



# Tecnológico de Monterrey

Instituto Tecnológico y de Estudios  
Superiores de Monterrey

## E2: Reporte Final del Reto

Adrian Pineda Sánchez	A00833147
Alexis Daniel Leyva Yáñez	A01770308
Kevin Antonio González Díaz	A01338316
Luis Maximiliano López Ramírez	A00833321

Análisis de ciencia de datos TC2004B.201

(Gpo 201)

Link del código en github:

[https://github.com/kevingonzal/cienciaDatos/blob/main/Entrega3\\_final%20\(1\).ipynb](https://github.com/kevingonzal/cienciaDatos/blob/main/Entrega3_final%20(1).ipynb)

Profesores:

Felipe Castillo Rendón  
María de los Ángeles Constantino González

Socio Formador: Ternium

Monterrey, Nuevo León.

06 de mayo de 2023

# **ÍNDICE**

**1) Descripción del Negocio**

**2) Entendimiento de los datos**

**3) Generación y Evaluación de Modelos de Aprendizaje Automático**

**4) Selección y Despliegue**

**5) Conclusiones y Recomendaciones**

**6) Fuentes bibliográficas en formato APA.**

**7) Anexos (Individual)**

# Etapa 1: Descripción del Negocio

## 1.1 OBJETIVOS DEL NEGOCIO

### Datos generales de la empresa



El socio formador, **Ternium**, es una de las mayores empresas con presencia en América Latina en la industria de la producción de acero.

Los cuales se especializan en la creación y procesamiento de una amplia variedad de productos de acero (laminado en caliente, laminados en frío, aceros galvanizados, hojalata, barras y alambrón, tubos y perfiles, estructuras metálicas prediseñadas) impactando en los mercados Automotriz, construcción, línea blanca, bienes de capital, envases y energía, los cuales cuentan con un alto valor agregado justificado por medio la tecnología de punta empleada, así como la metodología ajustada a los más altos estándares de seguridad, calidad, así como amigable con el medio ambiente.

Cuenta con presencia en países como: Argentina, Brasil, Colombia, Estados Unidos, Guatemala y México, en los cuales ya cuenta con 17 centros productivos, proveyendo a industrias y mercados afines a tal categoría. [\[1\]](#)

Entre su visión y misión podemos encontrar [\[1\]](#):

#### Visión

“Nuestra visión es ser la empresa siderúrgica líder de América, comprometida con el desarrollo de sus clientes, a la vanguardia en parámetros industriales y destacada por la excelencia de sus recursos humanos”

## **Misión**

Crear valor con nuestros clientes, mejorando la competitividad y productividad conjunta, a través de una base industrial y tecnológica de alta eficiencia y una red comercial global.

Y en cuestión de Estadísticas, Ternium en Latinoamérica tiene:

- Ventas netas de 10 millones de dólares anuales
- Capacidad instalada de 12.5 millones de toneladas
- 20,000 empleados (+20,000 externos y en México solamente +10,000)
- 10,000 clientes en 35 países
- 4,000 proveedores

## **Descripción del proceso asociado al reto**

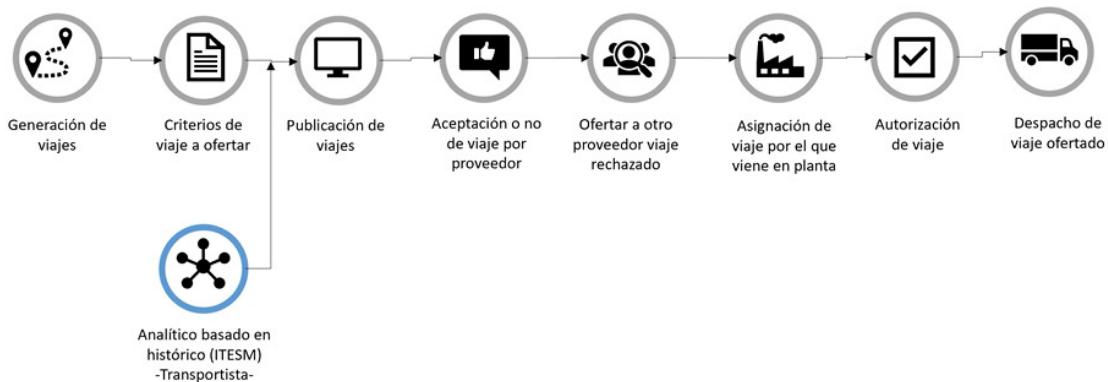
Uno de los puntos fundamentales de la empresa es el transporte de sus productos y elementos, para ello el proceso de asignación de viajes en ocasiones supone un retraso debido al factor de verificación manual que el transportista debe efectuar.

En cuestión de la definición de los pasos que tiene la empresa en cuestión de la asignación de un viaje son:

1. Llegada del transportista
2. Almacén Ternium
3. Asignación Automática/Manual
4. Carga/Peso Material
5. Y finalmente la entrega a cliente Local / Foráneo

El proceso asociado y su relación con el reto consiste en: analizar un conjunto de datos reales con el objetivo de construir un modelo predictivo para el área de Logística de la empresa, esto con el fin de la optimización del proceso de asignación de viajes de productos finales a diferentes transportistas. Nos apoyaremos de técnicas de aprendizaje automático para realizar este reto, en la que intervienen factores como: la gestión de inventarios, servicio al cliente, almacenamiento, despacho y transporte, la gestión de inventarios, servicio al cliente, almacenamiento, despacho y transporte.

En la siguiente imagen, se muestra de manera sencilla el proceso actual de asignación de viajes así como el proceso que se quiere lograr al cumplir el objetivo:



*Imagen sobre el proceso de asignación de viajes automático.*

Como se puede observar en la imagen, primero se genera el viaje de acuerdo al pedido o solicitud, después, de acuerdo a ciertos criterios o características del viaje, se publican los viajes con una propuesta de los transportistas asignados a cada viaje. El transportista puede aceptar o rechazar dicha asignación por lo que en caso de rechazo, se asignará un suplente. Una vez autorizado el viaje, se carga el producto, se pesa el camión junto con los productos y se despacha el camión de la planta con el viaje asignado.

Lo que se busca en este proyecto, es un algoritmo que realice una asignación automática para generar las propuestas de cada viaje con cada transportista asignado de manera eficiente teniendo en cuenta los criterios o características del viaje (Analítico basado en histórico).

## Objetivos

Mejorar el valor para los grupos de interés, afianzando la posición de la compañía como productor líder en Latinoamérica y sólido jugador en las Américas, reforzando a la vez su competitividad". [3]

En cuestión de los objetivos en específico del área de logística:

- Es la "optimización en cuestión de la metodología y proceso en la cual se realiza la asignación de viajes de los productos en su estado final a la variedad de transportistas y conductores que son proveedores de Ternium".
- Mayoría de la automatización de la asignación de un viaje en cuestión de las variables asociadas (carga, destino, transportista, etc.)
- Incremento de la productividad en cuestión de la eficiencia y eficacia en la asignación de viajes.
- Reducción de irregularidades minimizando el proceso manual de asignación de viaje.

## Criterios de Éxito

En cuestión de los criterios de éxito para el cumplimiento de objetivos tenemos los siguientes:

- Aumentar al menos un 40% en el total de automatización de viajes para alcanzar como mínimo un 60% del porcentaje total efectuado por herramientas computacionales de análisis de datos en lugar del proceso manual. (Esto dado que la empresa nos informa que cerca del 80% de los viajes realizados son de forma manual)
- Reducir al menos un 80% de las irregularidades causadas en los procesos de asignación de viajes a través del aumento de la automatización mediante el resultado de análisis.
- Aumentar de al menos un 30% de la productividad final en cuestión de la asignación de viajes.
- Reducir al menos un 40% del tiempo en logística destinado a la asignación de viajes.
- Aumentar al menos de un 5% de viajes totales realizados debido a la disponibilidad de fechas y ahorro de tiempo por medio de la automatización.

## 1.2 PROBLEMA ACTUAL

La problemática principal de nuestro proyecto consiste en la ineficiente asignación de viajes de manera automática a transportistas en la entrega de productos. Esto puede resultar en pérdida de tiempo, posibles errores en la carga de productos, etc..

Cuando los camiones de reparto no se asignan de manera óptima de acuerdo a la carga, productos y destinos, pueden terminar en un retraso considerable de tiempo así como realizar una mayor cantidad de viajes innecesarios, por ejemplo, si hablamos de un transportista de reparto el cual tiene 10, 20 o 40 entregas que tiene que hacer durante la jornada, es muy probable que la productividad sea muy baja si no se cuenta con las herramientas adecuadas a la hora de la asignación, además, si tenemos en cuenta que no solamente es un transportista, sino varios transportistas al mismo día, la problemática se vuelve aún mayor [4].

Los recursos disponibles con los que contamos o se pueden desarrollar son:

- Bases de datos: Durante la realización del reto, se contará con dos bases de datos las cuales corresponden a la asignación de viajes de dos plantas, Guerrero y Pesquería.
- Jupyter Notebook: Es una aplicación web de código abierto que nos permite crear y compartir código y documentos. Es un entorno informático interactivo, que permite a los usuarios experimentar con el código y compartirlo.
- Librerías de Python: Algunas librerías que se usarán para el análisis de datos son Pandas, numpy, matplotlib, seaborn, sklearn entre otros.

Sobre las bases de datos, son bases de datos en Excel en donde los datos pertenecen a dos plantas, una llamada “Guerrero” y otra llamada “Pesquería”. La base de datos “Guerrero” contiene 9,154 datos o

registros mientras que la base de datos de “Pesquería”, cuenta con 5,286 datos. Ambas bases de datos contienen distintas columnas como fechas de los transportistas, como la fecha y hora de llegada a la planta, la hora en que pesan el camión, la llegada a la nave donde cargan el material, la salida de despacho de la planta, entre otras.

También se encuentran descritas los ID de los viajes, clientes, estados de destino y del conductor. Por otra parte, también se describe el transportista, desde la empresa, hasta el tipo de camión que opera, así como el material y el peso subido a cada uno de los camiones.

En cuanto a las restricciones que se presentan durante la asignación de viajes, se encuentran varias reglas del negocio. La primera es que hay transportistas que solamente circulan de forma local, es decir, solamente en el área metropolitana,. La segunda son las restricciones que se pueden encontrar en la relación entre el tipo de transportista con el tipo de producto, de acuerdo a una serie de reglas, no todos los transportistas pueden llevar cualquier tipo de producto. También un mismo viaje puede llevar varios productos así como un transportista puede tener varios tipos de plataformas o camiones. Por último, es importante mencionar que hay valores nulos, así como valores que no concuerdan con las mismas fechas, por ejemplo.

### **1.3 MARCO TEÓRICO - MODELOS DE CLASIFICACIÓN**

El modelo de clasificación que usamos, de Random Forest, porque el reto pedía predecir una variable categórica (empresa transportista) es una técnica de aprendizaje automático que utiliza múltiples árboles de decisión para realizar una predicción. En este contexto de logística y transporte, este modelo lo utilizamos para predecir la empresa transportista que se encargará de llevar un paquete desde una planta de origen (pesquería o guerrero) a una zona de destino, en función de ciertas variables. Estos modelos son especialmente útiles para empresas de logística y transporte que manejan grandes volúmenes de envíos y necesitan tomar decisiones rápidas y precisas sobre cómo manejar cada paquete. Al utilizar modelos de Random Forest, estas empresas pueden tomar decisiones basadas en datos y reducir la probabilidad de errores humanos, ya que Ternium hace esta asignación de viaje de forma manual en la mayoría de los casos. Este modelo se trata de una técnica bien establecida y ampliamente utilizada en la industria de la logística y el transporte.

El primer artículo que investigamos, A REVIEW OF TRAVEL-TIME PREDICTION IN TRANSPORT AND LOGISTICS [14], nos dice la importancia de la información de tiempo de viaje en el transporte y la logística, y cómo puede ser utilizada para ahorrar tiempo y mejorar la fiabilidad en la selección de rutas de viaje. También destaca cómo la información de tiempo de viaje puede reducir costos y mejorar la calidad del servicio en la logística, y presenta diferentes técnicas de predicción de tiempo de viaje. El segundo artículo “Optimización de rutas y logística: del reto a la

oportunidad" [4] se enfoca en la optimización de rutas en la logística, y cómo esta práctica puede generar importantes beneficios para las empresas. Se presenta a Ternium como ejemplo de empresa dedicada a la producción de acero en América Latina, y se describe su estructura organizacional, actividades, estados financieros e informes anuales. El tercer artículo de Optimización con algoritmo genético [6] nos explica la teoría de los algoritmos genéticos y cómo se implementan en Python para resolver problemas de optimización. Se presenta un ejemplo práctico de optimización mediante algoritmos genéticos para un problema de optimización de funciones matemáticas. Y el último artículo que investigamos: "Inventory management and logistics optimization" [5] propone una mejora en la clasificación de inventario en una industria de alimentos utilizando técnicas de minería de datos y el algoritmo "Partitioning Around Medoids". Los resultados mejoraron las rutas de viaje, la eficiencia en tiempo y la optimización de entrega de productos, mejorando la toma de decisiones a través de la automatización por análisis de datos.

## 1.4 PROYECTO DE ANÁLISIS PREDICTIVO

### Determinar las metas del Proyecto de Análisis Predictivo, sus criterios de éxito (Kevin Antonio González)

#### 1.4.1 Objetivo

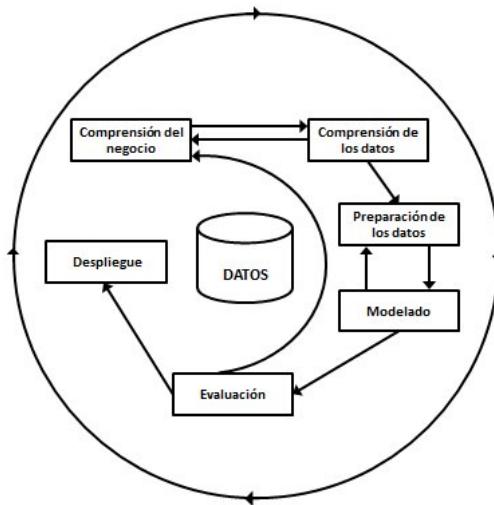
Optimizar, a través del análisis de datos, el proceso de asignación de viajes a transportistas de Ternium.

Actualmente, la asignación de viajes se hace de forma manual la mayoría de las veces, y esto provoca múltiples errores, como asignación errónea del destino adecuado o el producto correcto, dependiendo del tipo de transporte y su capacidad. La empresa tiene información histórica de los tipos de camiones, productos y clientes, que puede utilizar para que el proceso se realice de forma eficiente y poder asignar eficientemente un producto y su destino al transportista adecuado.

#### 1.4.2 Hipótesis

La automatización del proceso de asignación de viajes a transportistas de Ternium en comparación con el actual proceso manual, conlleva una disminución en el tiempo en la logística del transporte y una disminución en los errores de asignación, lo que provoca una mejora en el servicio al cliente, un aumento en los ingresos de la empresa y una reducción en el impacto ambiental.

#### 1.4.3 Metodología CRISP-DM



*Imagen de la metodología CRISP-DM*

La metodología CRISP-DM es un proceso estándar utilizado para guiar a los profesionales de la minería de datos en la realización de proyectos. CRISP-DM consta de seis fases principales:

Comprendión del problema: En esta fase se definen los objetivos del proyecto y se establece un plan de trabajo. Se identifican las fuentes de datos, se recopila información sobre el negocio y se establecen los criterios para medir el éxito del proyecto.

Comprendión de los datos: En esta fase se lleva a cabo la exploración de datos para comprender las características de los datos disponibles. Se realizan tareas de limpieza, preprocesamiento y transformación de los datos. Se verifica la calidad de los datos y se identifican posibles problemas.

Preparación de los datos: En esta fase se seleccionan los datos que se utilizarán en el análisis. Se realizan tareas de transformación y limpieza de datos, y se generan nuevas variables si es necesario. Los datos se formatean para el análisis y se dividen en conjuntos de entrenamiento y prueba.

Modelado: En esta fase se seleccionan las técnicas de modelado y se construyen modelos utilizando los datos preparados. Los modelos se validan y se ajustan para garantizar su precisión y generalización.

Evaluación: En esta fase se evalúan los modelos construidos. Se comparan los resultados de los modelos y se selecciona el mejor modelo para el despliegue.

Despliegue: En esta fase se implementa el modelo seleccionado en el entorno operacional. Se desarrollan planes para el monitoreo y mantenimiento continuo del modelo.

#### 1.4.4 Resultados y beneficios esperados

Con la automatización del proceso se busca minimizar los errores en la asignación, minimizar el tiempo de espera de los transportistas y la optimización en la logística permitiría una mayor cantidad de viajes en un menor tiempo, aumentando la productividad de la empresa en comparación con el

proceso manual. Al agilizar el proceso de asignación de viajes y la cantidad de estos tendrá un impacto en el servicio al cliente, en donde estos recibirán en menor tiempo los productos. [4]

Además al minimizar los errores, serán menos los clientes que se encuentran insatisfechos por problemas de logística de la empresa. Se espera que con la automatización del proceso, la reputación de la empresa sea más sólida debido a la calidad del servicio que proporciona, lo que atraerá más clientes y más inversionistas, esto provocaría un aumento en los ingresos económicos de la empresa.

El optimizar este proceso y minimizar los errores en la asignación de productos también tiene un beneficio en el medio ambiente, debido a que serán menores los viajes erróneos, en donde hay un gasto en gasolina y mayores emisiones de dióxido de carbono.

Una logística de transporte eficiente también puede hacer que la empresa sea más flexible, lo que significa que la empresa puede responder más rápidamente a las demandas del mercado, ajustar sus operaciones según sea necesario y adaptarse a los cambios en las condiciones del mercado. Esto también puede ayudar a la empresa a gestionar su inventario de manera más eficiente, lo que puede reducir los costos asociados con el almacenamiento y la gestión del inventario.

Así como se expone en el artículo “**Inventory management and logistics optimization**”, por medio de la utilización de herramientas y métodos basados en la minería de datos o algoritmos de búsqueda como: “**Partitioning Around Medoids**” (PAM), fundamentados en métodos de clustering, la reducción de costos en áreas de logística en términos de inventario (al cual se le atribuye un cercano de 15% de los mismos) es factible y de gran beneficio en optimización para empresas de amplia índole. [5]

Otro método de búsqueda de optimización heurística es por medio de los algoritmos genéticos que se basan en una técnica de optimización inspirada en la evolución biológica. Su nombre proviene de la analogía que se establece con la genética y la evolución de las especies.

La teoría de los algoritmos genéticos se basa en tres principios fundamentales:

- **Selección:** Se seleccionan las soluciones más aptas para sobrevivir y reproducirse.
- **Crossover (Cruce):** Se combinan las soluciones seleccionadas para crear nuevas soluciones.
- **Mutación:** Se introduce una pequeña variación aleatoria en algunas soluciones para evitar la convergencia prematura hacia una solución subóptima. [6]

Estos principios se aplican en cada iteración del algoritmo, que comienza con una población inicial de soluciones aleatorias. En cada iteración, se evalúa el desempeño de cada solución en el problema de

optimización, se seleccionan las soluciones más aptas, se realizan operaciones de cruzamiento y mutación para crear nuevas soluciones, y se repite el proceso hasta alcanzar un criterio de parada.

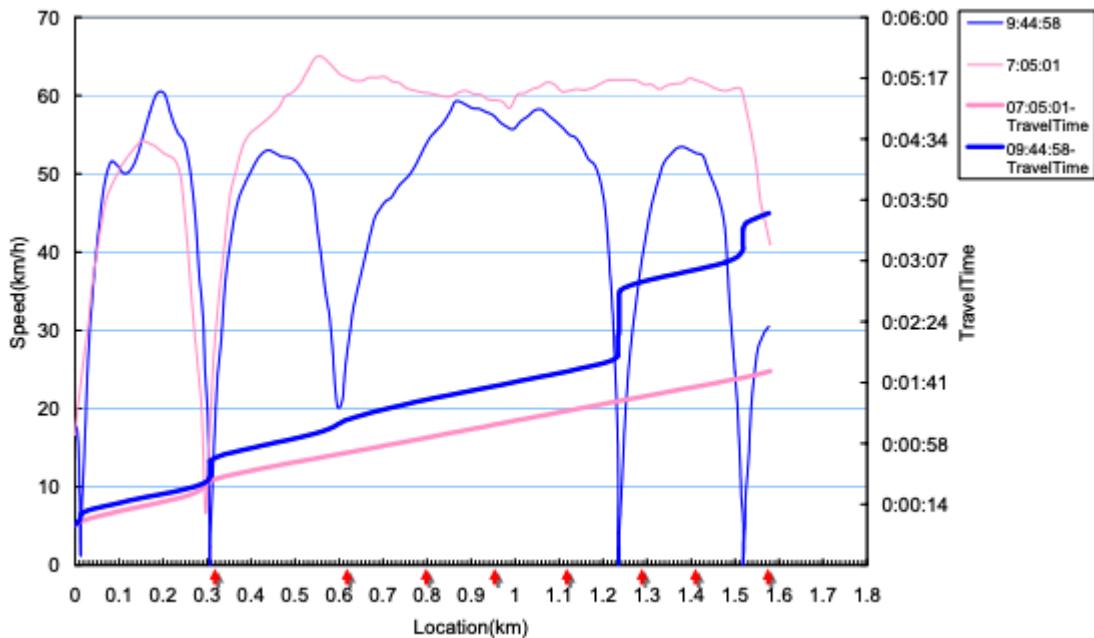
En la Tabla 1 podemos ver los beneficios de automatizar los procesos que permite a una organización desarrollar un sistema de gestión de calidad, minimizando errores, facilidad de seguimiento, integrando todos los elementos de forma dinámica, todo esto a través de la identificación, análisis, diseño, ejecución, monitoreo y control de los procesos de negocio de una empresa para asegurar que estos estén alineados con los objetivos estratégicos de la organización y que operen de manera eficiente y efectiva.

*Tabla 1. Documentación tradicional versus documentación enfocada a BPM [2]*

Documentación actual	Documentación BPM - Automatización
Información dispersa.	Información organizada.
Procesos documentados sin estándares internacionales.	Documentación bajo estándares internacionales.
Los cambios en los documentos y su implementación son procesos lentos y generan mayor resistencia.	Se dan cambios cuya implementación es ágil. Se reacciona de manera más rápida ante los imprevistos.
La integración y comunicación entre procesos no es clara ni evidente.	Integra todos los procesos de forma dinámica.
Los procesos se mantienen en secreto y no se divultan en todos los niveles de la organización.	Existe transparencia ante las partes interesadas. Se conocen y se pueden gestionar las mejoras. Existe una verdadera gestión del conocimiento en la organización.
Exceso de papel.	Eliminación del uso del papel o ahorro del mismo.
Exceso de vistos buenos en los procesos.	Disminución de vistos buenos. Análisis de generación de valor por facilidad en el seguimiento.
No se obtiene información en tiempo real.	Facilidad en el seguimiento en tiempo real.
Se confunden estrategias y procesos.	Se evidencian y se ejecutan los procesos.
No se identifican los errores y reprocesos.	Se identifican los errores y los reprocesos.
Dificultad para estandarizar tiempos y generar alertas o alarmas por retrasos.	Se generan alertas y alarmas por retrasos.
No se diferencia el diseñador del consumidor del proceso.	El diseñador y el ejecutor son claramente definidos e identificables.

En el artículo investigado [7] en la Gráfica 2 se busca analizar la relación entre las señales de tráfico y el tiempo de viaje en las carreteras arteriales, en donde la investigación consideró dos trayectos con salidas a las 7:05:01 y 9:44:58 respectivamente, representados por las líneas roja y azul. El resultado muestra claramente que el tiempo de viaje se ve afectado significativamente por los sistemas de señales. Para proporcionar una predicción precisa del tiempo de viaje en carreteras arteriales, los efectos de las señales son el primer desafío que se debe superar. Además, las pequeñas flechas debajo del eje x representan las ubicaciones de las intersecciones en el segmento de la ruta.

*Gráfica 2. Señales y tiempo de viaje [7]*



#### 1.4.6 Impacto social

Uno de los impactos sociales es las disminuciones de dióxido de carbono que generan los camiones debido a que al planear las rutas de recolección y entrega de productos, asignando rutas precisas para la entrega de productos de acuerdo a la localización y tamaños de estos, el número de camiones necesarios para transportar la misma cantidad de productos sería menor, y no solo la cantidad de camiones sino también el tiempo que ocupan estos camiones para la entrega.

Por lo que, principalmente, habría una disminución en la contaminación ambiental. Esto puede mejorar la calidad del aire en las áreas donde opera la empresa, lo que puede beneficiar a la salud de la población cercana.

Una mejor logística de transporte puede aumentar la disponibilidad y accesibilidad de los productos en la comunidad. Esto puede mejorar la calidad de vida de las personas que viven en áreas remotas o que tienen dificultades para acceder a los productos que necesitan, porque como menciona la empresa, empresa líder en el rubro del acero en Latinoamérica, tiene un gran impacto en la sociedad en la venta de estos materiales. También puede tener una reducción en el costo de estos productos y de los derivados, para los clientes, mejorando la economía de estos.

## 1.5 PLAN DE ACTIVIDADES

Etapa	Actividad	Tiempo / Fecha	Responsables
1: Comprensión del negocio	Determinar los objetivos del Negocio	Jueves 30 de Marzo 3:00-5:00 pm  y Viernes 31 de Marzo 5:00-7:00 pm	Adrián
	Evaluar la situación del Negocio		Luis
	Determinar las metas del Proyecto de Análisis Predictivo, sus criterios de éxito		Kevin
	Definir el plan del proyecto		Alexis
2: Exploración y Preprocesamiento de datos	Estudio de la correlación entre las variables.	Jueves 13 de Abril 3:00-5:00 pm  Viernes 14 de Abril 5:00-7:00 pm	Luis Kevin
	Construcción de gráficas de las variables fundamentales		Alexis Adrián
	Justificación de los resultados		Todos
	Limpieza de los datos y las variables redundantes		Alexis Kevin
	Normalización de los datos	Sábado 15 de abril	Adrián Luis
3: Modelación	Separación del set de variables de predicción	Jueves 20 de Abril	Luis Alexis
	Modelación mediante aprendizaje automático los escenarios más favorables	Viernes 20 de Abril	Kevin Adrián
4: Evaluación y creación de una aplicación web	Evaluar el modelo	Sábado 21 de Abril Y Martes 2 de Mayo	Todos
	Crear aplicación web operativa	Jueves 13, 20, 27 Viernes 14, 21, 28 Sábado 15, 22, 29 de Abril	Todos

Etapa	Actividad	Tiempo / Fecha	Responsables
1: Comprensión del negocio	Determinar los objetivos del Negocio	Jueves 30 de Marzo 3:00-5:00 pm  y Viernes 31 de Marzo 5:00-7:00 pm	Adrián
	Evaluar la situación del Negocio		Luis
	Determinar las metas del Proyecto de Análisis Predictivo, sus criterios de éxito		Kevin
	Definir el plan del proyecto		Alexis
5: Presentación del reto	Elaboración de la presentación	Jueves 4 de mayo	Alexis Adrián
	Elaboración del reporte general	Martes 2 , Jueves 4, y Viernes 5 de Mayo	Todos

## Etapa 2: Entendimiento de los datos

En la siguiente carpeta se encuentran los documentos individuales y los notebooks individuales y el grupal:

<https://drive.google.com/drive/folders/1UVN3hmJFX71XxfG3puljaatYjXXrDtYi?usp=sharing>

### 2.1 DESCRIPCIÓN DEL SET DE DATOS

El set de datos consistía en los datos de los meses de octubre, noviembre, enero, febrero y marzo de dos plantas de Ternium, la planta Guerrero y la planta Pesquería. En total fueron 7 archivos en formato Excel los que se nos proporcionaron y cada uno tenía entre 33 y 35 columnas así como de 4,695 a 9,154 datos o filas. En todos los archivos se encontraron valores nulos sobre todo en ciertas columnas por lo que decidimos quitar dichas columnas. El tipo de dato y variable de cada columna se describe en el siguiente enlace:

Link del Excel:

<https://docs.google.com/spreadsheets/d/1leSz2zUanKaMYlsnfPYIe0rr4iBLTVB/edit?usp=sharing&ouid=102937065857529183175&rtpof=true&sd=true>

### 2.2 INTEGRACIÓN DE DATOS

Lo primero que hicimos fue importar las siguientes librerías:

- *import pandas as pd*: la usamos para el análisis y manipulación de datos en formato de tabla. Pandas nos da una estructura de datos llamada DataFrame, lo que facilita el análisis y la manipulación de datos, es lo que usamos en todo el trabajo con los df.
- *import numpy as np*: la ocupamos para realizar cálculos numéricos. Proporciona una serie de funciones matemáticas y de álgebra lineal para manipular matrices y arreglos multidimensionales.
- *import matplotlib.pyplot as plt*: la usamos para la visualización de datos. Proporciona una serie de herramientas para crear gráficos de alta calidad, incluyendo gráficos de línea, de barras, de dispersión y de torta, entre otros.
- *import seaborn as sns*: nos dio una serie de herramientas para crear gráficos estadísticos avanzados, incluyendo gráficos de regresión, diagramas y mapas de calor, entre otros.
- *from scipy.stats import chi2\_contingency*: es utilizada para el análisis estadístico y la optimización numérica. Proporciona una serie de funciones estadísticas avanzadas, nosotros usamos el test chi-cuadrado.

Descargamos los siete archivos de excel los datos de la planta Guerrero y el de la planta Pesquería, y los convertimos a archivos csv, los guardamos en la misma carpeta en la que creamos el jupyter notebook. Leímos los siete archivos y como vimos que algunas columnas venían con distintos nombres, como Q\_CANTIDAD en lugar de CANT\_PROGRAMADA,

decidimos hacer el cambio de los nombres para que tuvieran el mismo nombre, además de ordenarlas de igual forma así como eliminar algunas columnas sobrantes.

Renombrando las columnas y eliminando las columnas sobrantes de algunos datasets:

```
df1 = df1.rename(columns={'C_CLIENTE.1': 'D_CLIENTE','Q_CANTIDAD':  
'CANT_PROGRAMADA','F_InicioCargaBulto': 'F_INICIOCARGABULTO'})  
df1 = df1.drop(['TIPO_ASIGNACION'], axis=1)
```

```
df2 = df2.drop(['TIPO_ASIGNACION'], axis=1)
```

```
df3 = df3.rename(columns={'FORMA': 'TIPO_FORMA','Q_CANTIDAD':  
'CANT_PROGRAMADA', 'TIPO_PRODUCTO': 'TIPO_PRODUCTO'})  
df3 = df3.drop(['F_FECHA_ALTA', 'TIPO_SERVICIO_1', 'TIPO_ASIGNACION'], axis=1)
```

```
df4 = df4.rename(columns={'FORMA ': 'TIPO_FORMA','Q_CANTIDAD':  
'CANT_PROGRAMADA', 'TIPO_PRODUCTO': 'TIPO_PRODUCTO'})  
df4 = df4.drop(['F_FECHA_ALTA', 'TIPO_SERVICIO_1', 'TIPO_ASIGNACION'], axis=1)
```

```
df5 = df5.rename(columns={'FORMA': 'TIPO_FORMA','Q_CANTIDAD':  
'CANT_PROGRAMADA', 'TIPO_PRODUCTO': 'TIPO_PRODUCTO'})  
df5 = df5.drop(['F_FECHA_ALTA', 'TIPO_SERVICIO_1', 'TIPO_ASIGNACION'], axis=1)
```

```
df6 = df6.rename(columns={'FORMA': 'TIPO_FORMA','Q_CANTIDAD':  
'CANT_PROGRAMADA', 'TIPO_PRODUCTO': 'TIPO_PRODUCTO'})  
df6 = df6.drop(['F_FECHA_ALTA', 'TIPO_SERVICIO_1', 'TIPO_ASIGNACION'], axis=1)
```

```
df7 = df7.rename(columns={'FORMA': 'TIPO_FORMA','Q_CANTIDAD':  
'CANT_PROGRAMADA', 'TIPO_PRODUCTO': 'TIPO_PRODUCTO'})  
df7 = df7.drop(['F_FECHA_ALTA', 'TIPO_SERVICIO_1', 'TIPO_ASIGNACION'], axis=1)
```

Ordenando las columnas:

```
# Reorganizar las etiquetas de columna en el orden deseado  
cols = ['PLANTAORIGEN', 'FECHADESPACHO', 'C_ID_VIAJE', 'FECHAVIAJE',  
'C_CLIENTE', 'D_CLIENTE', 'ID_ESTADO', 'ESTADO',  
'D_EMPRESA_TRANSPORTISTA', 'C_ID_CONDUCTOR', 'NOM_APE_COND',  
'TIPO_SERVICIO', 'TIPOCAMION', 'TIPOTRANSPORTE', 'TIPO_PRODUCTO',  
'TIPO_FORMA', 'CANT_PROGRAMADA', 'F_PRESENTACION',  
'F_LLEGADANAVE',  
'F_EGRESONAVE', 'F_INGRESOPLANTA', 'F_PESAJEENTRADA', 'F_ASIGVIAJE',  
'F_INICIOCARGABULTO', 'F_FINCARGA', 'F_PESAJESALIDA',  
'C_PLANTA', 'D_PLANTA', 'TIPO_PERMISO', 'PESO_NETO', 'CAP_MAXIMA',  
'ZONA_DESTINO', 'CUIDAD']
```

```
# Reordenar las columnas del DataFrame
df3 = df3.reindex(columns=cols)
df4 = df4.reindex(columns=cols)
df5 = df5.reindex(columns=cols)
df6 = df6.reindex(columns=cols)
df7 = df7.reindex(columns=cols)
```

Finalmente juntamos los 7 dataframes en uno solo usando df= pd.concat([df1,df2,df3,df4,df5,df6,df7])

## 2.3 MODELO CONCEPTUAL DE DATOS

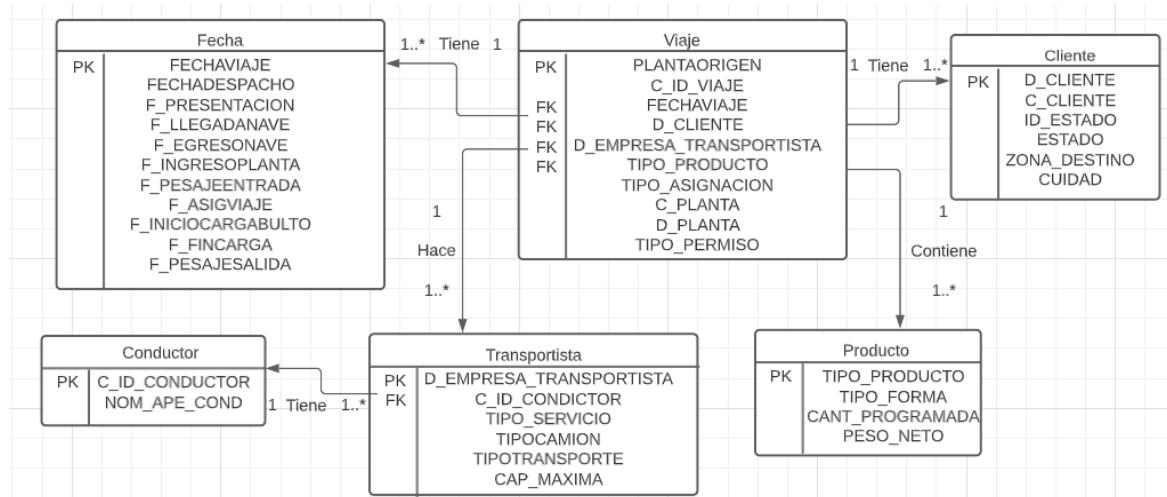


Imagen 15. Diagrama ER de ambas bases de datos

## 2.4 SELECCIÓN DE DATOS

Primero, a través de las gráficas de dependencia y las pruebas de correlación y chi.cuadrado explicadas en la entrega pasada pudimos seleccionar las variables de interés, decidimos quitar algunas porque estaban muy relacionadas con otras, por ejemplo el peso neto y la cantidad programada, daban casi la misma información o porque no eran relevantes para la asignación de viajes como las fechas o los IDs, por lo que nos quedamos sólamente con la siguientes variables:

Variable relevante	Datos no	Tipo	Descripción
--------------------	----------	------	-------------

	nulos		
PLANTAORIGEN	28689	entero categórica nominal predictora	Lugar de donde sale el material, las dos ubicaciones son en monterrey pero diferentes plantas GUERRERO y PESQUERIA es una variable que recibe texto y no tiene valores nulos
D_CLIENTE	28689	entero categórica nominal predictora	Es el cliente de ternium que compró el acero y al cual se le va a enviar el producto a sus instalaciones es un valor de texto, y cuenta con 18 datos nulos
ESTADO	28689	entero categórica nominal predictora	Nombre del estado donde se va a entregar el material y es texto con los estados de México, sin valores nulos
D_EMPRESA_TRANSPORTISTA	28689	entero categórica nominal predictora	ID de empresa transportista que se encarga del servicio a ternium de mover el material desde las instalaciones de la empresa a el cliente y es un valor de texto, sin valores nulos
TIPOTRANSPORTE	28689	entero categórica nominal predictora	Tipo de transporte que mueve el material, recibe texto y no tiene valores nulos
PESO_NETO	28689	entero categórica nominal predictora	Peso en toneladas de material que será enviado al cliente (puede variar segun la cantidad programada ), valor entero entre 0 y 41200, sin valores nulos

CUIDAD	28689	entero categórica nominal predictora	Ciudad en la cual se va entregar el material, recibe texto y no tiene valores nulos
--------	-------	---	---

En este caso, nuestra variable objetivo es la variable D\_EMPRESA\_TRANSPORTISTA, la cual es la que se desea predecir, mientras que las variables predictoras son las demás, las cuales son: PLANTAORIGEN, D\_CLIENTE, ESTADO, TIPOTRANSPORTE, PESO NETO y CIUDAD. La cantidad de valores no nulos es después de la limpieza, es por ello que disminuyó con respecto al dataframe del integrado.

Y usamos esa base de datos para hacer las pruebas de nuestros modelos de aprendizaje.

	PLANTAORIGEN_Pesqueria	D_CLIENTE	PESO_NETO	ESTADO	CIUDAD	TIPOTRANSPORTE	target
0		1	187	30670	1	39	9 20
1		1	107	36890	8	35	5 50
2		1	145	23300	14	61	6 39
3		1	147	20480	22	1	5 15
4		1	99	36980	1	39	5 40
...	...	...	...	...	...	...	...
14403		0	229	34580	14	23	6 50
14404		0	131	19090	5	20	5 35
14405		0	131	19090	5	20	5 35
14406		0	131	19090	5	20	5 35
14407		0	205	22880	2	9	5 48

*Gráfica 1. Dataframe de datos con variables relevantes*

## 2.5 LIMPIEZA DE DATOS

Antes que nada, iniciamos con un conocimiento previo del tipo de dato analizado en nuestra base de datos:

```
df_dirty.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 33453 entries, 0 to 6187
Data columns (total 33 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   PLANTAORIGEN    33453 non-null   object 
 1   FECHADESPACHO   33453 non-null   object 
 2   C_ID_VIAJE      33453 non-null   float64
 3   FECHAVIAJE     33453 non-null   object 
 4   C_CLIENTE       7954 non-null    object 
```

*Imagen 16. Resumen de nuestra base de datos*

A través de esto podemos conocer que en nuestra base de datos la mayoría de datos son categóricos (28), int(4) y float(2)

	Total	Nulos	Percent
PLANTAORIGEN	0	0.000000	
FECHADESPACHO	0	0.000000	
C_ID_VIAJE	0	0.000000	
FECHAVIAJE	0	0.000000	
C_CLIENTE	25499	76.223358	
D_CLIENTE	14	0.041850	

Imagen 17. Porcentaje de comparación de valores nulos en nuestra base de datos

Si nuestras columnas, teniendo como estándar que si una de ellas excede un porcentaje >40% de valores nulos, nuestra columna es candidata definitiva a ser descartada en la limpieza de nuestra base de datos.

```
# percent of data that is missing
percentage_missing_values = (total_missing/total_cells) * 100
print("Hay un porcentaje total de:",percentage_missing_values,"% de valores faltantes")
```

Hay un porcentaje total de: 3.160542610395959 % de valores faltantes

Imagen 18. Función de Porcentaje de comparación de valores nulos en nuestra base de datos con respecto del total

Se procede a realizar una simplificación de nuestro DataFrame, en torno a realizar una limpieza más eficiente y efectiva, tomando en cuenta las consideraciones anteriores, hemos decidido eliminar las columnas: 'C\_CLIENTE', 'FECHADESPACHO', 'C\_ID\_VIAJE', 'FECHAVIAJE', 'ID\_ESTADO', 'C\_ID\_CONDUCTOR', 'NOM\_APE\_COND', 'TIPOCAMION', 'CAP\_MAXIMA', 'CANT\_PROGRAMADA', 'F\_PRESENTACION', 'F\_LLEGADANAVE', 'F\_EGRESONAVE', 'F\_INGRESOPLANTA', 'F\_PESAJEENTRADA', 'F\_ASIGVIAJE', 'F\_INICIOCARGABULTO', 'F\_FINCARGA', 'F\_PESAJESALIDA', 'C\_PLANTA', 'D\_PLANTA' y 'ZONA\_DESTINO'

Esto debido a que excedían el 40% de valores nulos o eran variables numéricas o categóricas redundantes debido a que podemos encontrar otra columna que nos brinde esa información, o bien, consideramos que algunas variables categóricas como los ID y fechas, son inutilizables para nuestros diagramas de correlación, por lo que no tendría sentido indagar en ellos.

Esta simplificación es hecha mediante la función .drop aplicado de la siguiente manera:

```
df = df.drop(['C_CLIENTE', 'FECHADESPACHO', 'C_ID_VIAJE', 'FECHAVIAJE', 'ID_ESTADO', 'C_ID_CONDUCTOR', 'NOM_APE_COND', 'TIPOCAMION', 'F_PRESENTACION', 'F_LLEGADANAVE', 'F_EGRESONAVE', 'F_INGRESOPLANTA', 'F_PESAJEENTRADA', 'F_ASIGVIAJE', 'F_INICIOCARGABULTO', 'F_FINCARGA', 'F_PESAJESALIDA', 'C_PLANTA', 'D_PLANTA', 'ZONA_DESTINO'], axis=1)
```

Imagen 19. Eliminación columnas redundantes o sobre poblada de valores nulos

### a) Elimina duplicados

Para la visualización de valores duplicados es posible de observar a través de valores booleanos con la siguiente función:

```
df_dirty.duplicated()
```

```
0      False
1      False
2      False
3      False
4      False
...
6179   False
6180   False
6181   False
6186   False
6187   False
Length: 33453, dtype: bool
```

*Imagen 20. Visualización de valores duplicados*

A través de la eliminación de duplicados se decidió actuar mediante una eliminación total de los datos sobre una parcial (Dado que estaríamos cotejando una eliminación total de dos filas repetidas completamente)

```
# a) Elimina duplicados
#Eliminar duplicados completa
df_dirty = df_dirty.drop_duplicates()
df_dirty
```

*Imagen 21. Función de eliminación de eliminación de valores duplicados de forma completa*

### b) Corrige valores erróneos.

En el código no nos permitía leer el archivo con acentos, por lo que usamos Visual Studio Code, para reemplazar todos los acentos a caracteres sin acentos. Además también teníamos texto en mayúsculas, otros en minúsculas, como CIUDAD DE MÉXICO y Ciudad de México, y corregimos estos valores para que quedara de un solo tipo, algunos estaban como CD JUAREZ, Ciudad Juarez, JUAREZ, y todos los unimos en uno sólo. Guardamos el archivo .csv y ahora si lo importamos en nuestro código

Para corregir todo ello usamos la función str.upper() para hacer todos los datos de las columnas ESTADO y CIUDAD en mayúsculas. Después reemplazamos y juntamos en uno solo todos aquellos datos que significaban lo mismo pero debido a errores de ortografía o de términos, el código las marcaba como diferentes.

Aquí un ejemplo de ello con la columna de ESTADO:

```

reemplazos_Estados = {'SAN LUIS POTOSI': 'SAN LUIS POTOSI', 'ESTADO DE MIXICO': 'ESTADO DE MEXICO', 'ESTADO DE MOXICO': 'ESTADO DE MEXICO',
'CIUDAD DE MOXICO': 'CIUDAD DE MEXICO', 'CIUDAD DE MIXICO': 'CIUDAD DE MEXICO', 'QUERITARO': 'QUERETARO',
'QUEROTARO': 'QUERETARO', 'YUCATON': 'YUCATAN', 'MICHOACON': 'MICHOACAN',}

df = df.replace(reemplazos_Estados)

```

*Imagen 22. Función para corregir faltas de ortografía.*

## 2.6 EXPLORACIÓN DE DATOS

- 1) Calcula medidas estadísticas

### Variables cuantitativas

Medidas de tendencia central: promedio, media, mediana y moda de los datos.

Medidas de dispersión: rango: máximo - mínimo, varianza, desviación estandar.

Al aplicar las funciones de .mean(), .median(), .max(), .min(), etc. obtuvimos los siguientes resultados de nuestras variables cuantitativas:

#### CANT\_PROGRAMADA

Media: 28.70895768698061

Mediana: 29.7635

Máximo: 41.446

Mínimo: 0.818

Varianza: 60.49343800284963

Desviación Estándar: 7.777752760460417

Moda: 0 25.929

#### PESO\_NETO

Media: 24506.418258452155

Mediana: 25690.0

Máximo: 41360.0

Mínimo: 0.0

Varianza: 139450969.13895392

Desviación Estándar: 11808.935986741308

Moda: 0 0.0

#### CAP\_MAXIMA

Media: 53.52590201177772

Mediana: 54.0

Máximo: 66.0

Mínimo: 0.0

Varianza: 9.129194245677132

Desviación Estándar: 3.021455650125802

Moda: 0 54.0

## Variables cualitativas o categóricas

Tablas de distribución de frecuencia

A continuación se muestran algunas tablas de frecuencias obtenidas:

	Valor	Frecuencia_TIPO_FORMA
0	PLANOS	28354
1	TUBOS	304
2	SUBPRODUCTOS	73
3	LARGOS	30
4	POLIN	25
5	ACERIA	21
6	ROLLO	16
7	REDONDO	14
8	RECTANGULAR	7
9	PLANOS, TUBOS	6
10	PLANOS, POLIN	3
11	CINTA	2
12	HOJA	1,

Tabla 1. Frecuencias TIPO\_FORMA

	Valor	Frecuencia_ESTADO
0	NUEVO LEON	29018
1	COAHUILA	1763
2	TAMAULIPAS	939
3	SAN LUIS POTOSI	590
4	MEXICO	197
5	BAJA CALIFORNIA	168
6	CHIHUAHUA	126
7	GUANAJUATO	114
8	CHIAPAS	101
9	QUERETARO	76

Tabla 2. Frecuencias ESTADO

	Valor	Frecuencia_D_EMPRESA_TRANSPORTISTA
0	TRANSPORTE98	3902
1	TRANSPORTE115	2915
2	TRANSPORTE100	2495
3	TRANSPORTE103	1730
4	TRANSPORTE108	1334
.	...	...
163	TRANPORTE2	1
164	TRANPORTE5	1
165	TRANSPORTE172	1
166	TRANSPORTE169	1
167	TRANSPORTE168	1

Tabla 3. Frecuencias D\_EMPRESA\_TRANSPORTISTA

Estas tablas se obtuvieron generando un arreglo de las columnas con variables categóricas y mediante un for se fue produciendo cada una de las tablas de frecuencias con la función de value\_counts().

## Moda

Mediante diversos prints, se imprimieron las distintas modas seleccionado cada columna con variable categórica en el data frame y usando la función mode()

```
La moda de "TIPO_FORMA" es: PLANOS
La moda de "PLANTAORIGEN" es: Pesqueria
La moda de "FECHAVIAJE" es: 12/01/23
La moda de "C_CLIENTE" es: H000126500
La moda de "ESTADO" es: NUEVO LEON
La moda de "D_EMPRESA_TRANSPORTISTA" es: TRANSPORTE98
La moda de "C_ID_CONDUCTOR" es: 92866.0
La moda de "TIPO_SERVICIO" es: FO
La moda de "TIPOCAMION" es: Plataforma 3 ejes Neumatica
La moda de "TIPO_PRODUCTO" es: ROLLO
La moda de "C_PLANTA" es: PSQ
La moda de "TIPO_PERMISO" es: Traslado Externo
La moda de "ZONA_DESTINO" es: 0
La moda de "CUIDAD" es: SAN NICOLAS DE LOS G
```

*Imagen 2. Moda de las variables categóricas*

## 2) Explora los datos usando herramientas de visualización

Variables cuantitativas:

Medidas de posición no-central: cuartiles, outlier (valores atípicos), boxplots

Para hallar los cuartiles se seleccionó las columnas de interés ('PESO\_NETO', 'CAP\_MAXIMA') del data frame y se usó la función quantile() resultando en:

```
Cuartil 25:
CANT_PROGRAMADA      21.7395
PESO_NETO             21407.5000
CAP_MAXIMA            54.0000
Name: 0.25, dtype: float64

Cuartil 50 (Mediana):
CANT_PROGRAMADA      29.7635
PESO_NETO             28840.0000
CAP_MAXIMA            54.0000
Name: 0.5, dtype: float64

Cuartil 75:
CANT_PROGRAMADA      35.66075
PESO_NETO              35210.00000
CAP_MAXIMA             54.00000
Name: 0.75, dtype: float64
```

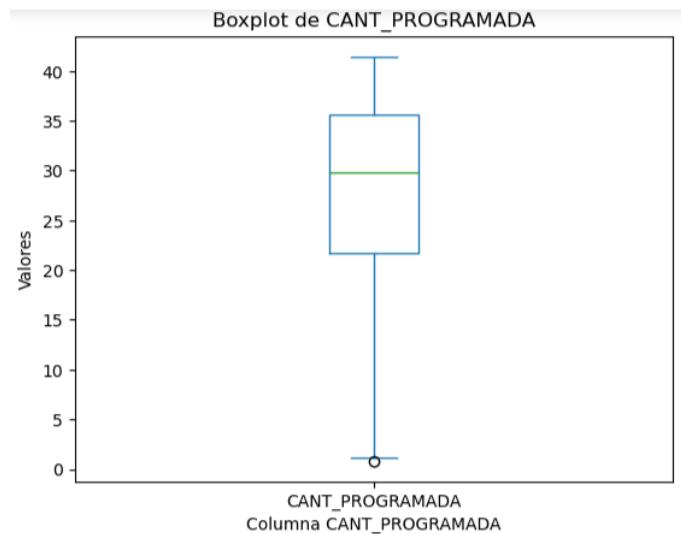
*Imagen 3. Cuartiles de las variables cuantitativas*

Para encontrar los valores atípicos (outliers), se utilizó el método de los valores Z, el cual se obtiene restando el valor de una columna con su media y dividirlo entre su desviación estándar. Si el valor Z absoluto es mayor al umbral (en este caso se utilizó un umbral de 3), se considerará un valor atípico. Como resultado se mostró un data frame vacío, es decir, no se hallaron valores atípicos.

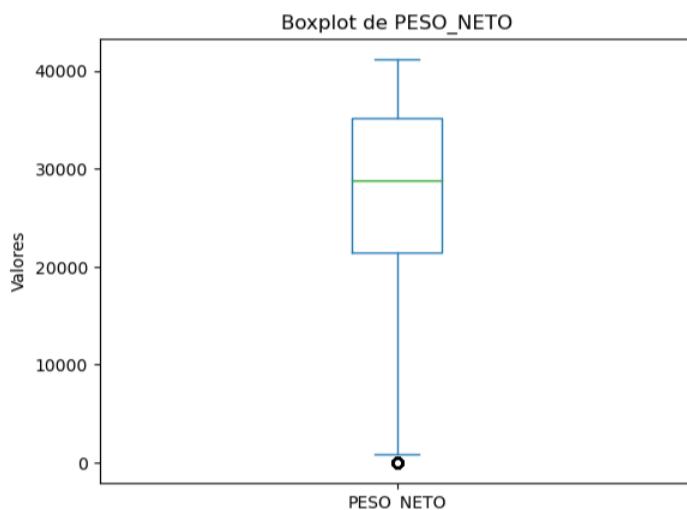
```
Empty DataFrame
Columns: [CANT_PROGRAMADA, PESO_NETO, CAP_MAXIMA]
Index: []
```

*Imagen 4. Dataframe de valores atípicos*

Por último, mediante la función `plot(kind='box')` se obtuvieron los siguientes boxplots los cuales indican cómo se distribuyen los datos de cada columna en cuartiles:



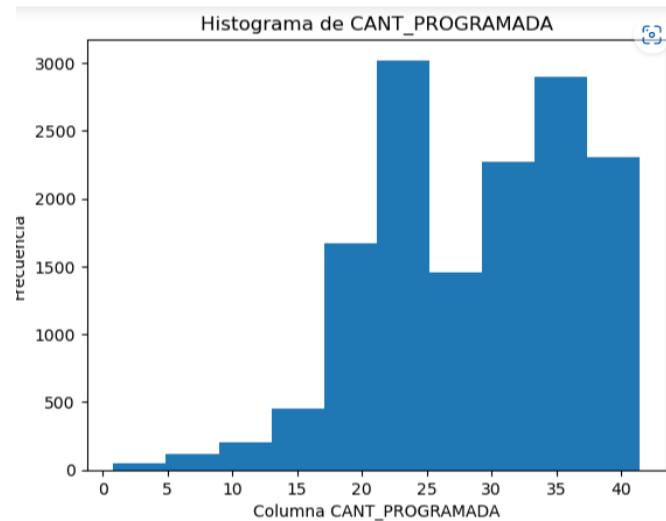
*Imagen 5. Boxplot de CANT\_PROGRAMADA*



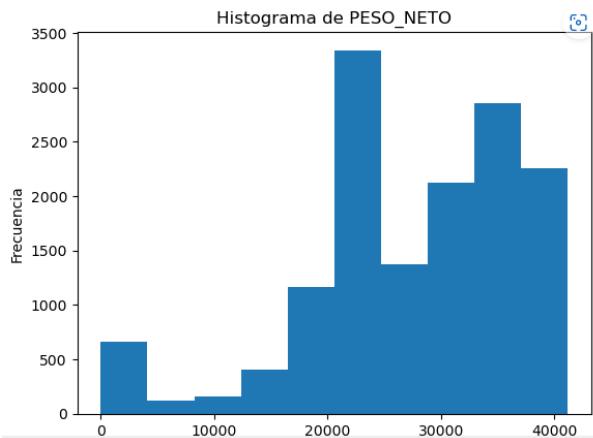
*Imagen 6. Boxplot de PESO\_NETO*

Análisis de distribución de los datos (Histogramas). Identificar si tiene forma simétrica o asimétrica

Mediante la función `plot(kind='hist')` se obtuvieron los siguientes histogramas que muestran las distribución de las variables cuantitativas y, como se podrá observar, tiene una distribución o una forma asimétrica a la derecha, se muestran solo un par de ellos.



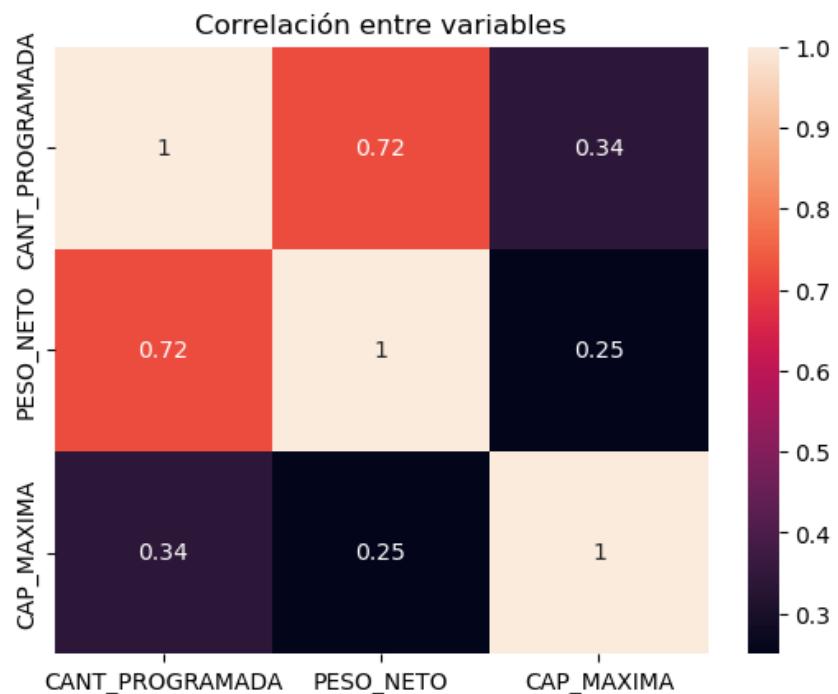
*Imagen 7. Histograma de CANT\_PROGRAMADA*



*Imagen 8. Histograma de PESO\_NETO*

Análisis de correlación de los datos, mapa de color

Para obtener un análisis de correlación, se realizó una matriz de correlación con la función `corr()` y `heatmap()`. Los valores más cercanos a 1 son aquellos que tienen una mayor correlación:

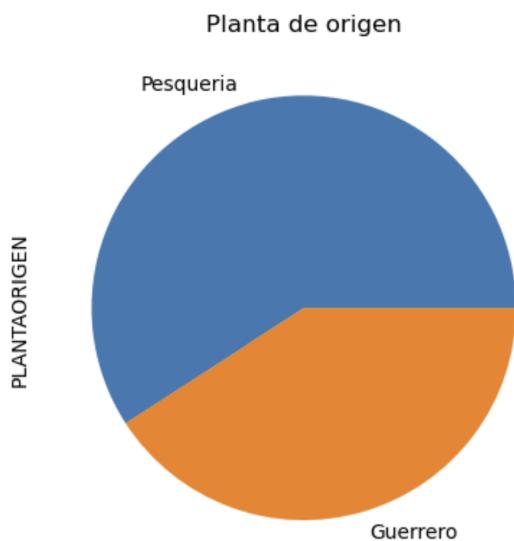


*Imagen 9. Matriz de correlación*

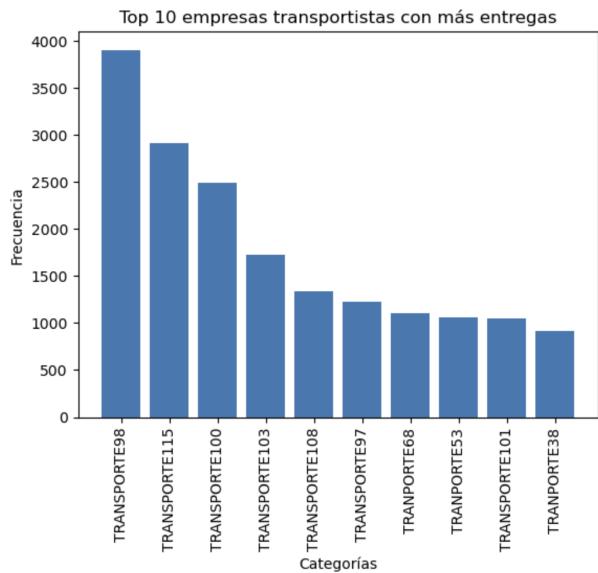
Variables cualitativas o categóricas

Distribución de los datos (diagramas de barras, diagramas de pastel)

A continuación se muestran algunas de las gráficas generales.

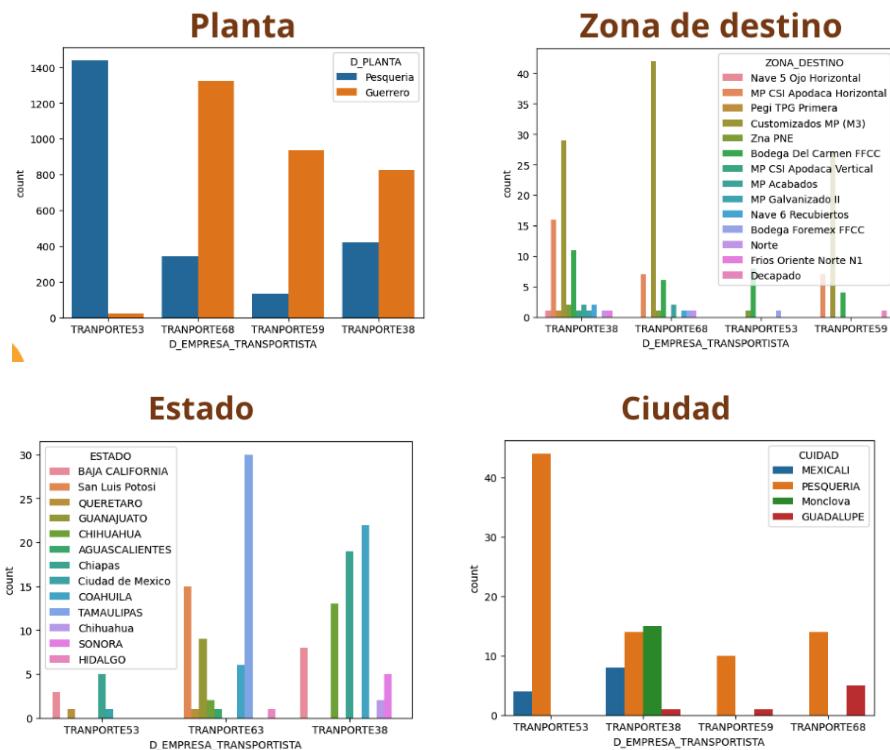


*Imagen 10. Diagrama de pastel de planta de origen*

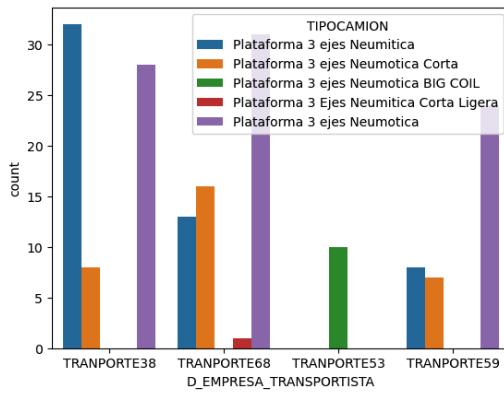


*Imagen 12. Histograma principales transportistas*

A continuación se muestran gráficas hacia lo particular:



*Gráficos de exploración de datos*



*Gráfica Tipo de camión y empresa transportista*

```
# Calcular La media y la desviación estándar
mean = np.mean(df_clean['PESO_NETO'])
std_dev = np.std(df_clean['PESO_NETO'])

# Definir un umbral para identificar outliers
threshold = 3

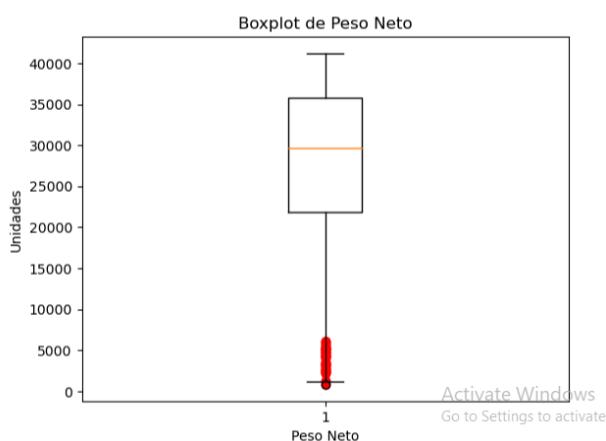
# Identificar Los valores outliers
outliers = []
for value in df_clean['PESO_NETO']:
    z_score = (value - mean) / (std_dev)
    if np.abs(z_score) > threshold:
        outliers.append(value)

# Imprimir Los valores outliers
print("Valores outliers encontrados:", outliers)
# Crear el gráfico de boxplot
plt.boxplot(df_clean['PESO_NETO'])
plt.title('Boxplot de Peso Neto')
plt.xlabel('Peso Neto')
plt.ylabel('Unidades')
plt.scatter(x=[1]*len(outliers), y=outliers, c='red', marker='o')
plt.show()
```

*Imagen 27. Código de obtención de Valores outliers y su gráfica Boxplot*

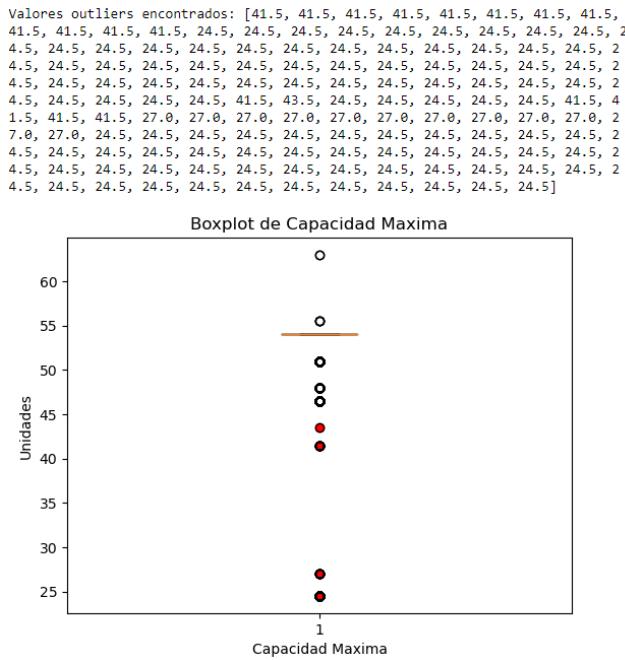
Para encontrar los valores outliers en cada conjunto de datos numérico primero calculamos la desviación estándar y la media, y mediante un boxplot ilustramos estos valores outliers de la siguiente manera:

Valores outliers encontrados: [5340, 5230, 2590, 4340, 1350, 4260, 4210, 4200, 2500, 4820, 5140, 3090, 3920, 3320, 3370, 5990, 2480, 6090, 5210, 3240, 4540, 2800, 2650, 3450, 3580, 5890, 5100, 4610, 3470, 4650, 5240, 2190, 5070, 4740, 6100, 5050, 5300, 3140, 5600, 1200, 830, 2700, 4140, 3810, 4300, 4130, 2260, 2440, 4730, 5110, 4440, 5150, 2380, 2300, 5690, 3330]



*Imagen 28. Boxplot de Peso Neto*

Repetimos de igual medida para los conjuntos de datos numéricos que sabemos lo podrían incluir, los cuales serían solamente CAP\_MAXIMA, dado que los otros son identificadores, por lo que nos quedaría así:



*Imagen 29. Boxplot de Capacidad Máxima*

## 2.7 TRANSFORMACIÓN DE DATOS

```

# d) Maneja datos categóricos: Transforma datos categóricos a datos numéricos si

# CATEGÓRICOS A FECHAS
df_clean['FECHADESPACHO'] = pd.to_datetime(df_clean['FECHADESPACHO'])
df_clean['FECHAVIAJE'] = pd.to_datetime(df_clean['FECHAVIAJE'])
df_clean['F_PRESENTACION'] = pd.to_datetime(df_clean['F_PRESENTACION'])
df_clean['F_LLEGADANAVE'] = pd.to_datetime(df_clean['F_LLEGADANAVE'])
df_clean['F_EGRESONAVE'] = pd.to_datetime(df_clean['F_EGRESONAVE'])
df_clean['F_INGRESOPLANTA'] = pd.to_datetime(df_clean['F_INGRESOPLANTA'])
df_clean['F_PESAJEENTRADA'] = pd.to_datetime(df_clean['F_PESAJEENTRADA'])
df_clean['F_ASIGVIAJE'] = pd.to_datetime(df_clean['F_ASIGVIAJE'])
df_clean['F_INICIOCARGABULTO'] = pd.to_datetime(df_clean['F_INICIOCARGABULTO'])
df_clean['F_FINCARGA'] = pd.to_datetime(df_clean['F_FINCARGA'])
df_clean['F_PESAJESENTIDA'] = pd.to_datetime(df_clean['F_PESAJESENTIDA'])

# CATEGÓRICOS A NUMÉRICOS
plantaOrg = {'Guerrero': 0, 'Pesqueria': 1} #creando diccionario
df_clean['PLANTAORIGEN'] = df_clean['PLANTAORIGEN'].map(plantaOrg)

```

*Imagen 235 Transformación de datos de variables categóricas a variables datetime y numéricas*

En cuestión de la conversión de datos categóricos se han elegido las fechas que están guardadas como datos categóricos, y convertirlas a datetime, de igual medida se escogió una variable categórica que será convertido a numérica (PLANTA ORIGEN), esto con el fin de identificar con mayor facilidad la salida de la planta, sirviendo como un ID de la planta, pero con oportunidad de correlación.

Quedando finalmente de la siguiente manera:

```
df_clean.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 12173 entries, 0 to 9150
Data columns (total 23 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   PLANTAORIGEN    12173 non-null   int64  
 1   FECHADESPACHO   12173 non-null   datetime64[ns]
 2   FECHAVIAJE      12173 non-null   datetime64[ns]
 3   ESTADO           12173 non-null   object  
 4   D_EMPRESA_TRANSPORTISTA 12173 non-null   object  
 5   C_ID_CONDUCTOR  12173 non-null   int64  
 6   NOM_APE_COND    12173 non-null   object  
 7   TIPOTRANSPORTE 12173 non-null   object  
 8   TIPO_PRODUCTO   12173 non-null   object  
 9   TIPO_FORMA      12173 non-null   object  
 10  F_PRESENTACION  12173 non-null   datetime64[ns]
 11  F_LLEGADANAVE   12173 non-null   datetime64[ns]
 12  F_EGRESONAVE    12173 non-null   datetime64[ns]
 13  F_INGRESOPLANTA 12173 non-null   datetime64[ns]
 14  F_PESAJEENTRADA 12173 non-null   datetime64[ns]
 15  F_ASIGVIAJE     12173 non-null   datetime64[ns]
 16  F_INICIOCARGABULTO 12173 non-null   datetime64[ns]
 17  F_FINCARGA      12173 non-null   datetime64[ns]
 18  F_PESAJESALIDA  12173 non-null   datetime64[ns]
 19  PESO_NETO       12173 non-null   int64  
 20  CAP_MAXIMA      12173 non-null   float64 
 21  ZONA_DESTINO    12173 non-null   object  
 22  CUIDAD          12173 non-null   object  
dtypes: datetime64[ns](11), float64(1), int64(3), object(8)
memory usage: 2.2+ MB
```

Imagen 26. Visualización transformación de datos

### a) Revisa si es necesario discretizar los datos (binning)

El binning es una técnica utilizada en estadística y análisis de datos para agrupar un conjunto de valores numéricos en un número finito de "contenedores" o "bins" (en inglés), dicha técnica puede ser útil para analizar datos de grandes conjuntos y sus principales aplicaciones serían:

- Hacer que los **datos sean más manejables y simplificados**, especialmente cuando los datos son muy dispersos o es un modelo complejo
- También puede ayudar a **resaltar patrones y tendencias** en los datos que de otra manera podrían pasar desapercibidos.
- **Cambio a variables categóricas en lugar de variables continuas.**
- Cuando se quiere simplificar un modelo complejo. Discretizar los datos puede ayudar a reducir la complejidad de un modelo, lo que puede mejorar su capacidad de generalización y evitar el sobreajuste.
- Cuando se quiere **reducir el ruido en los datos**.

En nuestro caso aunque los datos en los cuáles podríamos aplicar la técnica ya están discretos y son más que nada categóricos podríamos aplicar binning o agrupamiento de datos en: Fechas, tipo de producto, estados, pesos y destinos; pero al final terminamos ignorando las fechas ya que consideramos que no sería un dato tan relevante para nuestro análisis.

```
In [38]: #Cantidad de valores únicos
print(df_clean['TIPO_PRODUCTO'].nunique())
```

17

```
In [40]: print(df_clean['TIPO_PRODUCTO'].value_counts())
```

ROLLO

3675

Comenzando con aplicar binning para el Tipo de producto, contamos las diferentes categorías de la variable y posteriormente mostramos las cantidades de cada una para que de esta forma integrar cada bit y basandonos en la regla de pareto (20/80) podemos seleccionar solo los 4 tipos de material más representativos.

```
In [48]: #Calculamos la frecuencia de cada estado en la columna "ESTADOS
estado_counts = df_clean['ESTADO'].value_counts(normalize=True)

In [49]: #Creamos un diccionario que mapee cada estado a su categoría correspondiente
bins = {}
for estado, freq in estado_counts.items():
    if freq > 0.5:
        bins[estado] = "Más frecuentes"
    elif freq > 0.25:
        bins[estado] = "Frecuentes"
    elif freq > 0.1:
        bins[estado] = "Menos frecuentes"
    else:
        bins[estado] = "Raros"

In [50]: #Creamos una nueva columna con esta información
df_clean['ESTADO_BINNED'] = df_clean['ESTADO'].map(bins)

<ipython-input-50-a28b9032f5da>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
df_clean['ESTADO_BINNED'] = df_clean['ESTADO'].map(bins)
```

Para la variable de estado podemos aplicar diferentes percentiles y con base al porcentaje clasificarlo en una categoría de frecuencia y posteriormente crear una nueva columna que indique esa clasificación

```
#Semejante a lo que hicimos con los estados Lo hacemos con destinos
ciudad_counts = df_clean['CUIDAD'].value_counts(normalize=True)
#Creamos un diccionario
bins = {}
for ciudad, freq in ciudad_counts.items():
    if freq > 0.5:
        bins[ciudad] = "Más frecuentes"
    elif freq > 0.25:
        bins[ciudad] = "Frecuentes"
    elif freq > 0.1:
        bins[ciudad] = "Menos frecuentes"
    else:
        bins[ciudad] = "Raros"
#Creamos una nueva columna con esta información
df_clean['CIUDAD_BINNED'] = df_clean['CUIDAD'].map(bins)

<ipython-input-57-8dd9cb35bbfc>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
df_clean['CIUDAD_BINNED'] = df_clean['CUIDAD'].map(bins)
```

El mismo proceso se aplica para los destinos o ciudades

```
# Definir el número de bins automáticamente utilizando La regla de Sturges
n_bins = int(np.ceil(np.log2(len(df_clean))) + 1)

# Creamos los bins utilizando la función cut de pandas
bins = pd.cut(df_clean['PESO_NETO'], n_bins)

# Visualizamos los conteos en cada bin
print(bins.value_counts())

(19742.308, 22807.692]      809
(35069.231, 38134.615]      708
(32003.846, 35069.231]      551
(38134.615, 41200.0]        477
(22807.692, 25873.077]      444
(28938.462, 32003.846]      444
(25873.077, 28938.462]      309
(16676.923, 19742.308]      166
(13611.538, 16676.923]      48
(10546.154, 13611.538]      31
(7480.769, 10546.154]        30
(4415.385, 7480.769]         16
(1310.15, 4415.385]          13
Name: PESO_NETO, dtype: int64
```

Finalmente para aplicar binning en el Peso Neto asignamos el número de bins utilizando la regla estadística de Sturges y posteriormente creamos la función que separar en categorías a todos los pesos, de la manera óptima (En este caso nuestro algoritmo decidió que 14 intervalos era lo adecuado) y en cada intervalo (bin) hay diferentes frecuencias

**b) Si es necesario escala y normaliza los datos.**

La escalación y la normalización son técnicas comunes de preprocesamiento de datos que se utilizan para ajustar las características de los datos a un rango específico y ambas técnicas se utilizan para ayudar a mejorar el rendimiento y la precisión de los algoritmos de aprendizaje automático al reducir la influencia de las características con valores extremadamente grandes o pequeños.

Por un lado la escalación se refiere al proceso de ajustar las características de los datos para que tengan una escala uniforme sin cambiar la distribución original de los datos, es decir en casos donde existen 2 variables o más que hacen referencia a una misma propiedad pero en diferentes sistemas de medida como la longitud o el peso, para que ambos compartan la misma escala aplicamos la conversión de las medidas.

Por su parte la normalización se refiere al proceso de ajustar los valores de las características de los datos para que se encuentren dentro de un rango específico, generalmente entre 0 y 1 o -1 y 1 (representando el porcentaje). La normalización puede ser útil cuando la escala de las características de los datos varía ampliamente, lo que puede afectar negativamente la precisión de algunos algoritmos de aprendizaje automático. Por ejemplo, si tenemos un conjunto de datos que contiene una característica que varía de 0 a 100 (porcentaje/normal) y otra que varía de 0 a 1 (binaria), la normalización puede ser útil para ajustar ambas características dentro de un rango común.

En general estas técnicas son recomendables cuando:

- Los datos se van a utilizar en algoritmos de aprendizaje automático
- Los datos tienen diferentes unidades de medida
- Los datos tienen valores extremadamente grandes o pequeños

En nuestro caso por la etapa en la que estamos apenas llevamos una exploración e inmersión en los datos, pero es probable que en un futuro emplearemos una técnica de aprendizaje automático para analizar nuestros datos por lo cuál normalizamos y re-escalamos los datos numéricos como en el caso anterior: Peso neto utilizando la media aritmética y la desviación estándar, y para el caso de datos categóricos se aplicó la codificación One Hot (Estados y ciudades)

```

#calculamos la media y desviación estandar
media = df_clean['PESO_NETO'].mean()
desv_std = df_clean['PESO_NETO'].std()

#Normalizamos el peso neto usando la media obtenida
df_clean['PESO_NETO_norm'] = (df_clean['PESO_NETO'] - media) / desv_std

<ipython-input-60-82c1e4396417>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
df_clean['PESO_NETO_norm'] = (df_clean['PESO_NETO'] - media) / desv_std

#utilizamos la codificación one-hot para escalar y normalizar ciudades
datos_codificados = pd.get_dummies(df_clean, columns=['CUIDAD'])

#utilizamos la misma técnica para estados
datos_cciudad = pd.get_dummies(df_clean, columns=['ESTADO'])

```

- c) Si lo consideras conveniente, construye nuevos atributos que se puedan obtener a partir de los datos disponibles. (atributos derivados).

Creamos los siguientes atributos:

- **Duración del viaje:** Indica el tiempo desde el envío del producto de la planta hasta la recepción del cliente
- **Eficiencia de transporte:** Nos muestra el porcentaje de la capacidad de peso máximo utilizado por cada camión independientemente del número de pedidos
- **Tiempo de espera en la planta:** Es el indicador que muestra el tiempo que se tarda en procesar el pedido del cliente desde que se recibe la orden, se pesa, se sube al camión hasta que se comienza el envío, igual consideramos que analizar este dato sería una buena medida para indicar la eficiencia que se le está dando al proceso logístico y con base a ello proponer soluciones para optimizar tiempos.

```

#Para duración del viaje
df_clean['DURACION_VIAJE'] = df_clean['FECHAVIAJE'] - df_clean['FECHADESCRACHO']

<ipython-input-70-bd5dde0aa44>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
df_clean['DURACION_VIAJE'] = df_clean['FECHAVIAJE'] - df_clean['FECHADESCRACHO']

```

Para la duración del viaje esta variable es obtenida de la diferencia del tiempo entre la fecha de viaje y la fecha de despacho y posteriormente se anexa la nueva columna al documento

```

Para tiempo de espera
df_clean['F_INICIOCARGABULTO'] = pd.to_datetime(df_clean['F_INICIOCARGABULTO'])
df_clean['F_INGRESOPLANTA'] = pd.to_datetime(df_clean['F_INGRESOPLANTA'])
df_clean['TIEMPO_ESPERA'] = df_clean['F_INICIOCARGABULTO'] - df_clean['F_INGRESOPLANTA']

<ipython-input-73-a3c95e97e5ae>:2: SettingwithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
df_clean['F_INICIOCARGABULTO'] = pd.to_datetime(df_clean['F_INICIOCARGABULTO'])

<ipython-input-73-a3c95e97e5ae>:3: SettingwithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
df_clean['TIEMPO_ESPERA'] = df_clean['F_INICIOCARGABULTO'] - df_clean['F_INGRESOPLANTA']

```

Para el tiempo de entrega el procedimiento y la idea es similar

```

Para eficiencia de transporte
df_clean['EFICIENCIA_TRANSPORTE'] = df_clean['PESO_NETO'] / df_clean['CAP_MAXIMA']

<ipython-input-75-7dbabad1019f>:2: SettingwithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
df_clean['EFICIENCIA_TRANSPORTE'] = df_clean['PESO_NETO'] / df_clean['CAP_MAXIMA']

```

Finalmente, la eficiencia del transporte es obtenida por la proporción que representa el peso neto entre la capacidad máxima, que es diferente de acuerdo al tipo de transporte.

## 2.8 y 2.9 Correlación entre variables y Principales Hallazgos:

En este apartado fue un poco complicado decidir los datos que ocuparemos para realizar la automatización, pero gracias a las pruebas de independencia del test de chi cuadrada pudimos demostrar que había una relación significativa entre algunas variables, como la de la empresa transportista con la ciudad. Entonces consideramos que por ejemplo, si el paquete es muy pesado, es posible que se necesite un camión de mayor capacidad o incluso un transporte especializado para su entrega. La zona de destino también es un factor importante a considerar al elegir el tipo de transporte y el transportista adecuado. Dependiendo de la distancia y la ubicación geográfica, algunos transportistas pueden ser más adecuados que otros. Además, algunos transportistas pueden tener restricciones o limitaciones en ciertas áreas o regiones. La capacidad del camión es un factor clave a considerar al seleccionar el tipo de transporte adecuado para el paquete. Si el paquete es grande o pesado, es posible que se necesite un camión de mayor capacidad para su transporte. Por otro lado, si el paquete es pequeño, es posible que se pueda utilizar un camión de menor capacidad para su entrega. Cada tipo de transportista tiene sus propias limitaciones y beneficios, por lo que es fundamental evaluar cuál es el más adecuado para el paquete y su destino.

Primero pensamos que era necesario relacionar las características del viaje que se tenía que realizar, como la zona de destino, el peso, la capacidad del camión, y entre otras variables para determinar la empresa transportista que sería asignada a determinado pedido. Por lo que comenzamos a buscar relaciones entre estos datos y lo más relevante que encontramos es que depende de la empresa transportista para determinar el mayor número de envíos de cada zona donde sale, por ejemplo, en la gráfica planta de Gráficos de exploración, la empresa TRANSPORTE53 prácticamente todos sus envíos han sido desde pesquería, mientras que las otras tres empresas salen principalmente de Guerrero. Y así con las demás gráficas, se puede observar que dependiendo la empresa transportista va a ser donde se hagan los envíos a ciertas zonas, estados y ciudades. En la gráfica de Estado se observa que TRANSPORTE 63 hace la mayoría de sus envíos a Tamaulipas, mientras que las otras dos empresas no hacen ningún envío a ese estado. Así que dependiendo de la planta donde salga, la zona de destino, estado y ciudad se puede hacer una asignación a cierto transportista, esto a parte de las demás variables a considerar en el siguiente punto

Después para buscar una relación entre la empresa transportista las características del pedido a enviar, como su peso, o cantidad programada y algunas características del camión como su capacidad máxima o el tipo de camión, donde las pruebas vienen en el notebook individual y grupal. Podemos observar en la gráfica de Tipo de camión y empresa transportista, que sí hay una dependencia de las variables, dependiendo de la empresa transportista va a ser el tipo de camión que predomine. Además en la gráfica de correlación de variables se observa que el peso neto, la cantidad programada y capacidad máxima tienen una relación significativa con la empresa transportista gracias a la prueba de chi cuadrada que realizamos. Además la cantidad programada tiene una correlación de 0.72 con el peso neto, lo cual tiene mucho sentido, al igual de la correlación de capacidad máxima con capacidad programada con un valor de 0.34, lo que podíamos deducir debido a que un pedido que tenga una cantidad programada mayor, también necesitará mayor capacidad del camión para transportarse.

### Cantidad programada

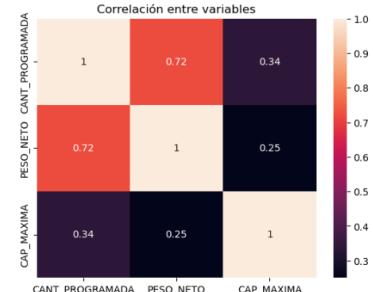
```
tabla_contingencia = pd.crosstab(df['D_EMPRESA_TRANSPORTISTA'], df['CANT_PROGRAMADA'])
resultado = chi2_contingency(tabla_contingencia)
print('Estadístico de chi-cuadrado:', resultado[0])
print('Valor p:', resultado[1])
print('Grados de libertad:', resultado[2])
Estadístico de chi-cuadrado: 866330.6276336021
Valor p: 0.0
Grados de libertad: 665010
```

### Peso neto

```
tabla_contingencia = pd.crosstab(df['D_EMPRESA_TRANSPORTISTA'], df['PESO_NETO'])
resultado = chi2_contingency(tabla_contingencia)
print('Estadístico de chi-cuadrado:', resultado[0])
print('Valor p:', resultado[1])
print('Grados de libertad:', resultado[2])
Estadístico de chi-cuadrado: 371496.1482955767
Valor p: 0.0
Grados de libertad: 236160
```

### Capacidad máxima

```
tabla_contingencia = pd.crosstab(df['D_EMPRESA_TRANSPORTISTA'], df['CAP_MAXIMA'])
resultado = chi2_contingency(tabla_contingencia)
print('Estadístico de chi-cuadrado:', resultado[0])
print('Valor p:', resultado[1])
print('Grados de libertad:', resultado[2])
Estadístico de chi-cuadrado: 33663.90954759114
Valor p: 0.0
Grados de libertad: 810
```



### Correlación de variables

En resumen podemos decir que sí hay una relación significativa entre el peso, la capacidad máxima, la cantidad programada, el tipo de camión, la zona, la ciudad y el estado de entrega, así como la planta de donde salga el pedido. Por lo que nosotros queremos encontrar la forma de que al tener todas estas características podemos proponer las empresas transportistas que mejor se adapten a esos requerimientos, y se pueda hacer una asignación automática.

### 3) Responsabilidades del equipo - Autoevaluación

**Incluyan en una tabla, por cada miembro del equipo, una autoevaluación.**

Integrante	Responsabilidades	Autoevaluación
Kevin González	<p>0) Integración de Datos</p> <p>1) Comprensión y exploración de los datos del negocio:</p> <ul style="list-style-type: none"> <li>a) Dimensión del dataset. Indica cantidad de registros y columnas</li> <li>b) Describe claramente en una hoja adicional del documento en Excel cada uno de los datos, incluyendo su nombre, descripción, tipo (categórico/Numérico), valores posibles que puede tomar, y total de valores nulos.</li> <li>c) Exploración de datos:</li> </ul> <p>1) Calcula medidas estadísticas</p> <p>Variables cuantitativas</p>	<p>Al principio fue un poco tedioso el proceso de eliminación de acentos, y el texto en diferentes formatos, pero pude lograr cambiarlo de forma fácil en VSC para poder leer los archivos. De ahí, lo más difícil fue encontrar cómo relacionar nuestras variables y encontrar las gráficas que mejor se adaptaran a lo que queríamos probar.</p> <p>Además la descripción de variables fue muy cansado porque eran muchas, y estar sacando sus máximos y mínimos y las definiciones y</p>

	<p>Medidas de tendencia central: promedio, media, mediana y moda de los datos.</p> <p>Medidas de dispersión: rango: máximo - mínimo, varianza, desviación estándar.</p> <p>Variables cualitativas o categóricas</p> <p>Tabla de distribución de frecuencia</p> <p>Moda</p> <p>2) Preparación de los datos:</p> <p>2.- Limpieza de datos</p> <p>b) Corrige valores erróneos.</p>	<p>todo eso me quitó algo de tiempo pero creo que era necesario para familiarizarme con la base de datos.</p>
Luis Maximiliano López	<p>Tabla de distribución de frecuencia</p> <p>2) Explora los datos usando herramientas de visualización</p> <p>Variables cuantitativas:</p> <p>Medidas de posición no-central: cuartiles, outlier (valores atípicos), boxplots</p> <p>Análisis de distribución de los datos (Histogramas). Identificar si tiene forma simétrica o asimétrica</p> <p>Análisis de correlación de los datos, mapa de color</p> <p>Variables cualitativas o categóricas</p> <p>Distribución de los datos (diagramas de barras, diagramas de pastel)</p> <p>d) Verifica la calidad de los datos: ¿existen valores faltantes, valores incorrectos en los datos, errores de ortografía?</p> <p>e) Genera Modelo de Datos en UML o ER a partir de la estructura del archivo csv (fuente de datos)</p>	<p>En mi opinión, la parte más difícil fueron las medidas de posición no-central y el diagrama ER de la base de datos. Ya que no recordaba cómo se hallaban los valores atípicos de un conjunto de datos así como los cuartiles. Por otra parte el diagrama ER resultó tedioso por la gran cantidad de columnas y el tratar de discernir entidad/clases y sus relaciones fue algo complicado. Lo demás estuvo bastante bien ya que eran cuestiones vistas en clase en su mayoría.</p>
Adrian Pineda Sánchez	<p>c) Exploración de datos:</p> <p>2) Explora los datos usando herramientas de visualización</p>	<p>En lo personal, fue algo extenuante el saber cómo manejar la información en torno a las medidas de</p>

	<p>VARIABLES CUANTITATIVAS:</p> <p>Medidas de posición no-central: cuartiles, outlier (valores atípicos), boxplots</p> <p>Análisis de distribución de los datos (Histogramas). Identificar si tiene forma simétrica o asimétrica</p> <p>Análisis de correlación de los datos, mapa de color</p> <p>VARIABLES CUALITATIVAS O CATEGÓRICAS</p> <p>Distribución de los datos (diagramas de barras, diagramas de pastel)</p> <p><b>2) Preparación de los datos:</b></p> <p>2)</p> <ul style="list-style-type: none"> <li>a) Elimina duplicados</li> <li>c) Maneja valores faltantes (imputa valores de acuerdo a lo que se considere apropiado).</li> <li>d) Maneja datos categóricos: Transforma datos categóricos a datos numéricos si es necesario.</li> <li>e) Maneja adecuadamente los valores atípicos (outliers) que encuentres en el dataset.</li> </ul>	<p>correlación, ya que el punto de coherencia, así como de utilidad en cuanto a un tipo de variable con el objetivo final del reto fue un desafío en general.</p> <p>En torno a la preparación de los datos, personalmente, aunque algo extenuante, fue de las partes más comprensibles y prácticas a realizar, aún y cuando para ello, ya hubiésemos definido las columnas objetivas, con las que ya deberíamos haber realizado ese análisis en torno a relaci</p>
Alexis Daniel Leyva	<p>(Notebook Individual)</p> <p><b>3.- Transformación de Datos:</b></p> <ul style="list-style-type: none"> <li>a) Revisa si es necesario discretizar los datos (binning)</li> <li>b) Si es necesario escala y normaliza los datos.</li> <li>c) Si lo consideras conveniente, construye nuevos atributos que se puedan obtener a partir de los datos disponibles. (atributos derivados).</li> </ul> <p>4.- Reformatea/reestructura los datos si es necesario</p>	<p>Desde el notebook individual pude notar las dificultades de la limpieza de la base de datos, que aunque aparentemente ya estaba estructurada, como era de esperar algunos valores no los leía python por los acentos y formatos como las fechas, o los tomaba como distintos dependiendo si estaban escritos con mayúsculas o minúsculas; y aunque quizás el formateo no fue lo más difícil si era algo frustrante porque cuando creímos que ya estaba bien y no funcionaba alguna parte, no era tan fácil saber si se debía al formato o a que estábamos equivocandonos en la redacción de la función, una vez</p>

solucionado esto el mayor reto fue saber y aprender que era binning y cuando era recomendable utilizar esta técnica, y aunque quizás pensaba que no era necesario del todo aplicarla, decidí hacerlo para después que como equipo tuviéramos un panorama más amplio para comparar las diferencias de los análisis con binning y sin binning y escoger el que consideramos más adecuado, sin duda es hasta que ya pones en práctica y experimentas con el código de python cuando te das cuenta si realmente estás aprendiendo o que necesitas reforzar, y como es la primera vez que me introduzco a la ciencia de datos como tal en lo que va de mi carrera se me hizo una experiencia innovadora, interesante, y algo retadora.

# **Etapa 3: Generación y Evaluación de Modelos de Aprendizaje Automático**

## a) Variables relevantes para el análisis

Mostrar en una tabla las variables relevantes para el proceso de modelación de algoritmos de aprendizaje. Incluir para cada variable su nombre, descripción, tipo de dato (entero/string) y tipo de variable (categórica nominal/ordinal, numérica). Indicar en la tabla cuáles son las variables predictoras y cuál es la variable objetivo o resultado.

## b) Datos disponibles

- 1) Mostrar una imagen de la tabla de datos, con al menos 10 registros, considerando sólo las variables/columnas relevantes para esta fase.
- 2) Mostrar el total de datos disponibles (filas y columnas) y la distribución de los mismos de acuerdo a la variable objetivo. ¿Se tienen datos balanceados?

Primero, a través de las gráficas de dependencia y las pruebas de correlación y chi.cuadrado explicadas en la entrega pasada pudimos seleccionar las variables de interés, decidimos quitar algunas porque estaban muy relacionadas con otras, por ejemplo el peso neto y la cantidad programada, daban casi la misma información, por lo que nos quedamos sólamente con la siguientes variables:

Variable relevante	Datos no nulos	Tipo	Descripción
PLANTAORIGEN	14408	entero categórica nominal predictora	Lugar de donde sale el material, las dos ubicaciones son en monterrey pero diferentes plantas GUERRERO y PESQUERIA es una variable que recibe texto y no tiene valores nulos
D_CLIENTE	14408	entero categórica nominal	Es el cliente de ternium que compró el acero y al cual se le va a enviar el producto a sus instalaciones es un

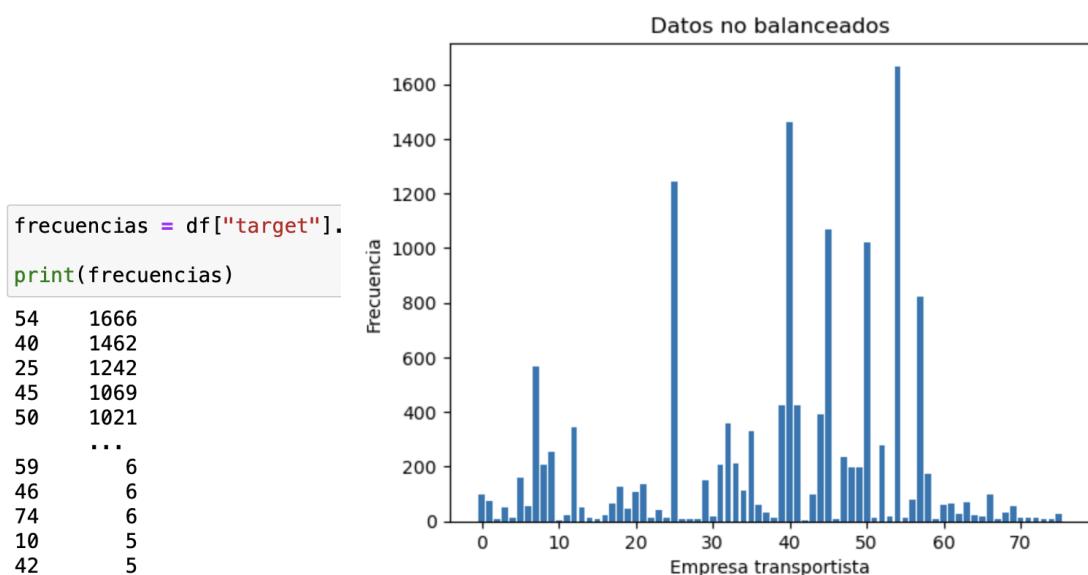
		predictora	valor de texto, y cuenta con 18 datos nulos
ESTADO	14408	entero categórica nominal predictora	Nombre del estado donde se va a entregar el material y es texto con los estados de México, sin valores nulos
D_EMPRESA_TRANSPORTISTA	14408	entero categórica nominal predictora	ID de empresa transportista que se encarga del servicio a ternium de mover el material desde las instalaciones de la empresa a el cliente y es un valor de texto, sin valores nulos
TIPOTRANSPORTE	14408	entero categórica nominal predictora	Tipo de transporte que mueve el material, recibe texto y no tiene valores nulos
PESO_NETO	14408	entero categórica nominal predictora	Peso en toneladas de material que será enviado al cliente (puede variar segun la cantidad programada ), valor entero entre 0 y 41200, sin valores nulos
CUIDAD	14408	entero categórica nominal predictora	Ciudad en la cual se va entregar el material, recibe texto y no tiene valores nulos

Y usamos esa base de datos para hacer las pruebas de nuestros modelos de aprendizaje.

PLANTAORIGEN_Pesqueria	D_CLIENTE	PESO_NETO	ESTADO	CIUDAD	TIPOTRANSPORTE	target
0	1	187	30670	1	39	9 20
1	1	107	36890	8	35	5 50
2	1	145	23300	14	61	6 39
3	1	147	20480	22	1	5 15
4	1	99	36980	1	39	5 40
...	...	...	...	...	...	...
14403	0	229	34580	14	23	6 50
14404	0	131	19090	5	20	5 35
14405	0	131	19090	5	20	5 35
14406	0	131	19090	5	20	5 35
14407	0	205	22880	2	9	5 48

*Gráfica 1. Dataframe de datos con variables relevantes*

Hicimos la gráfica de la frecuencia de las clases de nuestra variable objetivo empresa transportista y vimos que los datos no estaban balanceados como se muestra en la gráfica 2, en donde la empresa transportista 54 tiene 1666 viajes, mientras la empresa 42 tiene 5, por lo que las predicciones de nuestros modelos podrían no ser tan precisas. Decidí sacar los accuracy antes y después de balancear los datos para comparar los resultados.

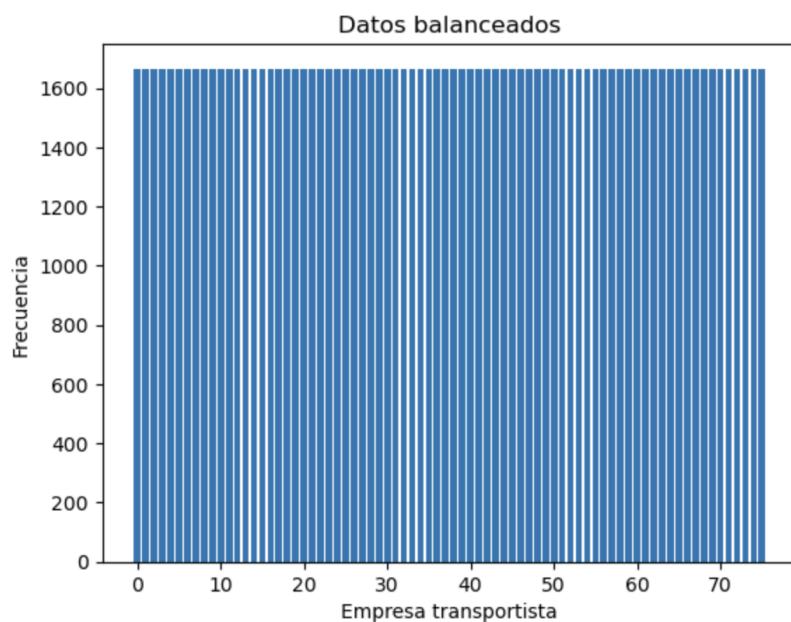


*Gráfica 2. Datos no balanceados*

Para balancear los datos utilizamos la función SMOTE(k\_neighbors=4) la cual es una técnica de sobremuestreo utilizada para resolver el problema que tenemos de desequilibrio de clases en el aprendizaje automático. Esta técnica genera nuevas muestras “sintéticas” de la clase con menos datos, usando puntos con valores conocidos o en una muestra de puntos para estimar valores en puntos desconocidos entre las muestras existentes de la clase. SMOTE se utiliza para crear una cantidad igual de muestras sintéticas para cada clase minoritaria, aumentando así la cantidad de datos de la clase minoritaria en el conjunto de datos. De esta manera, el modelo puede tener más datos para aprender sobre las clases con menos viajes, lo que ayuda a evitar el sesgo en la predicción. Como no tenemos tantos datos, y menos en el de las clases

con menos viajes, utilizamos la técnica oversampling para igualar la cantidad de muestras en las clases mayoritaria y minoritaria. El parámetro `k_neighbors=4` se refiere que se tomará el 4 como el número de vecinos que se utilizarán para seleccionar muestras para la interpolación. Este parámetro controla el nivel de interpolación y afecta la cantidad de muestras sintéticas que se generarán. Un valor mayor de `k_neighbors` generará más muestras sintéticas pero también puede llevar a una mayor sobreajuste. Un valor menor de `k_neighbors` generará menos muestras sintéticas pero puede ser más efectivo en la mejora del desequilibrio de clases.

Al final quedaron como se observa en la gráfica 3, todas las clases quedaron con una frecuencia de 1666 datos.



*Gráfica 3. Datos balanceados*

Después de esto creamos varios modelos de aprendizaje automático usando diferentes métodos de clasificación, como `KNeighborsClassifier`, `DecisionTreeClassifier`, `GaussianNB`, `RandomForestClassifier` y `MLPClassifier`. En `DecisionTreeClassifier` usé `criterion = 'entropy'` que significa que utiliza la entropía como medida de impureza para decidir cómo dividir el conjunto de datos en subconjuntos más homogéneos durante la construcción del árbol de decisión. Igual usamos la entropía en el random forest, y además le di el valor de 100 a `n_estimators` lo que significa que se crearán 100 árboles de decisión. En el de redes neuronales el parámetro "`hidden_layer_sizes`" indica la cantidad de neuronas que se deben

crear en cada una de las capas ocultas. Yo le puse (8,8,8), lo que significa que hay tres capas ocultas, cada una con 8 neuronas. El siguiente era "activation" que indica la función de activación que se utiliza en las neuronas de la red neuronal. En este caso, se establece en 'relu', que significa Unidad Lineal Rectificada, una función que se utiliza comúnmente en redes neuronales. Y "solver" indica el algoritmo utilizado para optimizar los pesos de las conexiones en la red neuronal y puse 'adam', que es un algoritmo de optimización basado en gradiente estocástico. El parámetro "max\_iter" indica el número máximo de iteraciones que se permiten en la optimización de los pesos de la red neuronal. En este caso, se establece en 500, lo que significa que el proceso de optimización se detendrá después de 500 iteraciones.

Luego los entrenamos con un conjunto de datos de entrenamiento llamado X\_train y y\_train. Después de eso, evaluamos el rendimiento de los modelos utilizando un conjunto de datos de prueba llamado X\_test y y\_test. Para hacer esto, calculamos la precisión de cada modelo usando el método score() y lo almacené en una variable diferente para cada modelo: acc\_knn, acc\_dt, acc\_nb, acc\_rf y acc\_mlp. Después creamos una gráfica que muestra los valores de precisión de cada modelo. Esto nos permitió comparar el rendimiento de los diferentes modelos y ver cuál funciona mejor.



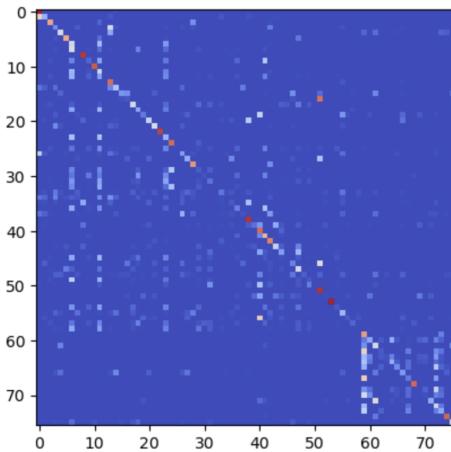
Gráfica 4. Modelos de aprendizaje

Después hicimos las gráficas de confusión para cada modelo, y en la gráfica 5 podemos ver la diferencia entre la de bosques aleatorios y la de NB, se supone que buscamos que la diagonal

principal de la imagen sea roja completamente, mientras que lo demás sea azul fuerte, ese sería el modelo más preciso y que no se equivocara, pero como el de bosques aleatorios tiene 0.766 de accuracy significa que hay veces que se equivoca, y los puntos blancos que se ven fuera de la diagonal son en los que falla el modelo, por eso se ven más puntos blancos en el de NB porque es menos preciso,

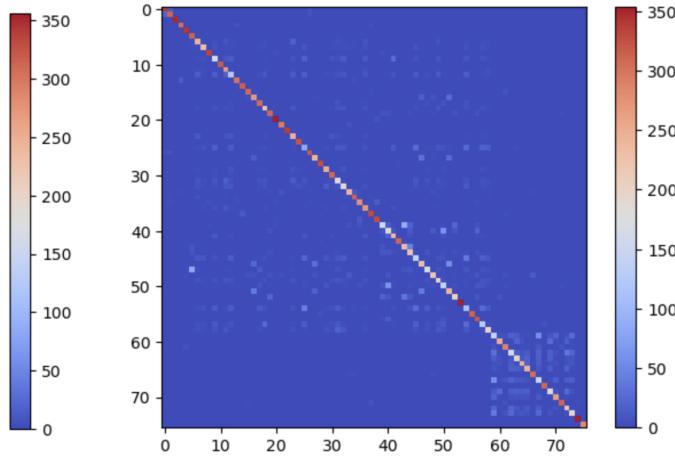
Exactitud del modelo Naive Bayes:  
0.35653925130311165

Matriz de confusión:  
 $\begin{bmatrix} 356 & 0 & 0 & \dots & 0 & 0 & 0 \\ 199 & 94 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 260 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & 83 & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 311 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 175 \end{bmatrix}$



Exactitud del modelo Bosques Aleatorios:  
0.7614910756594535

Matriz de confusión:  
 $\begin{bmatrix} 338 & 14 & 0 & \dots & 0 & 0 & 0 \\ 21 & 310 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 346 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & 195 & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 354 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 282 \end{bmatrix}$



*Gráfica 5. Matrices de confusión de Bosques Aleatorios y Naive Bayes*

Para mejorar la precisión del modelo, a Luis se le ocurrió mostrar las 3 clases con más probabilidades para que sea más probable que se coincida con la clase real. Para hacerlo, como se observa en la gráfica 6, primero, creamos el modelo de Bosques Aleatorios con 100 estimadores y el criterio de entropía. Luego utilizamos este modelo para hacer predicciones de probabilidades de clase en los datos de prueba, y se seleccionaron las tres clases con las mayores probabilidades utilizando la función argsort(). Después imprimimos la información de las tres mejores opciones de predicción y el resultado real para cada instancia de prueba. Después, se calcula el accuracy global del modelo para ver si alguna de las tres opciones de predicción coincide con la clase real, cada vez que coincidía la clase real con alguna de las tres predecidas se le iba sumando un 1 a nuestra variable aciertos, y al final dividimos el total de aciertos entre el total de predicciones (accuracy = aciertos / len(y\_pred\_classes)) y así sacamos el accuracy global.

```

: #Bosques Aleatorios
modelo = RandomForestClassifier(n_estimators = 100, criterion = 'entropy')
modelo.fit(X_train, y_train)

# Realizar predicciones para las instancias de prueba
y_pred_probs = modelo.predict_proba(X_test) # Probabilidades de predicción para cada instancia
y_pred_classes = modelo.classes_[y_pred_probs.argsort(axis=1)[:, -3:]] #Obtener las 3 mejores probabilidades

# Calcular el accuracy global para verificar si alguna de las 3 opciones coincide
aciertos = 0

# Imprimir las 3 mejores opciones de predicción para cada instancia de prueba
for i in range(len(y_pred_classes)):
    print(f'Instancia {i+1}: {y_pred_classes[i]} - Resultado Real:{y_test[i]}')

for i in range(len(y_pred_classes)):
    if y_test[i] in y_pred_classes[i]:
        aciertos += 1

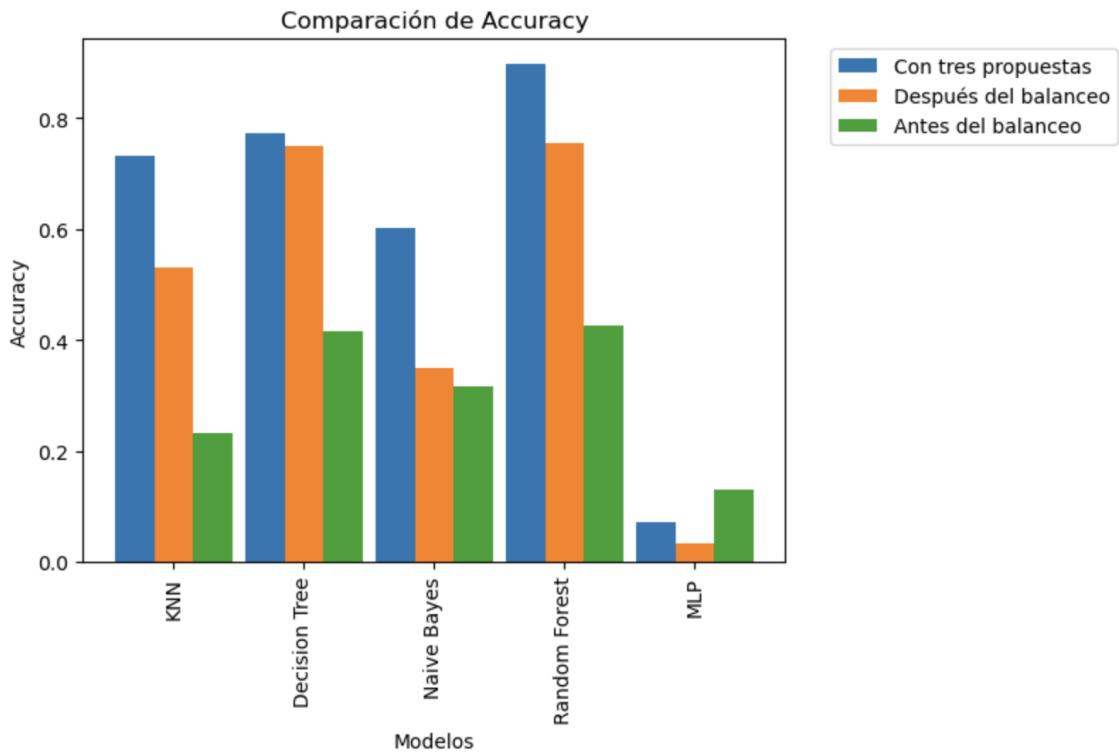
#Evaluar el modelo
accuracy = aciertos / len(y_pred_classes)
m4=accuracy
print(f'Accuracy global: {accuracy:.2f}')


Instancia 25307: [30 49 23] - Resultado Real:12
Instancia 25308: [37 2 28] - Resultado Real:28
Instancia 25309: [69 73 62] - Resultado Real:73
Instancia 25310: [ 7 54 57] - Resultado Real:57
Instancia 25311: [28 75 2] - Resultado Real:2
Instancia 25312: [21 28 8] - Resultado Real:8
Instancia 25313: [58 56 50] - Resultado Real:50
Instancia 25314: [27 35 38] - Resultado Real:38
Instancia 25315: [52 39 43] - Resultado Real:43
Instancia 25316: [27 51 16] - Resultado Real:51
Instancia 25317: [45 12 6] - Resultado Real:6
Instancia 25318: [18 47 49] - Resultado Real:47
Instancia 25319: [49 30 12] - Resultado Real:12
Instancia 25320: [20 27 0] - Resultado Real:0
Instancia 25321: [36 23 11] - Resultado Real:11
Instancia 25322: [43 47 5] - Resultado Real:5
Instancia 25323: [75 1 26] - Resultado Real:26
Instancia 25324: [75 13 3] - Resultado Real:3
Accuracy global: 0.90

```

Gráfica 6. Modelo Bosques Aleatorios con tres resultados

Y después graficamos los valores de los modelos con los tres resultados, en comparación con los de una sola predicción con datos balanceados y no balanceados como se muestra en la gráfica 7. Y seleccioné el modelo de Bosques aleatorios porque fue el que obtuvo mayor accuracy.



*Gráfica 7. Comparación de accuracy*

Ya que teníamos el modelo de bosques aleatorios saqué las métricas para evaluarlo, pero usando el de una sola variable para poder sacarlas. Para hacerlo usé las librerías de métricas de sklearn, como accuracy\_score, precision\_score, recall\_score, f1\_score y confusion\_matrix, para evaluar la efectividad y el rendimiento del modelo. Obtuvimos el accuracy, precisión, sensibilidad, f1 score, especificidad y los graficamos, como se observa en la gráfica 8.

```

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix

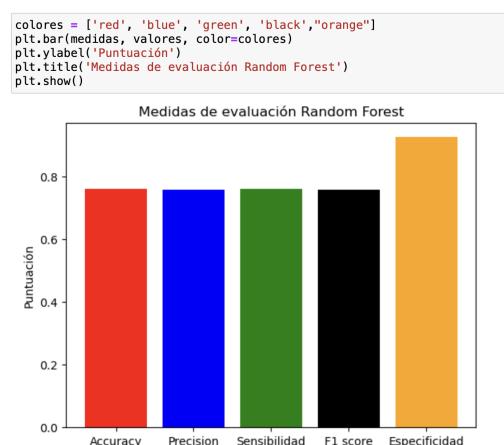
# Calcular métricas
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

tn = matriz[0,0]
fp = matriz.sum(axis=0)[0] - tn
especificidad = tn / (tn + fp)

medidas =['Accuracy','Precision','Sensibilidad','F1 score','Especificidad']
valores=[accuracy,precision,recall,f1,especificidad]
# Imprimir resultados
print(f'Accuracy: {accuracy:.3f}')
print(f'Precision: {precision:.3f}')
print(f'Recall: {recall:.3f}')
print(f'F1 score: {f1:.3f}')
print(f'Especificidad: {especificidad:.3f}')

Accuracy: 0.762
Precision: 0.758
Recall: 0.762
F1 score: 0.758
Especificidad: 0.926

```



*Gráfica 8. Evaluación del modelo Bosques Aleatorios*

### c) Modelos de Aprendizaje Supervisado

1) Elaborar un resumen de modelos de clasificación, modelos de regresión y la diferencia entre ellos.

En cuestión del aprendizaje automático existen los **métodos de clasificación y regresión**, los cuales son técnicas que ayudan y apoyan en la cuestión de análisis y predicciones de una base de datos en una diversa cantidad de entornos.

En materia de los **métodos de clasificación**, estos son usados en torno a la predicción de una categoría en específico y no de un valor numérico. Un ejemplo de clasificación son los correos o e-mails en el cual existen categorías como spam, correos deseados, destacados, etc.

A diferencia de los **métodos de regresión**, que tienen como objetivo principal predecir un valor numérico o continuo en función de un conjunto de características o atributos. Estos modelos se utilizan comúnmente en problemas de predicción, como la estimación de precios, la previsión del clima o el análisis financiero.

Resumiendo, **la principal diferencia** entre los modelos de clasificación y regresión es que los modelos de regresión predicen valores continuos, mientras que los modelos de clasificación predicen categorías o clases. [\[8\]](#)

#### **¿Qué tipo de modelos se aplicarán, clasificación o regresión? ¿Por qué?**

Es por eso que en esta situación vamos a aplicar **modelos de clasificación**, porque la variable que queremos predecir no es continua, sino que se trata de un tipo de clase, en este caso, la empresa transportista.

### **2) Describir brevemente los modelos de aprendizaje automático que se usarán, incluyendo los principales parámetros que se definen y ventajas/desventajas de cada uno.**

#### **1. K-Vecinos:**

El algoritmo de k-vecinos más cercanos, también conocido como k-NN (por sus siglas en inglés), es un método de clasificación y regresión en el aprendizaje automático. Este algoritmo se basa en encontrar los k puntos de datos más cercanos a un punto de datos individual y usarlos para predecir la clasificación o el valor de ese punto. El valor de k se puede ajustar para mejorar la precisión del modelo.

Según IBM (2021), el algoritmo k-NN es ampliamente utilizado en aplicaciones como la clasificación de correos electrónicos no deseados, la detección de fraudes y la recomendación de productos. El algoritmo también se puede utilizar para la imputación de valores faltantes en conjuntos de datos. [\[9\]](#)

## **2. SVC (Support Vector Classification):**

Las SVC (Support Vector Classification) son un tipo de modelo de clasificación utilizado en el contexto de las Máquinas de Vectores de Soporte (SVM), según lo explicado en el artículo "Support Vector Machines (SVM)" de UniPython.

Las SVC se basan en la identificación de un hiperplano óptimo que separa las clases de puntos de datos, y utilizan un conjunto de vectores de soporte para determinar la ubicación del hiperplano. El objetivo es maximizar la distancia entre los vectores de soporte y el hiperplano, lo que se conoce como la margen máxima.

Las SVC pueden ser lineales o no lineales, y para resolver problemas no lineales, las SVC utilizan la técnica del "kernel trick" para transformar los datos de entrada en un espacio de características de mayor dimensión donde se pueden separar de manera lineal. [\[10\]](#)

## **3. Regresión logística:**

La regresión logística es un método de aprendizaje automático supervisado utilizado para resolver problemas de clasificación binaria.

La regresión logística utiliza una función logística para modelar la probabilidad de que una instancia de entrada pertenezca a una de las dos clases. El objetivo del modelo es encontrar los valores óptimos de los parámetros para la función logística que mejor se ajusten a los datos de entrenamiento.

La regresión logística puede ser utilizada tanto para problemas lineales como no lineales, y puede manejar variables de entrada continuas y categóricas. Además, puede ser mejorada mediante el uso de técnicas de regularización para evitar el sobreajuste. [\[11\]](#)

## **4. Random Forest**

Los Random Forest son un tipo de algoritmo de aprendizaje automático basado en árboles de decisión, que se utilizan tanto para problemas de clasificación como de regresión, según el artículo "Understanding Random Forest" de Towards Data Science.

El Random Forest construye múltiples árboles de decisión a partir de datos de entrenamiento, y luego combina las predicciones de cada árbol para obtener una predicción final. Cada árbol se construye utilizando un subconjunto aleatorio de las características de entrada y un subconjunto aleatorio de las instancias de entrenamiento.

El Random Forest tiene varias ventajas, como la capacidad de manejar grandes conjuntos de datos, la resistencia al sobreajuste, la capacidad de manejar tanto variables continuas como categóricas, y la capacidad de proporcionar información sobre la importancia de las características de entrada. [\[12\]](#)

## 5. Decision Tree Classifier

El Decision Tree Classifier es un algoritmo de aprendizaje automático que se utiliza en la clasificación de datos. Es un árbol de decisiones que se construye a partir de un conjunto de datos de entrenamiento y se usa para predecir el valor objetivo de un nuevo conjunto de datos.

El árbol de decisión se divide en ramas y nodos, donde cada nodo representa una característica del conjunto de datos. Los nodos se dividen en ramas que representan los posibles resultados de esa característica.

El algoritmo de árbol de decisión es útil cuando los datos tienen una estructura jerárquica y es posible dividir los datos en diferentes grupos basados en diferentes características.

El proceso de construcción del árbol de decisión implica la selección de la mejor característica para dividir los datos y la creación de nodos y ramas adicionales. El árbol se sigue dividiendo hasta que se logra la máxima precisión de predicción. [\[13\]](#)

## 6. Naive Bayes

Naive Bayes se basa en la probabilidad condicional para clasificar instancias en diferentes categorías. Este algoritmo se basa en el teorema de Bayes y asume que todas las características son independientes entre sí.

Dado un conjunto de características, se puede calcular la probabilidad de cada clase y elegir la clase con la probabilidad más alta.

### d) Generación y Evaluación de los Modelos de Aprendizaje

1.- Por cada miembro del equipo, incluir imágenes de la aplicación de modelos de aprendizaje automático y los resultados obtenidos

Considerando los resultados obtenidos por cada miembro del equipo al generar y evaluar, individualmente, modelos de aprendizaje en su propio notebook en Python, generar una tabla incluyendo, para cada modelo aplicado, el tipo de modelo, valor de parámetros utilizados (diferentes entre los miembros del equipo) y resultados de evaluación de la métrica de

exactitud (accuracy). La tabla debe seguir el siguiente formato, dependiendo del número de alumnos por equipo.

	Kevin	Luis	Adrián	Alexis
<b>Regresión Logística</b>	Parámetros: cv=5  accuracy: <b>13%</b>	Parámetros: 20% de prueba random_state: 5 max_iter=7000  accuracy: <b>36%</b>	(Decidí intentar con 5 Clases) Parámetros: 10% de prueba random_state: 15 max_iter=5000  accuracy: <b>49%</b>	Parámetros: 15% de prueba random_state: 8 max_iter=4000  accuracy: <b>37%</b>
<b>Árboles de decisión</b>	Parámetros: criterion = 'entropy'  accuracy: <b>78%</b>	Parámetros: 20% de prueba random_state: 5 criterion='entropy'  accuracy: <b>43%</b>	Parámetros: 20% de prueba random_state: 20 criterion='entropy'  accuracy: <b>45%</b>	Parámetros: 15% de prueba random_state: 30 criterion='entropy'  accuracy: <b>45%</b>
<b>Bosques Aleatorios</b>	Parámetros: n_estimators = 100, criterion = 'entropy'  accuracy: <b>90%</b>	Parámetros: 20% de prueba random_state: 5 n_estimators:30  accuracy: <b>65%</b>	Parámetros: 20% de prueba random_state: 20 n_estimators: 50  accuracy: <b>73%</b>	Parámetros: 20% de prueba random_state: 20 n_estimators:70  accuracy: <b>67%</b>
<b>Máquinas de Soporte Vectorial</b>	Parámetros: test_size=0.2, random_state=42  accuracy: <b>18%</b>	Parámetros: 20% de prueba probability:True random_state=5  accuracy: <b>35%</b>	Parámetros: 20% de prueba probability:True random_state=5  accuracy: <b>49%</b>	Parámetros: 20% de prueba probability:True random_state=20  accuracy: <b>41%</b>
<b>KNN Vecinos</b>	Parámetros: n_neighbors = 5 metric = 'minkowski', p = 2  accuracy: <b>74%</b>	Parámetros: 20% de prueba n_neighbors=6  accuracy: <b>41%</b>	Parámetros: 20% de prueba n_neighbors=5  accuracy: <b>49%</b>	Parámetros: 20% de prueba n_neighbors=5  accuracy: <b>41%</b>

<b>Naives Bayes</b>	Parámetros: GaussianNB()  accuracy: <b>61%</b>	Parámetros: 20% de prueba  accuracy: <b>54%</b>	Parámetros: 20% de prueba  accuracy: <b>62%</b>	Parámetros: 20% de prueba  accuracy: <b>52%</b>
---------------------	---	--	--	--

2.- Señalar en la tabla, para cada tipo de modelo, el mejor resultado obtenido.

3.- Análisis de resultados y selección del modelo de predicción.

## Ajuste hiperparámetros

```
from sklearn.model_selection import GridSearchCV
rfc = RandomForestClassifier()

# Definir los hiperparámetros que queremos ajustar
param_grid = {'n_estimators': [10, 50, 100, 150],
              'criterion': ['gini', 'entropy']}

# Crear el objeto GridSearchCV
grid_search = GridSearchCV(estimator = rfc, param_grid = param_grid,
                           cv = 5, n_jobs = -1, verbose = 2)

# Ajustar el objeto GridSearchCV a los datos
grid_search.fit(X, y)

# Imprimir los mejores parámetros encontrados
print(grid_search.best_params_)

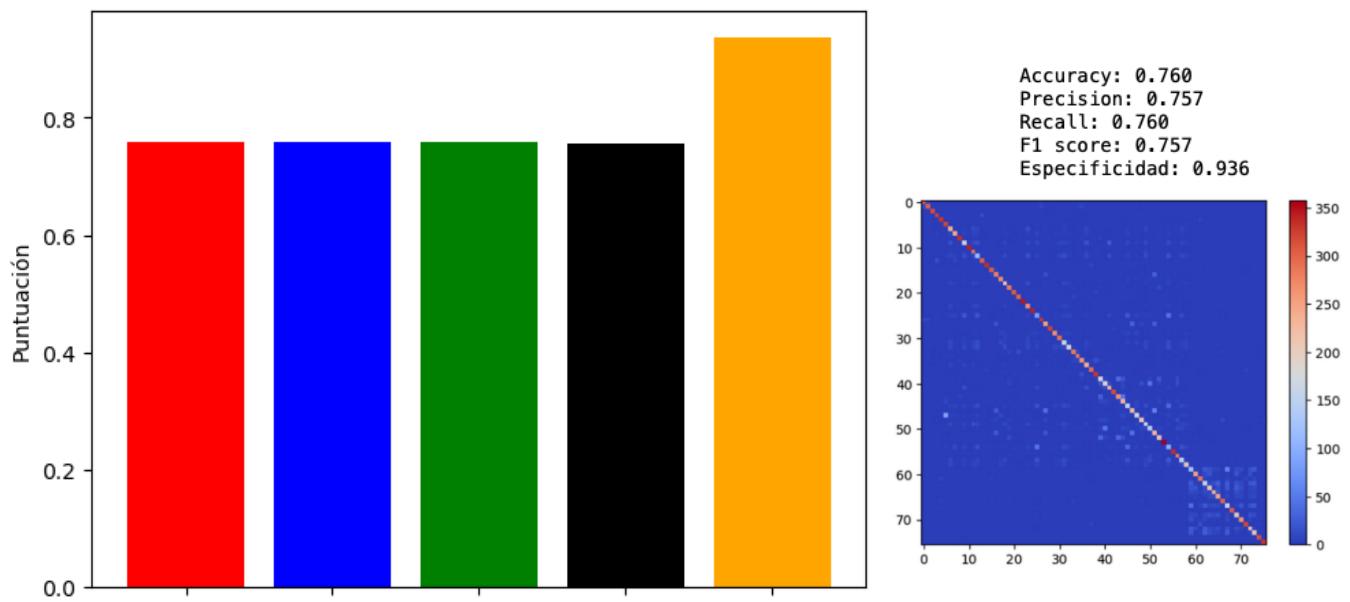
Fitting 5 folds for each of 8 candidates, totalling 40 fits
{'criterion': 'gini', 'n_estimators': 150}
```

*Gráfica Ajuste de hiperparámetros*

En la *Gráfica Ajuste de hiperparámetros* primero, importamos GridSearchCV de la biblioteca sklearn.model\_selection, que sirve para encontrar los mejores hiperparámetros para un modelo de aprendizaje automático, en este caso lo usamos para optimizar los hiperparámetros del modelo RandomForestClassifier y creamos una instancia de RandomForestClassifier sin ningún hiperparámetro específico. Luego, definimos los hiperparámetros que deseamos ajustar en "param\_grid". En este caso, estamos ajustando el número de árboles que se usan en el bosque aleatorio (n\_estimators) y el criterio que se usa para dividir los nodos del árbol (criterion). Después de esto, creamos un objeto GridSearchCV, que sirve para encontrar los mejores hiperparámetros para un modelo de aprendizaje automático, en este caso lo usamos para optimizar los hiperparámetros del modelo RandomForestClassifier y el diccionario de parámetros que definimos anteriormente. También especificamos otros argumentos como "cv = 5" que representa la cantidad de veces que se dividirá el conjunto de datos para realizar la validación cruzada, "n\_jobs = -1" que significa que se utilizarán todos los núcleos de la CPU disponibles para realizar el cálculo de manera más rápida, y "verbose = 2" que muestra

información detallada sobre el progreso de la búsqueda. Finalmente, ajustamos el objeto GridSearchCV a nuestros datos y utilizamos la función "print(grid\_search.best\_params\_)" para imprimir los mejores parámetros encontrados por la búsqueda en cuadrícula. Estos son los hiperparámetros que dan el mejor rendimiento para nuestro modelo de RandomForestClassifier en los datos proporcionados.

Luego calculamos las métricas del modelo, y como se observa en la Gráfica de Medidas de evaluación Random Forest se observa que el modelo tiene un accuracy del 0.760, lo que significa que aproximadamente el 76% de las predicciones fueron correctas. Tiene una precisión del 0.757, lo que significa que aproximadamente el 76% de las predicciones positivas realizadas por el modelo fueron verdaderos positivos. Tiene un recall del 0.760, lo que significa que aproximadamente el 76% de los valores verdaderos positivos fueron identificados correctamente por el modelo. También tiene un F1 score del 0.757, lo que indica que hay un equilibrio entre la precisión y el recall. Y finalmente cuenta con una especificidad del 0.936, lo que significa que aproximadamente el 94% de los valores verdaderos negativos fueron identificados correctamente por el modelo.



*Gráfica de Medidas de evaluación Random Forest*

# Etapa 4: Selección y Despliegue

Para una mayor facilidad de la empresa para hacer las predicciones y garantizar su privacidad, decidimos que lo más factible era que la empresa ingresara el archivo con los datos a los que desea predecir la empresa transportista y al aplicar el código retornar un archivo que incluya las 5 empresas con mayor probabilidad de parecerse a un caso real de acuerdo a los datos proporcionados.

En la Gráfica Ingreso de datos, se observa la parte del código donde se le pregunta qué archivo quiere leer y el nombre del archivo como desea que se guarden las predicciones. Ya que se cargó el archivo se muestra el archivo que cargó para corroborar que sea el que quería leerse.

## Ingreso de datos

```
leer_archivo = input("Por favor, escribe el nombre del archivo que quieras analizar: ")
archivo_prediccciones=input("Por favor, escribe el nombre donde quieras guardar las predicciones: ")

Por favor, escribe el nombre del archivo que quieras analizar: prueba_1.csv
Por favor, escribe el nombre donde quieras guardar las predicciones: predicciones_1.csv
```

## Archivo que se desea predecir

```
prueba=pd.read_csv(leer_archivo)
x= prueba.shape[0]
prueba
```

	PLANTAORIGEN_Pesqueria	D_CLIENTE	PESO_NETO	ESTADO	CIUDAD	TIPOTRANSPORTE	target	EMPRESA_TRANSPORTISTA
0		1	187	30670	1	39	9	20
1		1	107	36890	8	35	5	50
2		1	145	23300	14	61	6	39
3		1	147	20480	22	1	5	15
4		1	99	36980	1	39	5	40
...	...	...	...	...	...	...	...	...
126		1	99	28300	1	39	5	66
127		1	227	21240	14	42	5	57
128		1	28	34680	14	61	5	9
129		1	227	32620	14	42	6	52
130		1	145	38230	14	61	5	54

Gráfica Ingreso de datos

Después en la Gráfica Cálculo de predicciones, lo que se hace es obtener del archivo cada fila que se quiere predecir obteniendo las variables desde v1 que representa la planta de origen, gasta v6 que representa el tipo de transporte, este proceso lo metemos en un ciclo for, para que se haga la predicción para cada fila del archivo. Ya que se tienen estas variables las metemos en el algoritmo.predic\_proba para que nos de las probabilidades de las predicciones,

ordenamos de mayor probabilidad a menor probabilidad y obtuvimos las 5 más altas, y retornamos el nombre de estas empresas. Ya que tenemos la lista de todas las predicciones, creamos un nuevo archivo que tenga el original agregando las columnas de las 5 predicciones de las empresas transportistas y guardamos el archivo con el nombre que había elegido el usuario anteriormente en la misma carpeta.

## Cálculo de predicciones

```

prediccciones = pd.DataFrame(0, index=range(x), columns=['Prediccion1', 'Prediccion2', 'Prediccion3', 'Prediccion4', 'Prediccion5'])

for n in range(x):
    v1=prueba.iloc[n,0]
    v2=prueba.iloc[n,1]
    v3=prueba.iloc[n,2]
    v4=prueba.iloc[n,3]
    v5=prueba.iloc[n,4]
    v6=prueba.iloc[n,5]

    valores = [[v1,v2,v3,v4,v5,v6]]

    probabilidades = algoritmo.predict_proba(valores)
    orden_probabilidades = np.argsort(probabilidades[0])[::-1]

    prediccciones.iloc[n,0] =name[name["target"] == orden_probabilidades[0]].iloc[0,7]
    prediccciones.iloc[n,1] =name[name["target"] == orden_probabilidades[1]].iloc[0,7]
    prediccciones.iloc[n,2] =name[name["target"] == orden_probabilidades[2]].iloc[0,7]
    prediccciones.iloc[n,3] =name[name["target"] == orden_probabilidades[3]].iloc[0,7]
    prediccciones.iloc[n,4] =name[name["target"] == orden_probabilidades[4]].iloc[0,7]

prediccion= pd.concat([prueba, prediccciones], axis=1)
prediccion.to_csv(archivo_prediccciones, index=False)
prediccion

```

	PLANTAORIGEN_Pesqueria	D_CLIENTE	PESO_NETO	ESTADO	CIUDAD	TIPOTRANSPORTE	target	EMPRESA_TRANSPORTISTA	Prediccion1	Prediccion2	Prediccion3	Prediccion4	Prediccion5
0	1	187	30670	1	39		9	20	TRANSPORTE31	TRANSPORTE84	TRANSPORTE63	TRANSPORTE26	TRANSPORTE53
1	1	107	36890	8	35		5	50	TRANSPORTE63	TRANSPORTE8	TRANSPORTE57	TRANSPORTE26	TRANSPORTE53
2	1	145	23300	14	61		6	39	TRANSPORTE52	TRANSPORTE57	TRANSPORTE63	TRANSPORTE26	TRANSPORTE53
3	1	147	20480	22	1		5	15	TRANSPORTE26	TRANSPORTE26	TRANSPORTE63	TRANSPORTE26	TRANSPORTE53
4	1	99	36980	1	39		5	40	TRANSPORTE53	TRANSPORTE53	TRANSPORTE63	TRANSPORTE26	TRANSPORTE53
...	...	...	...	...	...		...	...	...	...	...	...	...
126	1	99	28300	1	39		5	66	TRANSPORTE8	TRANSPORTE8	TRANSPORTE63	TRANSPORTE26	TRANSPORTE53
127	1	227	21240	14	42		5	57	TRANSPORTE70	TRANSPORTE70	TRANSPORTE63	TRANSPORTE26	TRANSPORTE53
128	1	28	34680	14	61		5	9	TRANSPORTE19	TRANSPORTE38	TRANSPORTE63	TRANSPORTE26	TRANSPORTE53
129	1	227	32620	14	42		6	52	TRANSPORTE66	TRANSPORTE66	TRANSPORTE63	TRANSPORTE26	TRANSPORTE53
130	1	145	38230	14	61		5	54	TRANSPORTE68	TRANSPORTE68	TRANSPORTE63	TRANSPORTE26	TRANSPORTE53

Gráfica Cálculo de predicciones

## 5) Conclusiones y Recomendaciones

Resumen del proceso:

- Establecimiento del objetivo
- Integración de los datos
- Limpieza de datos
- Transformación de los datos
- Selección de variables más relevantes
- Aplicación de los modelos de aprendizaje supervisado
- Métricas de evaluación
- Despliegue de los resultados

En resumen, los modelos de aprendizaje supervisado por clasificación son algoritmos utilizados para predecir la clase o categoría de una variable objetivo discreta. Son adecuados para variables objetivo categóricas.

En nuestro caso, debido a que debemos predecir los mejores transportistas (categorías) a efectuar el viaje, estos modelos de clasificación nos sirvieron en gran medida para lograrlo. Algunos ejemplos de algoritmos de clasificación que usamos fueron la Regresión Logística, el Árbol de Decisión, el Bosque Aleatorio, el SVM y el K-NN.

Los modelos de clasificación son ampliamente utilizados en una variedad de aplicaciones, como detección de spam, diagnóstico médico, clasificación de imágenes y muchas otras áreas donde se requiere la clasificación de datos en categorías discretas. Los modelos de clasificación nos permiten clasificar datos en categorías o clases específicas, lo que los convierte en una herramienta valiosa para resolver una amplia gama de problemas en diferentes campos, como lo es en esta situación problema y también nos ayudan a saber si elegimos adecuadamente nuestras variables, sabemos que cada modelo puede ser mejor que otro dependiendo de la base de datos, pero en general nos dan una idea sobre cómo poder trabajar con nuestros datos.

Recomendaciones al socio formador:

- Un mayor énfasis en la captura de datos.
- Eficientizar otros procesos de logística con modelos de aprendizaje supervisado.

Recomendaciones de pasos a seguir de este proyecto:

Realizar una mejor interfaz gráfica que se adapte al empleado. Una app podría ser una excelente vía.

## 6. Referencias bibliográficas

- [1] Quiénes somos | Acero Ternium. (2021). Ternium.com.  
<https://ar.ternium.com/es/nuestra-empresa#:~:text=Ternium%20es%20el%20{mayor%20fabricante,calidad%2C%20seguridad%20y%20medio%20ambiente>
- [2] Supelano, K. L. (2015). Modelo de automatización de procesos para un sistema de gestión a partir de un esquema de documentación basado en Business Process Management (bpm). Universidad & Empresa, 17(29), 131-155
- [3] Ternium - Información General. (2018). Ternium.com.  
<https://investors.ternium.com/Spanish/centro-de-inversores-ternium/general-information/default.aspx>

- [4] Ruiz, R. (2020). Optimización de rutas y logística. Del reto a la oportunidad. Investigate to innovate.  
<https://www.iti.es/blog/optimizacion-de-rutas-y-logistica-del-reto-a-la-oportunidad>
- [5] Granillo-Macías R., 2020. Inventory management and logistics optimization: A Data Mining practical approach. LogForum 16 (4), 535-547, <http://doi.org/10.17270/J.LOG.2020.512>
- [6] Vega, David (2018). Optimización con Algoritmos Genéticos en Python. Recuperado de: [https://www.cienciadedatos.net/documentos/py01\\_optimizacion\\_ga](https://www.cienciadedatos.net/documentos/py01_optimizacion_ga)
- [7] Lin, H. E., Zito, R., & Taylor, M. (2005, September). A review of travel-time prediction in transport and logistics. In Proceedings of the Eastern Asia Society for transportation studies (Vol. 5, pp. 1433-1448).  
[https://www.researchgate.net/profile/Rocco-Zito/publication/237632154\\_A\\_review\\_of\\_travel-time\\_prediction\\_in\\_transport\\_and\\_logistics/links/0f31753166c0fde774000000/A-review-of-travel-time-prediction-in-transport-and-logistics.pdf](https://www.researchgate.net/profile/Rocco-Zito/publication/237632154_A_review_of_travel-time_prediction_in_transport_and_logistics/links/0f31753166c0fde774000000/A-review-of-travel-time-prediction-in-transport-and-logistics.pdf)
- [8] Cleverdata.io. (s.f.). Conceptos básicos de Machine Learning.  
<https://cleverdata.io/conceptos-basicos-machine-learning/#:~:text=Un%20sistema%20de%20clasificaci%C3%B3n%20predice,%E2%80%9D%20o%20como%20%E2%80%9Cleg%C3%A9timos%E2%80%9D>
- [9] IBM. (2021). Algoritmo de k vecinos más cercanos (KNN).  
<https://www.ibm.com/mx-es/topics/knn#:~:text=El%20algoritmo%20de%20k%20vecinos%20m%C3%A1s%20cercanos%20tambi%C3%A9n%20conocido%20como,un%20punto%20de%20datos%20individual>.
- [10] uniPython. (s.f.). Support Vector Machines (SVM).  
<https://unipython.com/support-vector-machines-svm/>
- [11] Amazon Web Services. (s.f.). ¿Qué es la regresión logística?  
<https://aws.amazon.com/es/what-is/logistic-regression/>
- [12] Towards Data Science. (2019, March 8). Understanding Random Forest.  
<https://towardsdatascience.com/understanding-random-forest-58381e0602d2>
- [13] DataCamp. (s.f.). Decision Tree Classification in Python.  
<https://www.datacamp.com/tutorial/decision-tree-classification-python>
- [14] Lin, H. E., Zito, R., & Taylor, M. (2005, September). A review of travel-time prediction in transport and logistics. In Proceedings of the Eastern Asia Society for transportation studies (Vol. 5, pp. 1433-1448).
- [https://www.researchgate.net/profile/Rocco-Zito/publication/237632154\\_A\\_review\\_of\\_travel-time\\_prediction\\_in\\_transport\\_and\\_logistics/links/0f31753166c0fde774000000/A-review-of-travel-time-prediction-in-transport-and-logistics.pdf](https://www.researchgate.net/profile/Rocco-Zito/publication/237632154_A_review_of_travel-time_prediction_in_transport_and_logistics/links/0f31753166c0fde774000000/A-review-of-travel-time-prediction-in-transport-and-logistics.pdf)

## 7. Anexos

Alumno	Título artículo	Resumen	Referencia
Kevin González	A REVIEW OF TRAVEL-TIME PREDICTION IN TRANSPORT AND LOGISTICS	<p>El artículo A REVIEW OF TRAVEL-TIME PREDICTION IN TRANSPORT AND LOGISTICS habla sobre la importancia de la información de tiempo de viaje y cómo puede ser utilizada en diferentes campos y propósitos. Nos dice que la información de tiempo de viaje puede ayudar a los viajeros a ahorrar tiempo y mejorar la fiabilidad al seleccionar rutas de viaje antes del viaje y durante el mismo. También dice que en la aplicación de la logística, la información de tiempo de viaje puede reducir los costos de entrega, aumentar la fiabilidad de la entrega y mejorar la calidad del servicio. Además, nos indica que la información de tiempo de viaje es un índice importante para la operación del sistema de tráfico y que se puede predecir mediante varios métodos y utilizando diferentes datos de entrada. El objetivo del artículo es proporcionar una revisión sistemática y comparación de las diferentes técnicas de predicción de tiempo de viaje y ampliar la perspectiva de investigación en esta área para futuros investigadores.</p>	[14] Lin, H. E., Zito, R., & Taylor, M. (2005, September). A review of travel-time prediction in transport and logistics. In Proceedings of the Eastern Asia Society for transportation studies (Vol. 5, pp. 1433-1448). <a href="https://www.researchgate.net/profile/Rocco-Zito/publication/237632154_A_review_of_travel-time_prediction_in_transport_and_logistics/links/0f31753166c0fde77400000/A-review-of-travel-time-prediction-in-transport-and-logistics.pdf">https://www.researchgate.net/profile/Rocco-Zito/publication/237632154_A_review_of_travel-time_prediction_in_transport_and_logistics/links/0f31753166c0fde77400000/A-review-of-travel-time-prediction-in-transport-and-logistics.pdf</a>
Luis López	Optimización de rutas y logística. Del reto a la oportunidad.	<p>El artículo "Optimización de rutas y logística: del reto a la oportunidad" de ITI aborda la importancia de la optimización de rutas en la logística y cómo esta práctica puede generar importantes beneficios para las empresas. Abarca los desafíos dentro del ámbito de logística, describe cómo la optimización puede superar estos desafíos, destaca los beneficios de la optimización de rutas y finalmente, señala la oportunidad que ésta representa para las empresas.</p>	[4] Ruiz, R. (2020). Optimización de rutas y logística. Del reto a la oportunidad. Investigate to innovate. <a href="https://www.iti.es/blog/optimizacion-de-rutas-y-logistica-del-reto-a-la-oportunidad">https://www.iti.es/blog/optimizacion-de-rutas-y-logistica-del-reto-a-la-oportunidad</a>

Alexis Leyva	Información General de Ternium	<p>El artículo habla que Ternium es una empresa dedicada a la producción de acero en diferentes países de América Latina. En ella se puede encontrar información general sobre la empresa, sus valores, su estrategia y su compromiso con la sustentabilidad. Así como un resumen de la historia de la empresa, su estructura organizacional, sus principales actividades, sus estados financieros y sus informes anuales.</p>	<p>[3] Ternium - Información General. (2018). Ternium.com. <a href="https://investors.ternium.com/Spanish/centro-de-inversores-ternium/general-information/default.aspx">https://investors.ternium.com/Spanish/centro-de-inversores-ternium/general-information/default.aspx</a></p>
	Optimización con algoritmo genético	<p>El artículo comienza explicando qué es la optimización y para qué se utiliza en el ámbito de la ciencia de datos. A continuación, se introduce la teoría de los algoritmos genéticos y se explica cómo funcionan, posteriormente cómo se implementa un algoritmo genético en Python, paso a paso y se muestra un ejemplo práctico de optimización mediante algoritmos genéticos. En concreto, se trata de un problema de optimización de funciones matemáticas. Finalmente, se presentan algunas consideraciones y consejos a tener en cuenta a la hora de utilizar algoritmos genéticos para optimización en Python.</p>	<p>[6] Vega, David (2018). Optimización con Algoritmos Genéticos en Python: <a href="https://www.cienciadedatos.net/documentos/py01_optimizacion_ga">https://www.cienciadedatos.net/documentos/py01_optimizacion_ga</a></p>

Adrian Pineda Sánchez	“Inventory management and logistics optimization”	<p>En este artículo se expone de acuerdo a un caso de una industria de alimentos, en el área de logística e inventario, como se propone una mejora a la clasificación del mismo en términos de variables cualitativas y cuantitativas a través de la utilización de métodos de minería de datos, así como técnicas de localización basándonos en el algoritmo “Partitioning Around Medoids” que aplica métodos de clustering para la optimización de recursos y tiempo en el modelo predictivo.</p> <p>Donde se obtuvo que los resultados mejoraron las rutas de viaje, así como la eficiencia en tiempo y en los suministros de carga y optimización de entrega de producto en cada viaje, mejorando la toma de decisiones a través de la automatización por análisis de datos.</p>	<p>[5] Granillo-Macías R., 2020. Inventory management and logistics optimization: A Data Mining practical approach. LogForum 16 (4), 535-547, <a href="http://doi.org/10.17270/J.LOG.2020.512">http://doi.org/10.17270/J.LOG.2020.512</a></p>
-----------------------------	--	--	---