1. **Introducción**

El comercio electrónico ha experimentado una transformación significativa en los últimos cinco años, consolidándose como un canal fundamental para las empresas a nivel global. Entre los últimos años, con la evolución tecnológica, los cambios en los hábitos de consumo y la aceleración provocada por la pandemia de COVID-19 han sido factores determinantes. Las empresas han adoptado soluciones digitales avanzadas, como inteligencia artificial, análisis de datos y plataformas multicanal, para mejorar la experiencia del cliente. Además, se ha incrementado la competencia en el sector, con la entrada de nuevos actores y la diversificación de modelos de negocio, como el comercio social y las suscripciones personalizadas.

En este periodo, los consumidores han mostrado una creciente preferencia por las compras en línea debido a la comodidad, la personalización y la disponibilidad de opciones. Este cambio ha impulsado a las empresas a desarrollar estrategias de marketing digital más sofisticadas, integrando SEO, redes sociales y campañas automatizadas. Paralelamente, la importancia de la logística y las entregas rápidas se ha intensificado, fomentando la colaboración con terceros. Este contexto ha consolidado el comercio electrónico como un pilar estratégico para las empresas, tanto para capturar nuevos mercados como para adaptarse a las expectativas cambiantes de los consumidores.

La empresa IA DELIVERY S.L. con sede en Madrid, se dedica a la manufactura de productos alimenticios perecederos de corta duración. Han identificado que una parte significativa de sus costes operativos proviene de la externalización de servicio de distribución a terceros. Tras realizar un análisis exhaustivo de estos costes, han decidido internalizar este proceso y crear su propia flota de vehículos, optimizando así la distribución de sus productos desde sus instalaciones centrales.

Con el fin de reducir costos y mejorar la eficiencia, la empresa busca desarrollar una herramienta de optimización de rutas. Esta herramienta deberá calcular las rutas más eficientes para la distribución de sus productos, teniendo en cuenta factores como la cantidad mínima de vehículos necesaria, las distancias a recorrer, el tiempo de entrega y la capacidad de los vehículos. El objetivo final es minimizar el uso de recursos y garantizar una distribución efectiva con la menor inversión de flota propia.

1. **Descripción general**
   1. **Objetivos**
      1. **Objetivo general**

Se busca minimizar los costes totales generados por el reparto de artículos vendidos y maximizar el número de entregas con la flota de vehículos disponibles, con el objetivo de alcanzar el máximo beneficio disponible.

* + 1. **Objetivos específicos**
* Realizar el análisis del histórico de datos de entregas y clientes de la empresa para conocer el contexto de ventas de la empresa.
* Desarrollar un modelo que planifique las rutas de entrega, minimizando el coste total y maximizando el número de entregas realizadas cumpliendo las restricciones de capacidad, autonomía y vehículos disponibles.
* Suponiendo que la empresa tenga que llevar acabo una reducción de su flota de vehículos disponibles para la entrega de pedidos (por causas de economía o mantenimiento), optimizar las rutas para mantener el máximo de los beneficios.
* Desarrollar un modelo de demanda de pedidos que predice el número de pedidos esperados para el próximo mes y desarrollar las rutas para cumplir con la demanda prevista y ajustarse a los pedidos futuros.
  1. **Business Criteria**

Para conocer si con el paso del tiempo el modelo está funcionando correctamente, se podrá comparar los datos de valores de tiempo y distancias entre las rutas, teniendo en cuenta que se busca el mínimo de tiempo invertido para la entrega de pedidos, y el máximo de entregas realizadas.

1. **Data understanding**

Proporciona la base para tomar decisiones informadas sobre cómo procesar, transformar y analizar los datos en las etapas posteriores del proyecto. Una comprensión sólida de los datos asegura un análisis más efectivo y resultados de mayor calidad. Para el proyecto nos ceñiremos a los datos proporcionados únicamente.

* 1. **Data exploration report**

Para el desarrollo del proyecto, disponemos de 6 archivos .csv con el contenido de datos a estudiar para la resolución y obtención del objetivo.

* **df\_distance\_km:** contiene los datos entre las localizaciones en km

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo** | **Descripción** |
| cliente\_k | Numérico | Contiene la distancia de Cliente\_k a Cliente\_X |
| almacen | Numérico | Contiene la distancia de Almacen a Cliente\_X |

* **df\_distance\_min:** contiene las datos de tiempo entre las localizaciones en minutos

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo** | **Descripción** |
| cliente\_k | Numérico | Contiene el tiempo de Cliente\_k a Cliente\_X |
| almacen | Numérico | Contiene el tiempo de Almacen a Cliente\_X |

* **df\_historic\_order\_demand:** registro del histórico de pedidos realizados

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo** | **Descripción** |
| cliente | Categórica | Contiene el cliente que realiza el pedido |
| mes\_anio | Fecha | Contiene el dato de mes y año en que se realiza el pedido |
| order\_demand | Numérico | Contiene el peso en kg del pedido |

* **df\_location:** contiene los datos de localizaciones

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo** | **Descripción** |
| cliente | Categórica | Contiene el cliente que realiza el pedido |
| latitud | Numérico | Contiene el valor de la latitud del cliente |
| longitud | Numérico | Contiene el valor de la longitud del cliente |

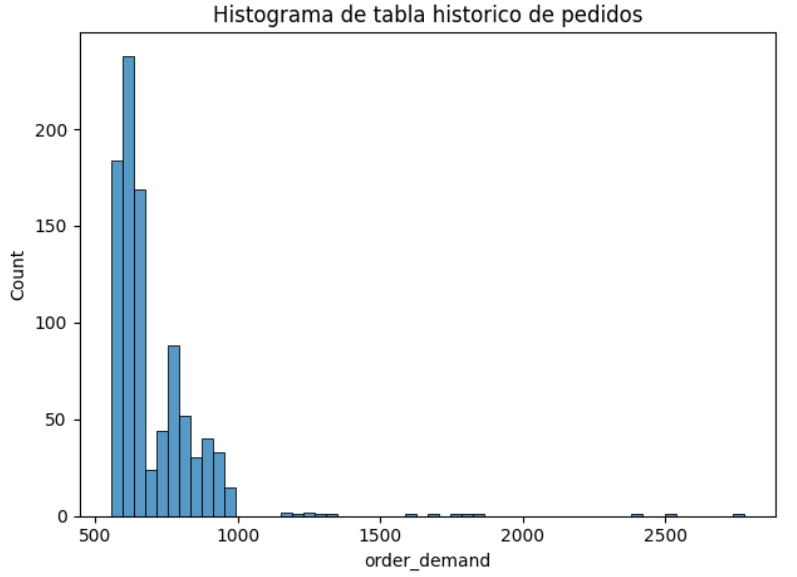
* **df\_orders:** registro del mes de diciembre de pedidos realizados

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo** | **Descripción** |
| cliente | Categórica | Contiene el cliente que realiza el pedido |
| mes\_anio | Fecha | Contiene el dato de mes y año en que se realiza el pedido |
| order\_demand | Numérico | Contiene el peso en kg del pedido |

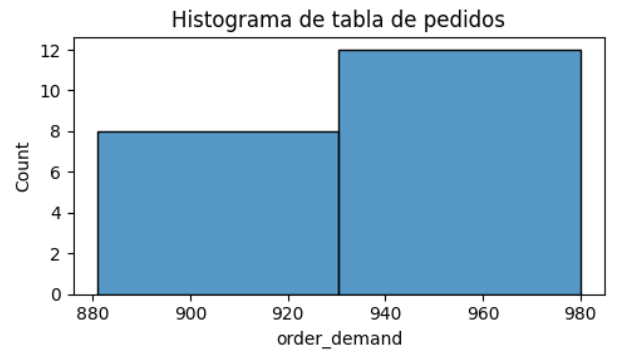
* **df\_vehicle:** contiene los datos de los vehículos disponibles

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo** | **Descripción** |
| vehiculo\_id | Numérica | Contiene el valor del id único del vehículo |
| capacidad\_kg | Numérica | Contiene el valor de la capacidad máxima que permite |
| costo\_km | Numérico | Contiene el valor de coste por km de cada vehículo |
| autonomia\_km | Numérico | Contiene el valor de la autonomía disponible de cada vehículo |

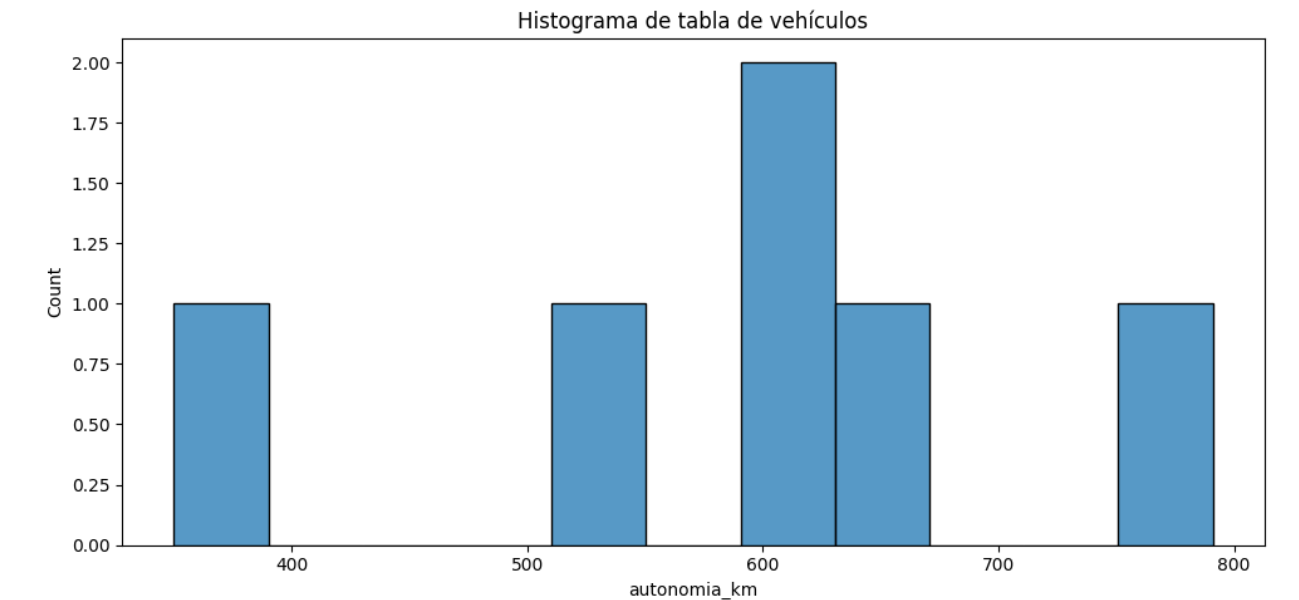
* 1. **Infographics**
* **Histograma de tablas historico de pedidos:** Encontramos la comparación entre el peso y la cantidad del pedido de los clientes, se observa que la mayoria de pedidos, tienen un peso entre 500 y 1000 kg.



* **Historico de tabla de pedidos:** Encontramos la comparación entre el peso y la cantidad de los pedidos realizados por los clientes en el último mes y se observa que la mayoria de pedidos, tienen un peso cercano a los 1000 kg.

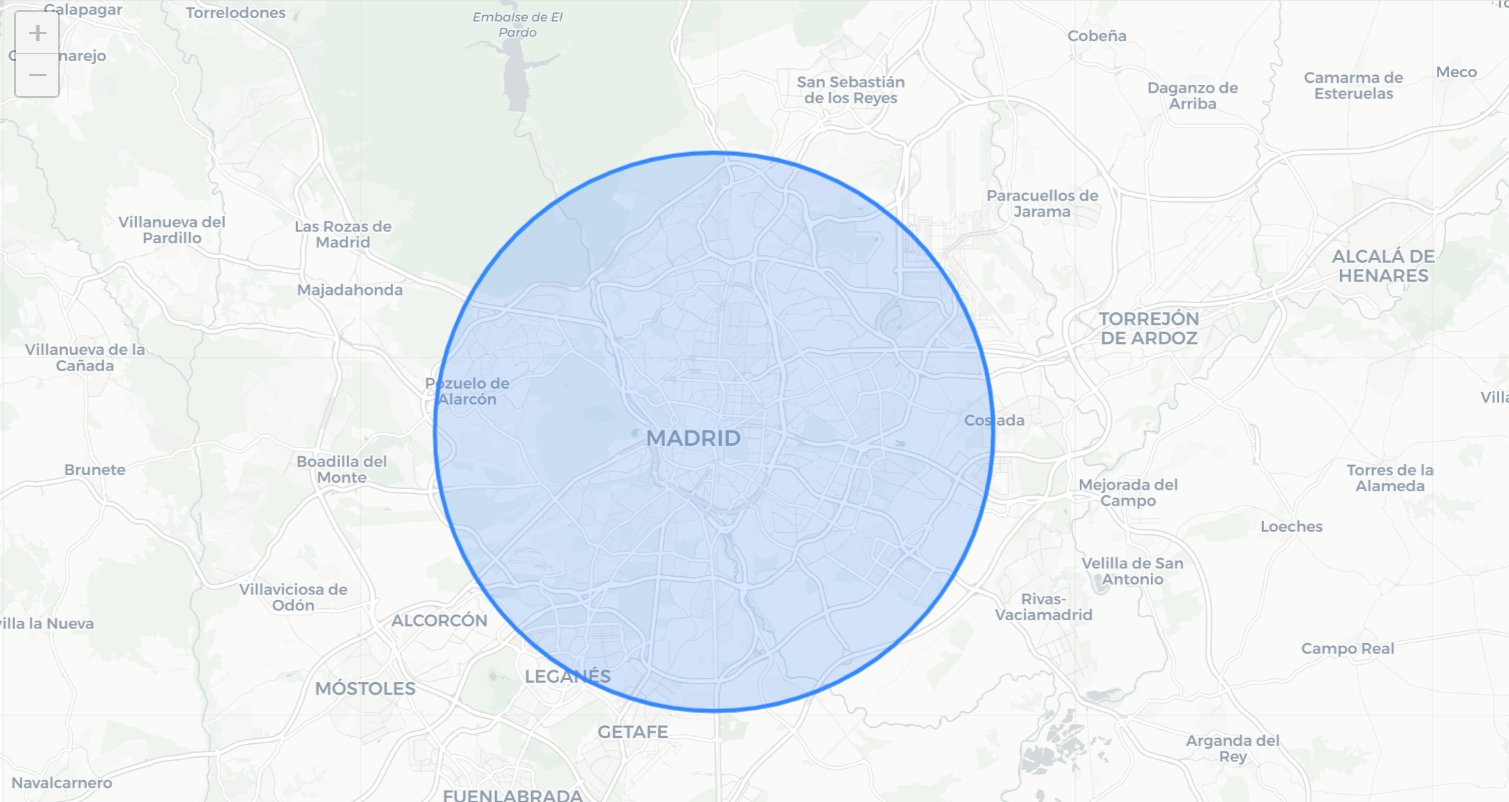


* **Histograma de tabla de vehículos:** Se muestra la autonomia de los diferentes vehiculos.

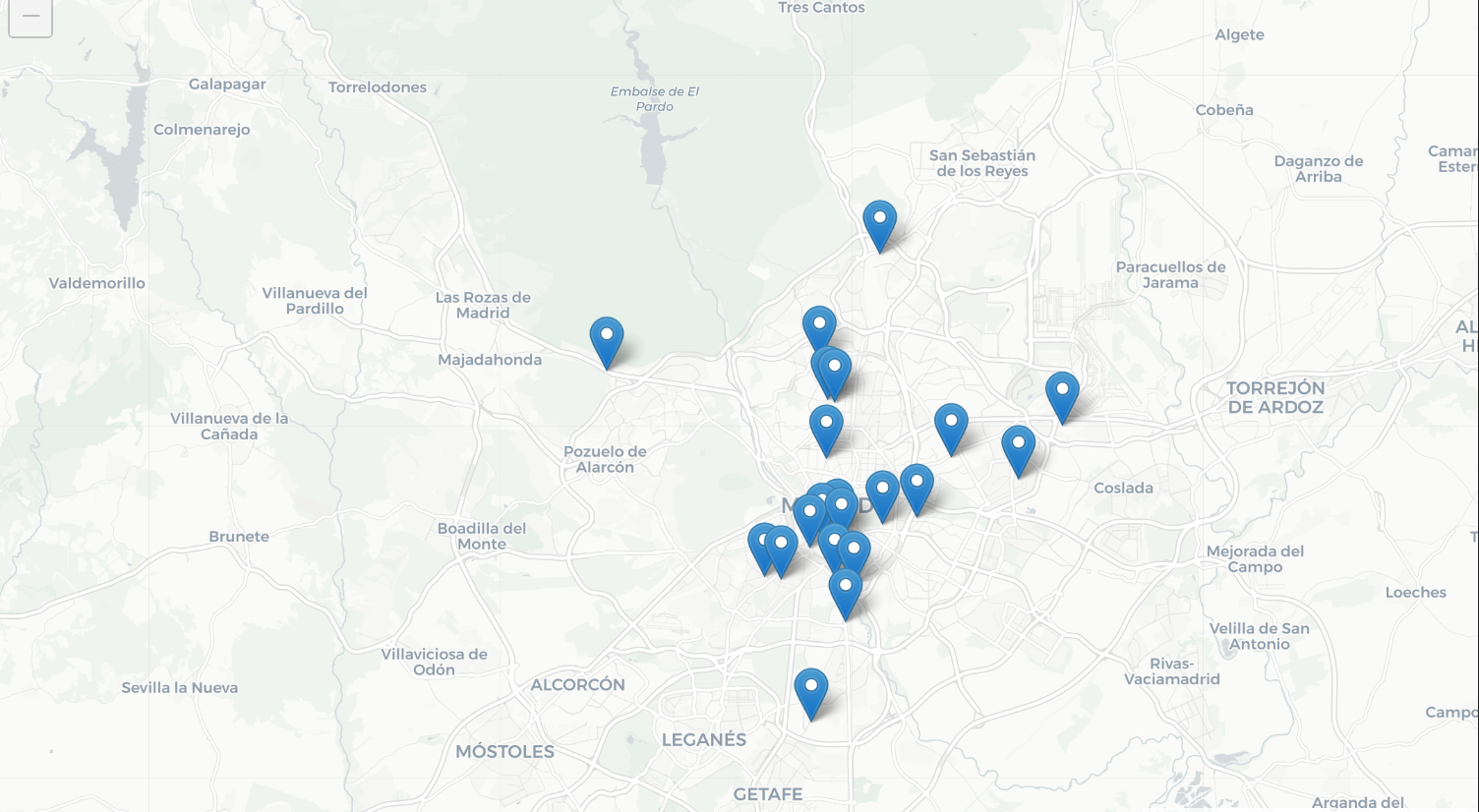


* **Mapa de ubicación de clientes:** Se muestran los distintas ubicaciones de los clientes en un mapa, primero se ve el radio de donde se encuentran las ubicaciones y por último los puntos exactos de las locaclizaciones.

Radio de ubicaciones



Radio de locaclizaciones



1. **Data Preparation**

**4.1 Clean Data and Construct Data**

La preparación de datos consiste en la limpieza y manipulación de los datos en crudo para mejorar el procesamiento y el análisis de dichos datos.

Para el caso número 3 del proyecto (predicción de demanda de clientes de enero de 2025) se ha tenido que realizar un análisis exploratorio del archivo **df\_historic\_order\_demand** para comprobar que los datos estaban correctamente.

Se ha encontrado que la columna order\_demand tenía valores nulos por lo que se ha procedido a su eliminación.

* 1. **Format Data**

Para el caso número 3 del proyecto (predicción de demanda de clientes de enero de 2025) se aplica un algoritmo de regresión (random forest regressor) para poder predecir la demanda y dicho algoritmo no acepta valores categóricos, por lo que las columnas mes\_anio ha tenido que ser transformada a numérica.

Para el caso de la columna mes\_anio, cuando se hizo el análisis exploratorio se vio que dicha columna estaba compuesta por mes y año en este formato “MM-YYYY” y todos los valores de dicha columna eran únicos por lo que aplicar get\_dummies no era la mejor solución.

Se optó por separar mes y año en dos columnas distintas y así poder convertirlas a numéricas.

1. **Modeling**

**A. Clean Data and Construct Data**

La exploración de los modelos se lleva a cabo con el indicado RMSE (Error medio), en el caso de los algoritmos Djikstra, PuLP y genético no se mide el RMSE ya que no es un algoritmo de predicción sino de optimización por lo que el resultado será la ruta más rápida entre una serie de puntos dados.

**B. Build Model**

**1) Algoritmo Dijkstra**

Para el modelo 1, se va a aplicar el algoritmo de optimización Djikstra que es un método utilizado para encontrar el camino más corto en un grafo ponderado (las aristas tienen un peso, que puede representar costo, distancia o cualquier métrica relevante) con pesos no negativos.

Funciona de la siguiente manera:

* El algoritmo de Dijkstra básicamente inicia en el nodo que escojas (el nodo de origen) y analiza el grafo para encontrar el camino más corto entre ese nodo y todos los otros nodos en el grafo.
* El algoritmo mantiene un registro de la distancia conocida más corta desde el nodo de origen hasta cada nodo y actualiza el valor si encuentra un camino más corto.
* Una vez que el algoritmo ha encontrado el camino más corto entre el nodo de origen y otro nodo, ese nodo se marca como "visitado" y se agrega al camino.
* El proceso continúa hasta que todos los nodos en el grafo han sido añadidos al camino. De esta forma, tenemos un camino que conecta al nodo de origen con todos los otros nodos siguiendo el camino más corto posible para llegar a cada uno de ellos.

**2) Algoritmo de Programación Lineal**

Para el modelo 2, se va a aplicar la librería PuLP de Python, que encuentra soluciones óptimas al problema como restricciones lineales.

PuLP permite indicar el tipo de problema que hay que optimizar mediante palabras reservadas de la propia librería, maximización (LpMaximize) o minimización (LPMinimize), que deberán usarse cuando comenzamos a definirlo. Acto seguido se definirá la función objetivo y sus restricciones para que se pueda iniciar la búsqueda de posibles soluciones.

**3) Algoritmo de Genético**

Para el modelo 3 se va a aplicar el algoritmo genético que se empleará para encontrar rutas para el modelo. Este algoritmo se basa en la teoría evolutiva de selección natural propuesta por Darwin en la que los individuos de una población se reproducen generando descendencia, cuyas características, son combinación de las características de los progenitores (más alguna mutación). De todos ellos, únicamente sobreviven los mejores individuos y pueden volver a reproducirse y transmitir sus características a la siguiente generación.

Explicada la teoría, la estructura de dicho algoritmo es la siguiente:

**1.** Crear una población inicial aleatoria de P individuos. En este caso, cada individuo representa una combinación de valores de las variables.

**2.** Calcular la fortaleza (fitness) de cada individuo de la población. El fitness está relacionado con el valor de la función para cada individuo. Si se quiere maximizar, cuanto mayor sea el valor de la función para el individuo, mayor su fitness. En el caso de minimización, ocurre lo contrario.

**3.** Crear una nueva población vacía y repetir los siguientes pasos hasta que se hayan creado P nuevos individuos.

**3.1** Seleccionar dos individuos de la población existente, donde la probabilidad de selección es proporcional al fitness de los individuos.

**3.2** Cruzar los dos individuos seleccionados para generar un nuevo descendiente (crossover).

**3.3** Aplicar un proceso de mutación aleatorio sobre el nuevo individuo.

**3.4** Añadir el nuevo individuo a la nueva población.

**4.** Reemplazar la antigua población por la nueva.

**5.** Si no se cumple un criterio de parada, volver al paso 2.

**C. Access Model**

**1) Algorimo Djikstra**

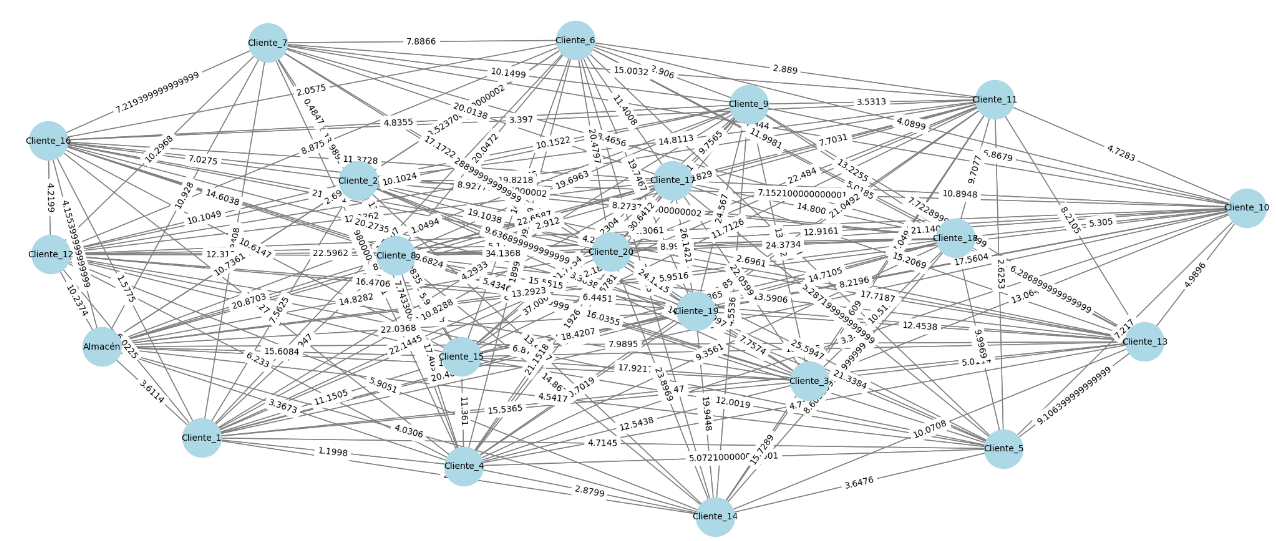
Partimos de los datos de distancias entre los distintos clientes (medida en kilómetros) en forma de matriz.



En la imagen se puede observar que la diagonal principal son todos ceros ya que la distancia entre Cliente\_1 y el mismo es cero.

Por otro lado, se aprecia que hay más ceros aparte de los de la diagonal ya que es posible que no haya una ruta directa entre dos clientes, es decir, que no hay camino.

Para hacernos una idea de cómo están relacionados todos los clientes, creamos un diagrama que muestra la conexión entre todos los nodos y sus correspondientes distancias como se muestra en la siguiente imagen.



* Nuestro primer caso de uso consiste en calcular la ruta más corta (km) entre el almacén y los distintos clientes. Estos han sido los resultados:



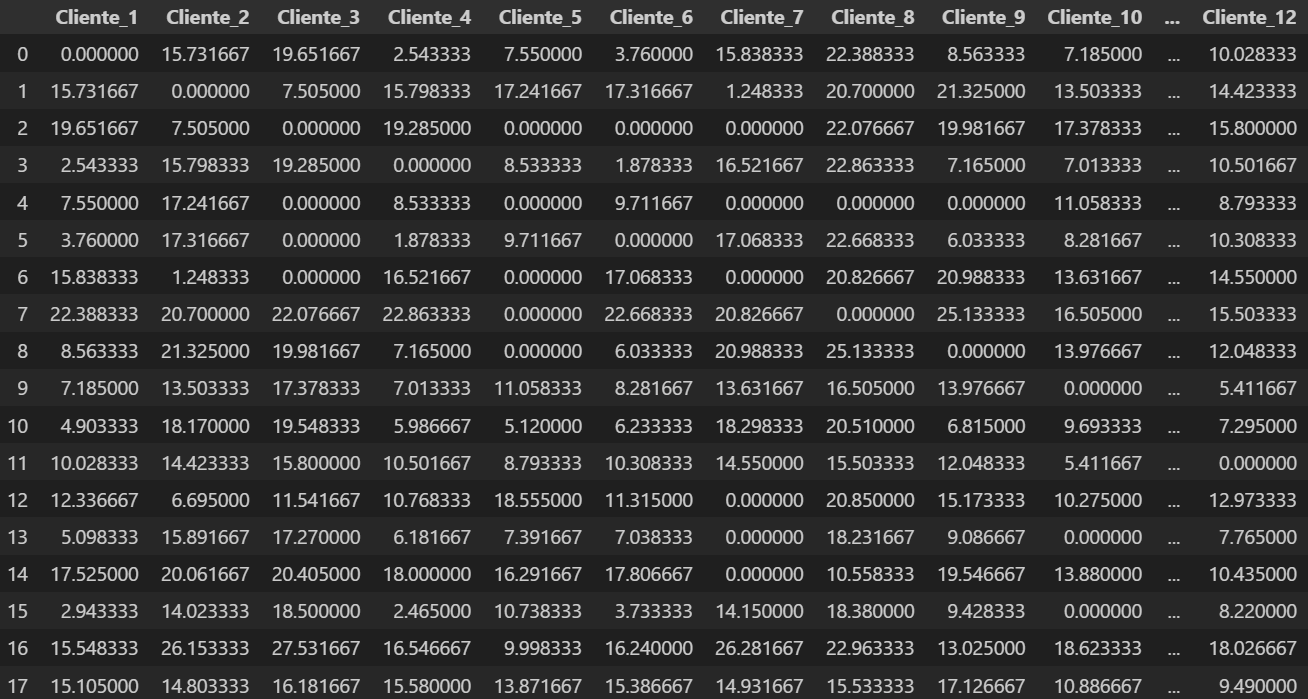
Se observa que hay casos en los que la ruta más corta es directamente ruta que hay entre ellos mismos y en otros casos pasa por otros clientes hasta llegar al nodo final.

Por ejemplo, en el camino más corto para llegar al Cliente\_2 la distancia entre dicho cliente y el almacén es de **10.7361 km,** pero Djikstra ha encontrado una ruta más rápida, pero pasando antes por el Cliente\_6 y el Cliente\_13 y la distancia total recorrida por este camino es de **10.4098 km,** que es ligeramente menor que la primera distancia.

En la siguiente imagen se muestra una lista de distancias entre ambos puntos:



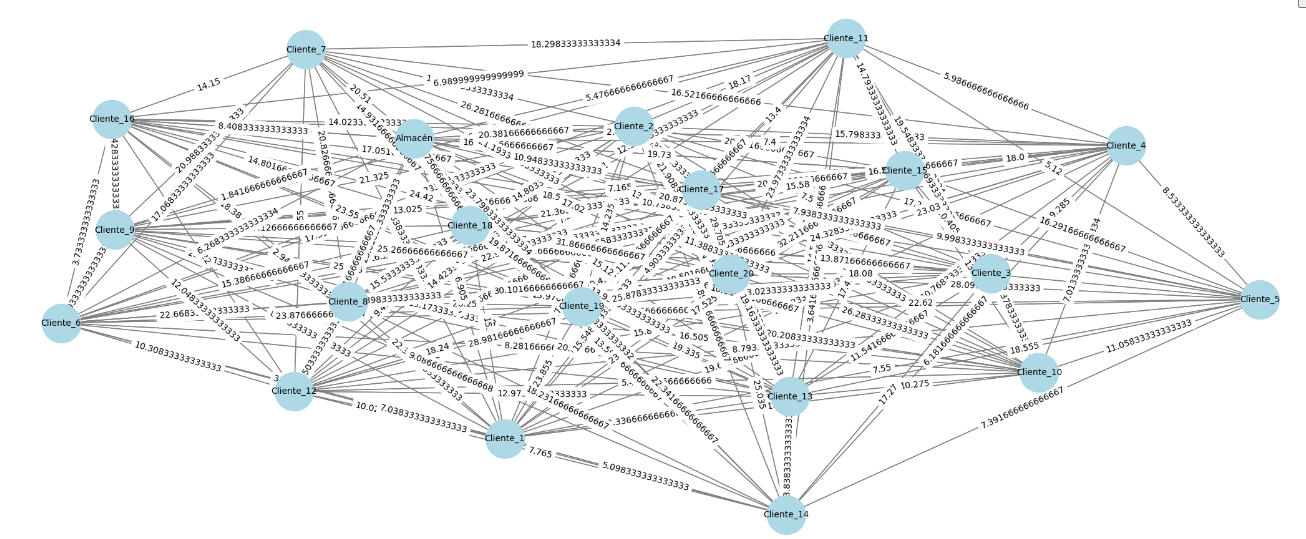
* Nuestro segundo caso de uso consiste en calcular la ruta más rápida en término de tiempo entre el almacén y los distintos clientes. Partimos de estos datos:



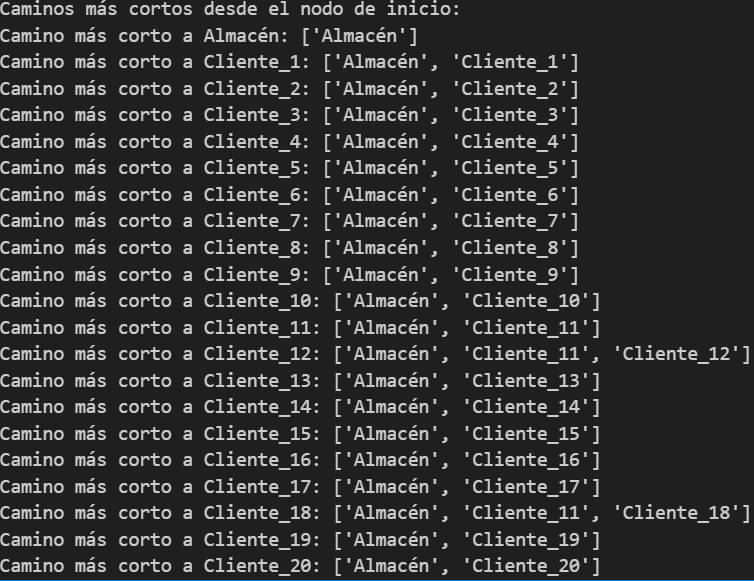
En la imagen se puede observar que la diagonal principal son todos ceros ya que la distancia en tiempo entre Cliente\_1 y el mismo es cero.

Por otro lado, se aprecia que hay más ceros aparte de los de la diagonal ya que es posible que no haya una ruta directa entre dos clientes por lo que el tiempo es 0.

Para hacernos una idea de cómo están relacionados todos los clientes, creamos un diagrama que muestra la conexión entre todos los nodos y sus correspondientes distancias en minutos como se muestra en la siguiente imagen.



Aplicado el algoritmo se han obtenido los siguientes resultados:

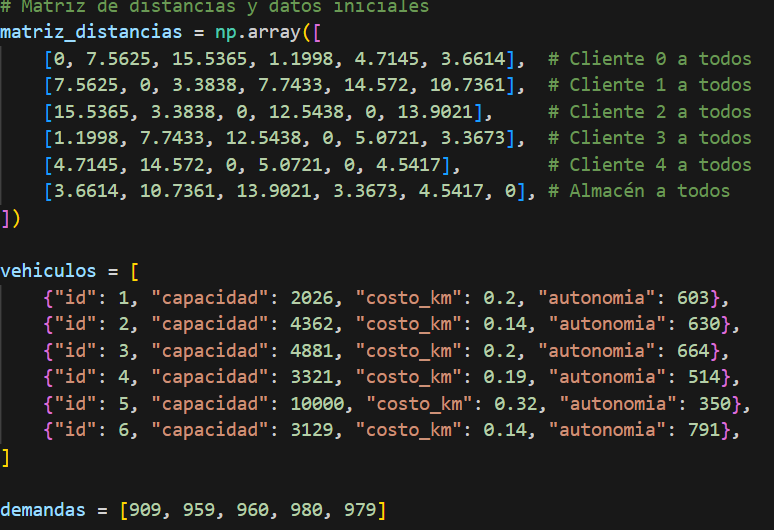


Al contrario que en el primer caso de uso, nos encontramos con que en general la ruta más rápida en término de tiempo es la ruta entre ambos puntos sin intermediarios.

Se observa que los resultados no son los mismo que en el anterior caso de uso ya que una distancia en kilómetros más corta no significa que el tiempo sea menor ya que puede haber obstáculos como semáforos o muchas curvas etc.

**2) Algoritmo de Programación Lineal**

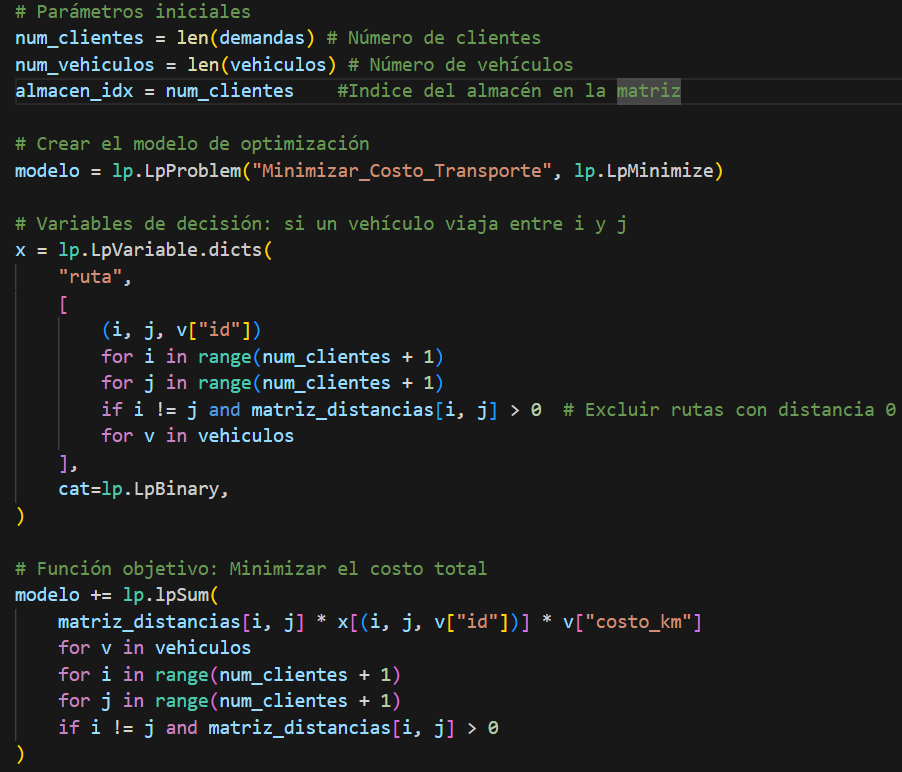
Partimos de los siguientes datos de prueba:



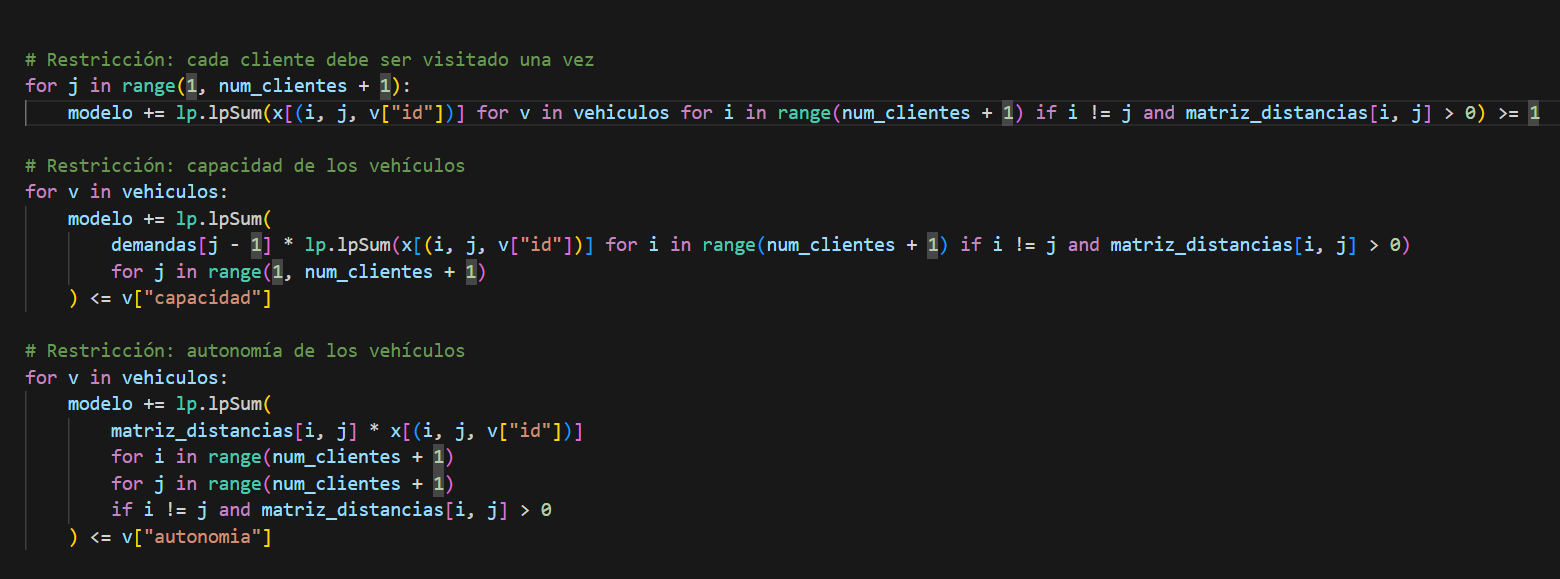
* La matriz de distancias representa la distancia en kilómetros entre clientes y el almacén. Los registros que son ceros significan que no hay ruta directa entre dichos puntos
* Los vehículos contienen información como su capacidad máxima en kilogramos, su autonomía en kilómetros y el costo por kilómetro.
* Las demandas contienen la demanda de pedido de cada cliente en kilogramos.

Nuestro caso de uso consiste en calcular las posibles rutas de cada vehículo que pasen por el máximo número de clientes y con el menor costo posible.

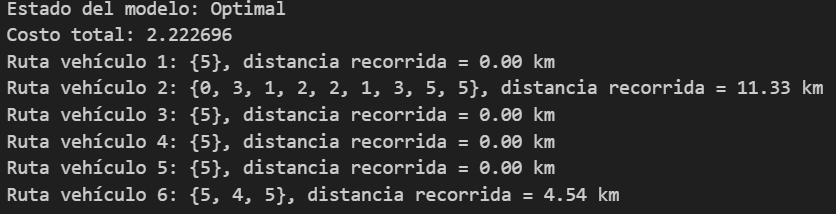
Para ello definimos nuestra función objetivo que en este caso es la de minimizar costes y creamos nuestras variables decisión que serán si un vehículo viaja entre dos clientes, como se muestra en la imagen:



Acto seguido procedemos a crear las restricciones del modelo que serán las relativas a capacidad máxima y autonomía de cada vehículo y que cada cliente debe ser visitado como mínimo una vez.



Ahora que tenemos nuestro objetivo y restricciones definidos pasaremos a resolver nuestro problema y estas han sido las rutas generadas:



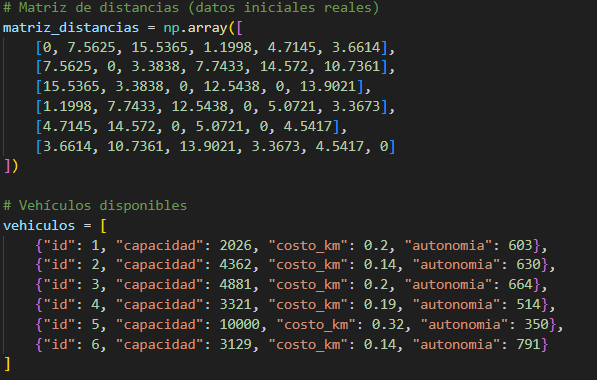
Como se puede observar, el estado del modelo es óptimo por lo que nuestras restricciones y objetivo han funcionado.

Hay cuatro vehículos que no tienen rutas asignadas y el vehículo dos es el que más clientes recorre. También se observa que el vehículo dos no pasa por el cliente 4 (que en realidad es el 5) y el vehículo 6 solo pasa por dicho cliente por lo que nuestra restricción funciona.

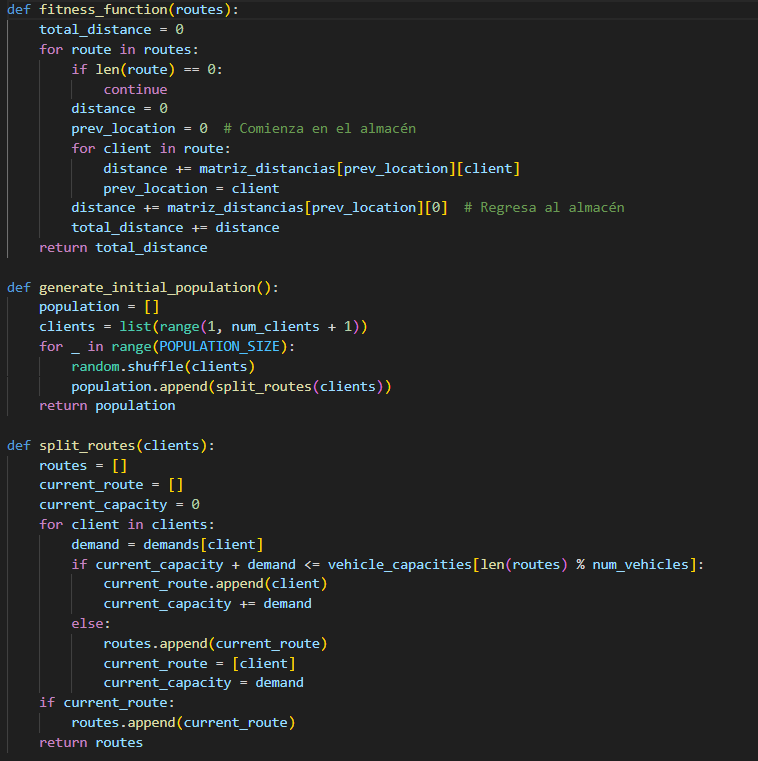
**3) Algoritmo Genético**

Partimos de los mismos datos de prueba del anterior algoritmo, para la realización del algoritmo genético.

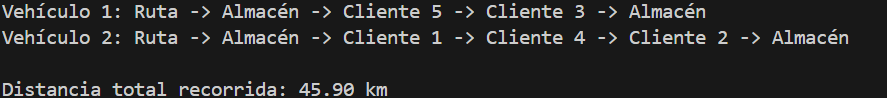
- Nuestro caso de uso es un método de optimización y búsqueda de las mejores rutas, que utiliza procesos como la selección, cruzamiento y mutación para encontrar las rutas óptimas de los vehículos para que repartan a más clientes con el menor coste.



Definimos una función de aptitud que minimiza la distancia total recorrida por los vehículos, asignando clientes a rutas según la capacidad de cada uno. Las soluciones se generan aleatoriamente en clientes y las variables de decisión son las rutas asignadas a los vehículos.



-Aquí tenemos la solución de nuestro algoritmo y las rutas y la distancia generada:



**4) Algoritmo heurístico**

Mediante el uso de la biblioteca ORTools se utilizará como solución de ruteo con restricciones.

Primero realizamos la asignación de los valores a utilizar en el código, teniendo en cuenta una ligera modificación sobre los datos de entrada, de manera que quedará lo siguiente:

Tabla, Excel

Descripción generada automáticamente

Partiendo del archivo modificado y los Excel de df\_orders y df\_vehicle, generamos la definición de las variables a utilizar en la función create\_data\_model(), siendo data la variable diccionario de retorno que contiene los datos a utilizar.

Texto

Descripción generada automáticamente

A continuación se llama a la función main() que ejecutará la totalidad del código, estructurado de la siguiente manera:

1. Almacena la variable retorno de la función previamente comentada
2. Mapea nodos reales a índices internos para OR-Tools y representa las rutas posibles y restricciones
3. La función distance\_callback devuelve la distancia entre nodos usando data['distance\_matrix']. Si no existe una conexión directa, asigna una penalización alta
4. La función demand\_callback garantiza que los vehículos no excedan su capacidad
5. Añadir dimensión de autonomía para verificar que las rutas respeten la autonomía de los vehículos.

Texto

Descripción generada automáticamente

Finalmente la función print\_solution() muestra el resultado de las rutas generadas para que resulten interpretables y claras al usuario

Texto

Descripción generada automáticamente

Para la ejecución del código se hace una llamada a la función main

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza media

Actualmente aún existen errores en la solución generada que deben ser retocados y por tanto las rutas obtenidas no son óptimas ni cumplen todos los requisitos.

Ejemplo de resultado:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

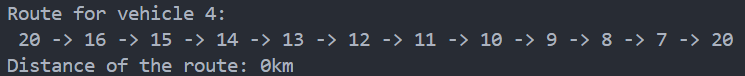
Interfaz de usuario gráfica, Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Texto

Descripción generada automáticamente



Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

1. **Casos de Uso y Escenarios**

Después de haber aplicado a todos los algoritmos realizados una serie de pruebas para comprobar su eficacia y explorar los distintos resultados obtenidos, se ha optado por escoger el algoritmo genético para su integración en los casos de uso reales del proyecto.

A continuación, se explicarán las distintas funciones que componen este algoritmo, ya que en todos los casos de usos las funciones son las mismas, pero en algunas su contenido cambia, pero eso se explicará detalladamente en cada escenario. Las funciones son las siguientes:

* *Generate\_initial\_population*: genera una población aleatoria de individuos y llama a la función Split\_routes.
* *Split\_routes*: crea rutas con la población de la anterior función cumpliendo con las restricciones de capacidad y autonomía de los vehículos.
* *Fitness\_function*: recibe las rutas creadas en la anterior función y comprueba que sean correctas, es decir, que las distancias entre clientes sean distintas de 0 y además añade al principio y al final el almacén y devuelve la distancia total recorrida. Si no son correctas devuelve un número muy alto para que no lo tenga en cuenta en los siguientes pasos.
* *Select\_parents*: se escoge aleatoriamente dos individuos de la población y sus respectivos fitness.
* *Crossover*: se recogen los individuos elegidos anteriormente y aleatoriamente se escoge un numero entre 0 y 1. Si es mayor que 0.9, que en nuestro caso, es la tasa de cruce nos quedamos con el primer individuo y no hay cruce, sino procedemos a coger las listas de rutas donde están involucrados y aleatoriamente cortamos en un punto de dichas rutas y el hijo será la combinación de la parte que hemos cortado del primer padre y de la que no está contenida del primer padre en el segundo padre y se llama a la función Split\_routes para comprobar que se cumplen las restricciones.
* *Mutate*: ya obtenido el resultado de la anterior función, se genera un número aleatorio y si es mayor a MUTATION\_RATE que en nuestro caso es 0.1 no surge mutación ya que buscamos que haya más crossover que mutaciones, en caso contrario se coge la ruta de dicho cliente y se mezclan aleatoriamente todos los clientes de dicha ruta para llamar a la función Split\_routes y crear una ruta nueva.
* *Info\_ruta*: devuelve la información del output final como la ruta completa, la distancia recorrida en cada ruta junto con su vehículo, la capacidad utilizada y el coste de cada ruta.
* *Genetic\_algorithm\_multiple\_runs:* función que engloba todas las funciones anteriores. Como el algoritmo genético funciona con muchos valores aleatorios, lo que hace esta función es repetir ‘x’ veces el algoritmo genético para comparar todas las soluciones y quedarnos con la que menos distancia total (suma de las distancias recorridas en cada ruta) recorra.

Ya aclarado su funcionamiento, se explicarán los distintos casos de uso en los que se verá involucrado y una descripción del mismo:

|  |  |
| --- | --- |
| **Casos de Uso** | **Descripción** |
| Caso 1: Optimización de rutas para los pedidos del último mes disponible | Utiliza los datos actuales de clientes, vehículos y pedidos para planificar las rutas de entrega, minimizando el coste total y maximizando el número de entregas realizadas dentro de las restricciones de capacidad, autonomía y vehículos disponibles. |
| Caso 2: Optimización de rutas para los pedidos del último mes disponible reduciendo la flota | Por decisión de la empresa, solo se pueden usar la mitad de los vehículos disponibles. Simula las rutas con una flota más pequeña (disminuir la flota en un 50% de vehículos) y evalúa el impacto en el número de entregas realizadas, el coste y la eficiencia general. ¿Qué cambios observas? |
| Caso 3: Predicción de demanda de pedidos para el próximo mes | Utilizando el histórico de pedidos, predice el número de pedidos esperados para el próximo mes. Con esta información, simula las rutas necesarias para cumplir con la demanda prevista. Para resolver este caso de uso es necesario usar la tabla de histórico de pedidos. |
| Caso 4: Optimización de rutas reduciendo la duración del trayecto | Para poner a prueba nuestras capacidades logísticas y demostrar nuestra eficiencia, se ha garantizado a nuestros clientes que su pedido será entregado en un plazo inferior a una hora desde que el vehículo sale del almacén. |

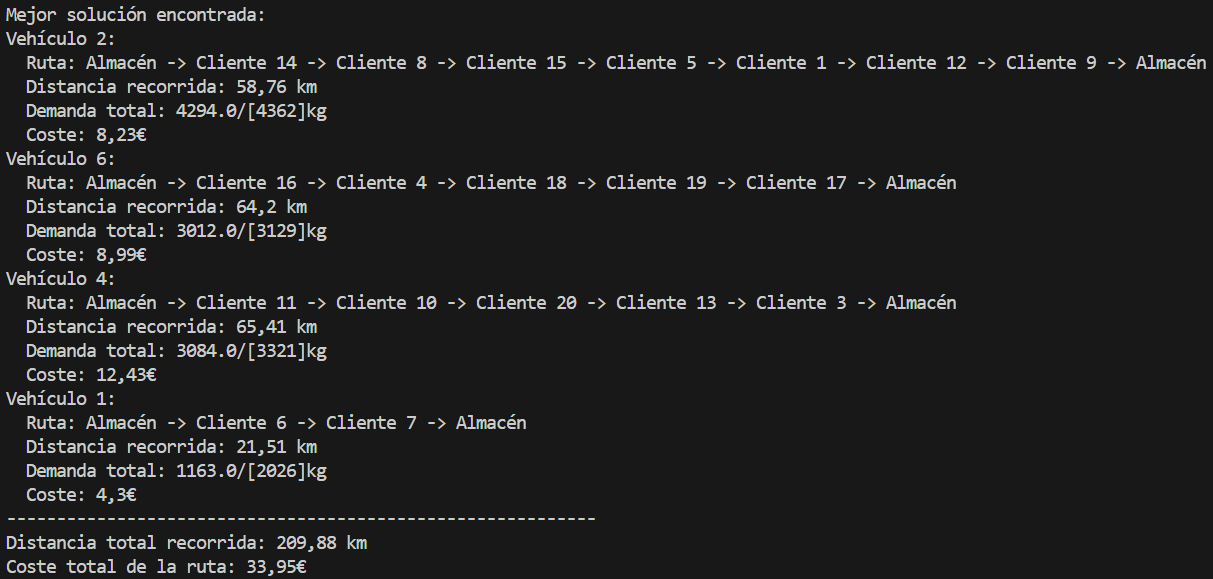
1. **Caso 1**

Para la aplicación de este escenario primero se procede a leer los datos que van a ser necesarios para dicho caso, que son: datos históricos de pedidos, datos de los vehículos disponibles y datos de las distancias entre los clientes y el almacén que se obtienen de los csv df\_distance\_km, df\_historic\_order\_demand y df\_vehicle.

Al aplicar el algoritmo en situaciones diferentes, nos damos cuenta de que para generar rutas escoge los vehículos en orden por lo que, por norma general, los últimos vehículos no suelen tener muchos clientes y las rutas son más baratas. Siguiendo esta idea, ordenamos los vehículos de menor a mayor consumo para que los vehículos de menor consumo tengan más clientes y las rutas, aunque más extensas, saldrán más baratas.

Al no especificar qué fecha de demanda escoger, se algoritmo está preparado para utilizar cualquier fecha que esté en dicho csv, lo que conlleva reforzar el algoritmo para que si en algún mes hay algún cliente que no haga pedido de producto, en la solución final no aparezca dicho cliente.

Mencionada la anterior información se muestra el resultado de aplicar el algoritmo a los pedidos realizados en el mes de febrero de 2021 en el que el Cliente\_2 no realiza ningún pedido:



Se puede observar que, como antes se ha mencionado, los primeros vehículos se llevan la mayor parte del trabajo viendo como el vehículo 1 solo reparte a dos clientes además de que en el csv de df\_vehicle hay seis vehículos y en esta solución solo hay cinco, lo que indica que no se ha necesitado dicho vehículo.

Por otro lado, el Cliente\_2 no aparece en ninguna ruta porque al no haber realizado ningún pedido no se incluye.

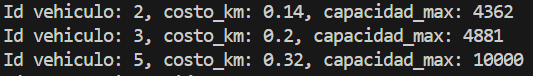
En relación con la capacidad máxima de cada vehículo, se ve nunca se ve superada ya que, por ejemplo, la suma de todos los pedidos realizados por los clientes de la primera ruta es de 4294 kg y la capacidad máxima del vehículo es de 4362 kg por lo que la restricción funciona además de la de autonomía.

Por último, se ve que el precio de cada ruta va de menos a más ya que antes se dijo que los vehículos se ordenaron de menos a más costo por km.

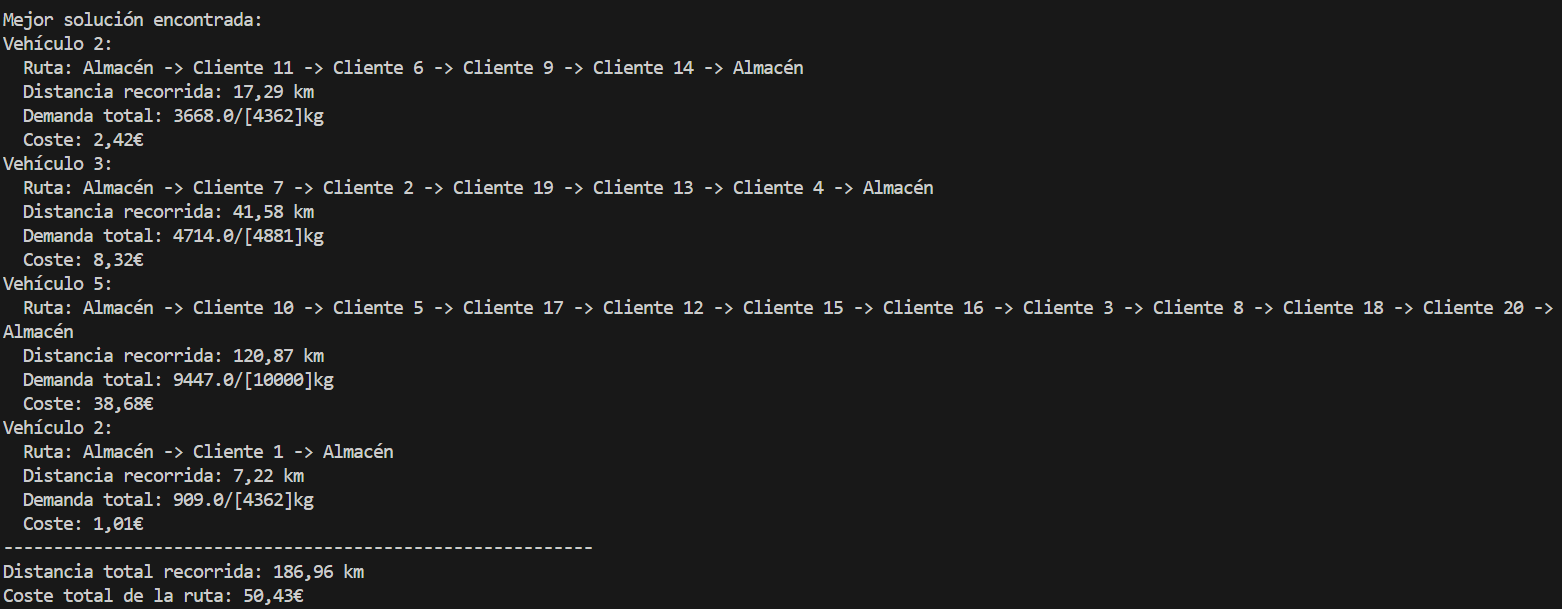
1. **Caso 2**

Para la aplicación de este escenario primero se procede a leer los datos que van a ser necesarios para dicho caso, que son: datos históricos de pedidos, datos de los vehículos disponibles y datos de las distancias entre los clientes y el almacén que se obtienen de los csv df\_distance\_km, df\_historic\_order\_demand y df\_vehicle.

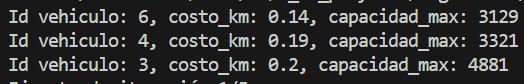
En ambos escenarios, nos vemos obligados a reducir la flota de vehículos a la mitad, que en nuestro caso se ha hecho aleatoriamente para ver el impacto en cada resultado. Al ser aleatorio, el resultado final dependerá del vehículo que se haya escogido porque cada vehículo tiene su costo por km y su capacidad máxima además de su autonomía. Para los siguientes escenarios, se escogerá los pedidos que se realizaron en diciembre de 2024 en el que todos los clientes hacen pedidos.

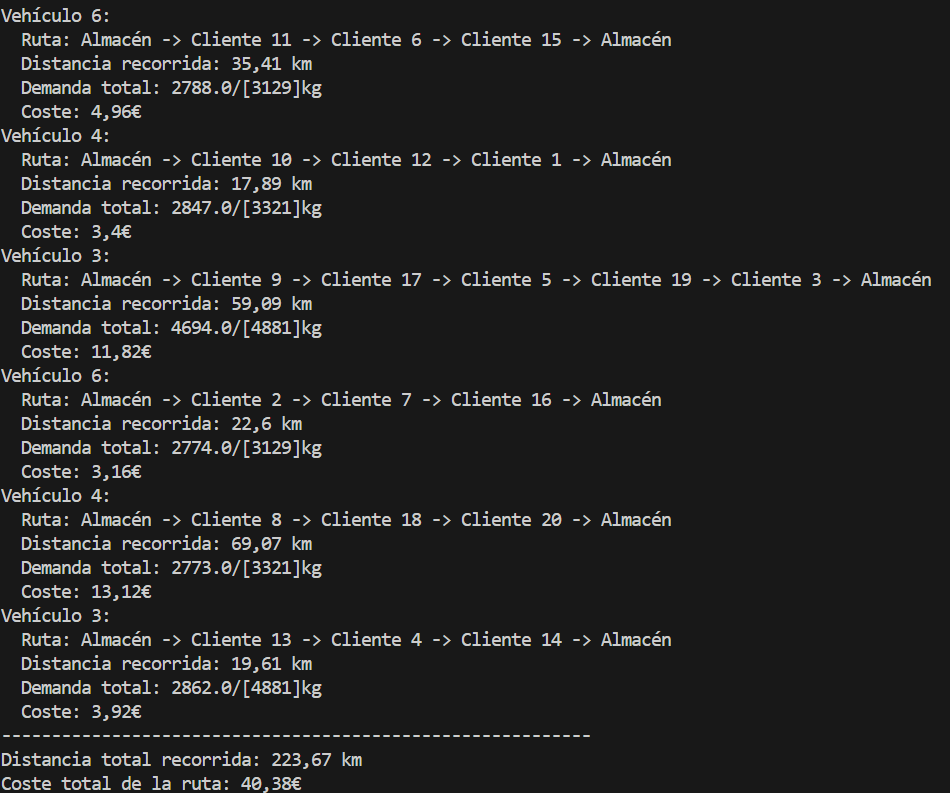
En nuestra primera ejecución vemos como se han escogido los vehículos con las siguientes características:  


En este escenario, al haber solo tres vehículos disponibles el número de rutas va a aumentar ya que es casi imposible que con las restricciones de capacidad todos los pedidos se puedan entregar en solo tres rutas y al haber un vehículo con una carga de 10000 kg hará que este sea el que más pedidos entregue, además de que es el más caro por kilómetro y eso afectará al precio final como se muestra en la siguiente imagen:



Se observa como las rutas de los vehículos 2 y 3 son más baratas debido a su precio por kilómetros, pero su capacidad se ve limitada. Por el mismo motivo, el vehículo 5 al tener mucha capacidad se lleva toda la demanda de clientes, pero su precio por kilómetro es mucho mayor lo que lleva a que el precio final de las rutas sea de 50 € aunque la distancia recorrida no sea muy elevada. Cabe mencionar que los veinte clientes que hicieron pedidos en la fecha anteriormente mencionada reciben su pedido.

En este otro escenario se han elegido estos vehículos:  


Como se ve dichos vehículos no tiene un costo por kilómetro muy elevado pero su capacidad es menor por lo que el precio final será barato pero el número de rutas crecerá:  


Comparado con el anterior escenario, se observa que se han generado más rutas debido a la capacidad máxima de cada vehículo, pero su precio es mucho más barato por cada ruta.

En conclusión, el resultado final se verá afectado en gran medida en la elección aleatoria de los vehículos, pero por norma general los vehículos se verán más cargados de pedidos de lo normal para intentar suplir la demanda lo que llevará a más rutas y más kilómetros recorridos.

1. **Caso 3**

Para la aplicación de este escenario primero se procede a leer los datos que van a ser necesarios para dicho caso, que son: datos de pedidos de enero de 2025, datos de los vehículos disponibles y datos de las distancias entre los clientes y el almacén que se obtienen de los csv df\_distance\_km, pedidos\_2025 y df\_vehicle.

En este escenario, ya no tenemos los datos de la demanda de cada cliente, ya que el objetivo es realizar las rutas para los pedidos realizados en enero de 2025 pero el histórico de pedidos termina en diciembre de 2024 por lo que tenemos que predecirlos a partir del histórico de demandas.

Se procede a realizar un pequeño exploratorio de los datos que ya se mencionó en los puntos anteriores y se encuentra que el csv está compuesto por tres columnas que son:

* **Cliente:** aquí se encuentran los clientes.
* **Mes\_anio:** la fecha compuesta por mes y el año de tipo string.
* **Order\_demand:** la demanda de producto realizada por cada cliente.

La columna order\_demand contenía nulos por lo que se procedió a su eliminación y la columna mes\_anio al ser de tipo string se tuvo que separar en dos columnas una de mes y otra de año y casteada a numérica.

Realizados los ajustes necesarios de los datos, se procedió a dividir los datos en entrenamiento (80 %) y test (20%) para realizar la predicción correctamente.

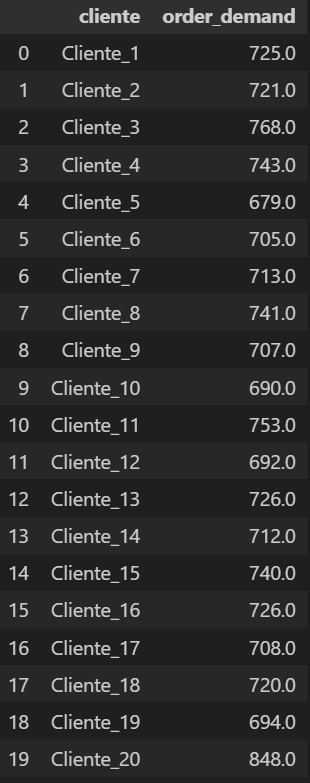
Para aplicar a la predicción se tienen en cuenta dos algoritmos:

* *Random Forest Regressor*: en este algoritmo se combinan varios árboles de regresión, cuya predicción final será la media de los resultados de cada árbol. Se ha tenido en cuenta por su posibilidad de personalizar muchos parámetros para realizar una mejor predicción y por estar mejor preparado para el overfitting que, por ejemplo, un árbol de regresión.
* *XGBRFRegressor:* este algoritmo también utiliza árboles de regresión como el anterior algoritmo, pero trata de mejorar el resultado teniendo en cuenta el resultado obtenido por el árbol anterior.

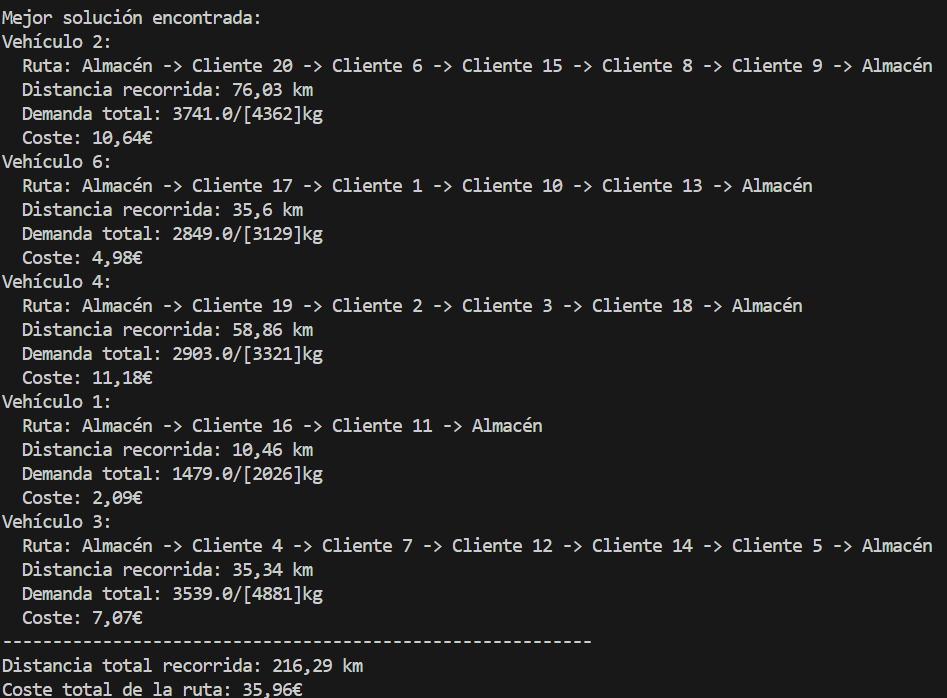
Finalmente se optó por la utilización del Random Forest Regressor al obtener mejores resultados.

Tras dividir los datos procedemos a entrenar el modelo. Para encontrar los mejores parámetros, se opta por crear una lista de valores por cada parámetro que influya en la predicción que son la profundidad de los árboles, el criterio utilizado y el min\_samples\_split que cuyo valor será el valor mínimo de muestras necesario para hacer una división en un nodo.

Hechas las listas se procede a crear un modelo por cada cliente para realizar una correcta predicción de los valores y el resultado obtenido para enero de 2025 es el siguiente:



Ya obtenidas las predicciones, se procede a implementarlas al algoritmo genético y el resultado obtenido es el siguiente:



Como se observa, es un escenario parecido al del caso 1 en el sentido de que se utilizan cinco vehículos de los seis disponibles y las rutas más largas la obtienen vehículos con costo por kilómetro barato obteniendo finalmente un precio muy parecido.

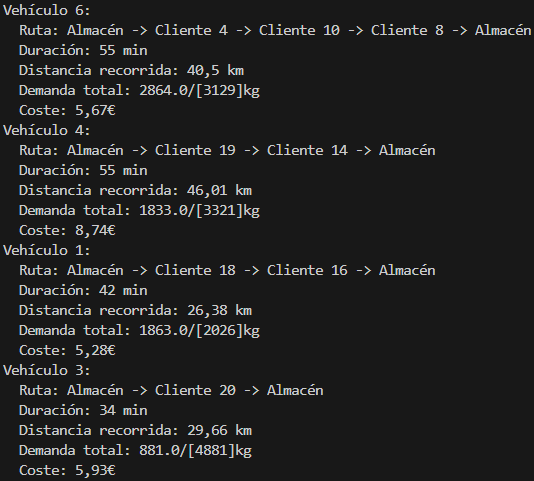
1. **Caso 4**

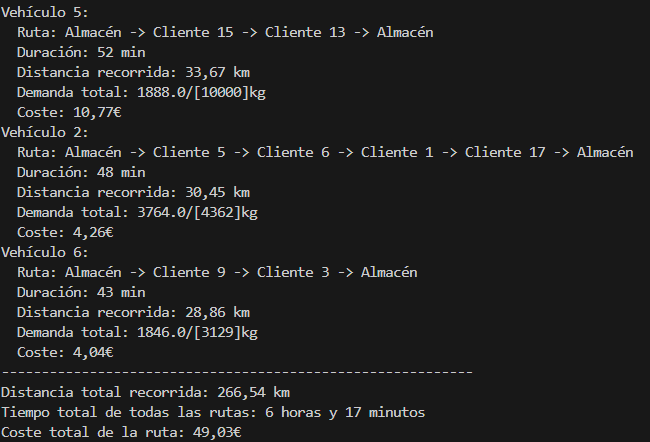
Para la aplicación de este escenario primero se procede a leer los datos que van a ser necesarios para dicho caso, que son: datos históricos de pedidos, datos de los vehículos disponibles, datos de las distancias entre los clientes en kilómetros y datos de las distancias entre clientes y el almacén en minutos que se obtienen de los csv df\_distance\_km, df\_distance\_min, df\_historic\_order\_demand y df\_vehicle.

En este último caso de uso, para poner a prueba nuestras capacidades logísticas y demostrar nuestra valía en situaciones límite a nuestros clientes y a la competencia, se ha garantizado a nuestros clientes que su pedido será entregado en un plazo inferior a una hora desde que el vehículo sale del almacén.

Para ello, se han cambiado el contenido de algunas funciones para este escenario especial como Split\_routes que ahora trabaja con las distancias en minutos en vez de kilómetros y poner la restricción de que cada ruta debe durar 60 o menos de 60 minutos. La función fitness\_function ahora trabaja con distancias en minutos para posteriormente de la función genetic\_algorithm\_multiple\_runs se comparé entre distancias en minutos.

Explicado los pequeños retoques de nuestro algoritmo el resultado obtenido se muestra en las siguientes imágenes:





Se observa que al poner la restricción de menos de 60 min hay más rutas de lo normal, en este caso siete por lo que el vehículo dos al volver del primer trayecto tiene que repostar y volver a salir.

Aunque haya habido más rutas y se haya recorrido más distancia en kilómetros, el precio no se ha visto demasiado afectado y esto se ha debido a que los trayectos más largos se los han llevado los vehículos con costo por kilómetro más barato.

Para finalizar, vemos la eficacia y el cumplimiento de las restricciones de minutos, capacidad y autonomía que no han sido rebasadas en ningún momento.

En conclusión, se ha visto que el número de rutas ha aumentado en relación con el aumento de restricciones pero el resultado ha sido mejor de lo esperado.