria.
- **Implementar modelos más complejos:** Como árboles de decisión, random forests o modelos basados en gradiente boosting, que pueden manejar mejor el desbalanceo de clases.

5.3. Modelo de Clusterización K-medias

El modelo de clusterización K-medias se utilizó para segmentar los productos en grupos homogéneos basados en características como el precio de lista, categoría, unidad de medida y el promedio de unidades compradas por factura. Se identificaron cuatro clústeres, cuyas características principales son:

- **Clúster 0:** Compuesto por 72 productos de categorías 7 a 9, con un precio de lista promedio de 20.85 y un promedio de compra por factura de aproximadamente 0.97 unidades. Estos productos parecen ser de precio medio y de categorías específicas, con una baja cantidad comprada por factura.

- **Clúster 1:** Contiene un único producto con un precio de lista de 40.0, perteneciente a la categoría 1 y unidad de medida 4. Este clúster singular podría indicar un producto atípico o exclusivo que no encaja en los demás grupos.
- **Clúster 2:** Incluye 86 productos de la categoría 5, con un precio de lista promedio de 22.68 y un promedio de compra por factura de 13.45 unidades. Estos productos tienen un volumen de venta alto en términos de unidades por factura, lo que sugiere que son productos de alta rotación.
- **Clúster 3:** Con 108 productos de categorías 5 y 6, presentan un precio de lista promedio más alto (39.98) y un promedio de compra por factura de 2.87 unidades. Estos productos podrían ser de mayor valor y comprados en cantidades moderadas.

El análisis de estos clústeres permite a la empresa:

- **Personalizar estrategias de marketing:** Dirigir campañas específicas para cada segmento de productos, optimizando recursos y mensajes.
- **Optimizar el inventario:** Ajustar los niveles de stock según la demanda esperada de cada clúster, reduciendo costos de almacenamiento y evitando quiebres de stock.
- **Definir políticas de precios:** Establecer precios competitivos y promociones basadas en las características y comportamiento de venta de cada grupo de productos.

5.4. Consideraciones Finales

Los modelos desarrollados proporcionan *insights* valiosos sobre las ventas y el comportamiento de los clientes y productos. Sin embargo, existen limitaciones que deben ser abordadas en futuros trabajos:

- **Datos adicionales:** La incorporación de más variables, como información demográfica de los clientes, tendencias de mercado o datos temporales más detallados, podría mejorar la precisión y utilidad de los modelos.
- **Mejora de los modelos:** Experimentar con diferentes algoritmos y técnicas de modelado, como redes neuronales, modelos no lineales o métodos ensemble, podría capturar relaciones más complejas en los datos.
- **Tratamiento del desbalanceo de clases:** Especialmente en el modelo de regresión logística, es fundamental abordar el desbalanceo para mejorar la capacidad predictiva en la clase minoritaria.
- **Validación y pruebas adicionales:** Realizar validaciones cruzadas y pruebas con conjuntos de datos independientes para evaluar la generalización de los modelos.

6. APOYO EN BUSINESS INTELLIGENCE

6.1. Implementación de un ERP/CRM (Odoo)

Antes de implementar Odoo, Tres Cielos enfrentaba ineficiencias operativas, retrasos en pedidos, errores de inventario y una experiencia inconsistente para los clientes debido a la fragmentación de la información en múltiples plataformas y sistemas manuales. Para abordar estos problemas, se seleccionó Odoo tras evaluar proveedores en funcionalidad, costo, soporte técnico, escalabilidad e integración con herramientas existentes.

La implementación se realizó en fases: definición de requisitos, diseño del sistema, migración de datos y personalización de módulos. Se capacitaron a todos los empleados en el uso básico y en módulos clave como ventas, inventarios y CRM. Tras pruebas exhaustivas, se lanzó el sistema con soporte técnico continuo.

Los beneficios incluyen la integración de procesos empresariales en una plataforma unificada, mejorando la colaboración y reduciendo redundancias. La centralización de la información optimizó la comunicación interna y la eficiencia operativa. Además, el acceso a datos en tiempo real y reportes detallados facilitó decisiones estratégicas, mientras que el módulo CRM mejoró la gestión de relaciones con clientes, incrementando su satisfacción y fidelización. Odoo también permite la incorporación de nuevas funcionalidades para soportar el crecimiento futuro de la empresa.

6.2. Unificación de Canales Telefónicos

Tres Cielos utilizaba múltiples líneas telefónicas para gestionar pedidos y consultas, lo que causaba duplicaciones, errores y tiempos de espera prolongados, afectando la experiencia del cliente y la eficiencia operativa. Para resolver esto, se consolidaron todas las líneas en una única línea centralizada.

Este cambio se comunicó a los clientes mediante una campaña que incluyó correos electrónicos, redes sociales, actualizaciones en el sitio web y folletos en puntos de venta. Además, el personal de atención al cliente realizó llamadas directas a clientes clave para explicar el cambio.

Se implementó un sistema de monitoreo para supervisar el uso de la nueva línea

y recopilar retroalimentación, permitiendo resolver rápidamente cualquier inconveniente. Los beneficios incluyen una gestión más eficiente de los pedidos, optimización de recursos al manejar un mayor volumen de llamadas sin aumentar significativamente el equipo, reducción de tiempos de espera y mejora en la precisión de los pedidos. Esto ha incrementado la satisfacción y fidelización de los clientes y mejorado la coordinación interna entre los equipos de ventas y logística.

6.3. Capacitación en Herramientas de Inteligencia Artificial (IA)

Para innovar y optimizar procesos, Tres Cielos incorporó herramientas de inteligencia artificial en áreas como generación de contenido para redes sociales, asesorías legales, atención al cliente y creación de contenido visual. Sin embargo, el personal necesitaba capacitación para utilizar estas tecnologías eficazmente.

Se evaluaron las necesidades y se seleccionaron herramientas con interfaces amigables y buena documentación. La capacitación incluyó manuales, guías, talleres interactivos y proyectos piloto para aplicar lo aprendido. Se realizaron evaluaciones periódicas para medir el progreso y ajustar las capacitaciones según fuera necesario.

Las aplicaciones implementadas incluyeron ChatGPT para generación de contenido, chatbots especializados en derecho laboral y atención al cliente, y DALL-E para creación de contenido visual. Los beneficios incluyen mayor eficiencia y creatividad del equipo, presencia en línea más profesional, reducción de costos operativos, mejora en la atención al cliente y la capacidad de escalar operaciones sin incrementar proporcionalmente los costos. Además, se fomentó una cultura de innovación dentro de la organización.

6.4. Asesorías en la Interpretación de Tableros del ERP

Aunque Tres Cielos contaba con tableros de ingresos y egresos en Odoo, el personal tenía dificultades para interpretarlos y analizarlos eficazmente, limitando la toma de decisiones informadas y subutilizando las funcionalidades del ERP.

Las asesorías se enfocaron en capacitar al personal para interpretar y utilizar los tableros del ERP. Se desarrollaron manuales y guías sobre la lectura de gráficas y la interpretación de datos financieros y operativos. Las asesorías incluyeron talleres interactivos y ejercicios prácticos, así como prácticas supervisadas y evaluaciones periódicas para medir el progreso.

Durante las asesorías, se enseñó a interpretar diferentes tipos de gráficas y a desglosar ingresos y egresos para identificar fuentes principales de ingresos, áreas de mayor gasto y tendencias financieras. Se promovió una cultura de toma de decisiones basada en datos, enseñando a identificar oportunidades de mejora, realizar planificación estratégica y monitorear indicadores clave de desempeño (KPIs).

Los resultados incluyen una mejor comprensión de la situación financiera y operativa de la empresa, facilitando decisiones más acertadas y estratégicas. Además, se mejoró la asignación de recursos, aumentando la eficiencia y reduciendo costos innecesarios. La transparencia en la gestión ha aumentado, proporcionando mayor control sobre las operaciones empresariales. Finalmente, el personal se siente más capacitado y empoderado para utilizar los datos en su trabajo diario, incrementando la moral y productividad y permitiendo una respuesta ágil a los desafíos del negocio.

CONCLUSIONES

1. La creación del *Data Warehouse* ha centralizado y estructurado la información clave del negocio, facilitando el acceso y análisis de los datos. La implementación efectiva del proceso ETL garantiza que los datos sean consistentes, íntegros y actualizados, lo cual es esencial para respaldar decisiones informadas y desarrollar modelos de inteligencia de negocio robustos.

2. El desarrollo e implementación de tres modelos analíticos (regresión lineal múltiple, regresión logística y clusterización K-medias) ha proporcionado valiosos *insights* para la empresa. El modelo de regresión lineal múltiple reveló que el precio de lista y la cantidad vendida son variables significativamente positivas que impactan en el monto total de ventas, explicando el 28.8 % de la variabilidad. Además, se observó que las ventas tienden a ser mayores los martes y miércoles. La presencia de heterocedasticidad y valores atípicos sugiere la necesidad de refinar el modelo incluyendo variables adicionales o utilizando enfoques no lineales. Por otro lado, el modelo de regresión logística mostró una alta exactitud del 99 %, pero su capacidad predictiva para identificar clientes que compran el producto es nula debido al desbalanceo de clases, lo que resalta la importancia de equilibrar el conjunto de datos para mejorar la identificación de clientes potenciales. Finalmente, la segmentación de productos en cuatro clústeres homogéneos permitió identificar diferentes patrones de comportamiento y características de los productos, facilitando la personalización de estrategias de marketing, optimización de inventarios y definición de políticas de precios específicas para cada segmento, mejorando la eficiencia operativa y la satisfacción del cliente.

3. La implementación de Odoo ha integrado procesos empresariales en una plataforma unificada, mejorando la colaboración y eficiencia operativa. La unificación de canales telefónicos simplificó la atención al cliente y optimizó la gestión de pedidos. Además, la capacitación en herramientas de inteligencia artificial, como ChatGPT y DALL-E, ha incrementado la eficiencia y creatividad del equipo, mejorando la generación de contenido y la atención al cliente.

RECOMENDACIONES

1. Implementar políticas y procedimientos rigurosos para asegurar la calidad, consistencia y seguridad de los datos. Esto incluye procesos de limpieza de datos, validación continua y monitoreo de la integridad de los datos en el *Data Warehouse*. Una buena gobernanza de datos garantiza que los modelos analíticos se basen en información precisa y confiable, mejorando la efectividad de las decisiones empresariales.

2. Incorporar variables adicionales que influyen en las ventas y el comportamiento del cliente, como información demográfica, preferencias, historial de interacciones y métricas de satisfacción. Esto proporcionará *insights* más profundos y aumentará la capacidad predictiva de los modelos.

3. Continuar aprovechando las funcionalidades del ERP/CRM implementado y explorar técnicas avanzadas de análisis de datos, incluyendo modelos de aprendizaje automático más sofisticados. Promover la capacitación del personal en el uso de herramientas de inteligencia artificial y en la interpretación de datos para optimizar procesos, mejorar la toma de decisiones y adaptar estrategias a las necesidades específicas de cada segmento de mercado.

BIBLIOGRAFÍA

[1] M. A. ABD ELMONEM, E. S. NASR, and M. H. GEITH, *Benefits and challenges of cloud ERP systems–A systematic literature review*, *Future Computing and Informatics Journal*, vol. 1, no. 1-2, pp. 1–9, 2016.

[2] M. AL-MASHARI, S. K. GHANI, and W. AL-RASHID, *A study of the critical success factors of ERP implementation in developing countries*, *International Journal of Internet and Enterprise Management*, vol. 4, no 1, pp. 68–95, 2006.

[3] H. M. BEHESHTI, *What managers should know about ERP/ERP II*, *Management Research News*, vol. 29, no 4, pp. 184–193, 2006.

[4] Benchmarking Partners, *ERP's Second Wave: Maximizing the Value of ERP-Enabled Processes*, Cambridge, MA, 1998.

[5] H. BIDGOLI, *Encyclopedia of Information Systems: EJ*, Academic Press, 2003.

[6] P. BINGI, M. K. SHARMA, and J. K. GODLA, *Critical issues affecting an ERP implementation*, *Inf. Syst. Manag.*, vol. 16, no 3, pp. 7–14, 1999.

[7] S. BUCKHOUT, E. FREY, and J. NEMEC, JR, *Making ERP succeed: turning fear into promise*, *IEEE Engineering Management Review*, pp. 116–123, 1999.

[8] J. Carr, *ERP Implementation Survey Shows Real Facts About ERP*, Ultra Consultants, 2023. [Online]. Available: `https://ultraconsultants.com/erp-software-blog/erp-implementation-survey/`