PROYECTO 1

Plataforma de Distribución de Mercancías Perecederas

Documentación proyecto

JiaJiao Xu, Jordi Vidal

Índice

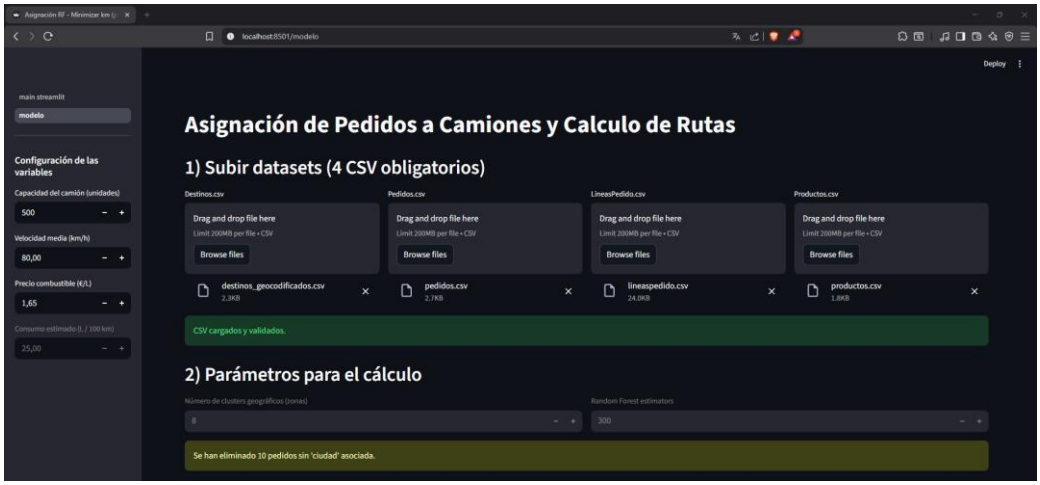
Introducción 2
Objetivos y requisitos del proyecto 3
Objetivos 3
Requisitos funcionales 3
Requisitos no funcionales 4
Análisis previo 5
Criterio adoptado 6
Origen de los datos 7
Tratamiento de datos 9
Modificaciones en los datos: 10
Análisis de tecnologías 11
Librerías de python 11
API 12
Antes del cálculo 13
Flujo de cálculo 14
Definición modelos usados 14
Visualización de resultados 16
Datos complementarios 17
Resultados de pruebas unitarias, de integración y de aceptación 18
Análisis de rendimiento de resultados 20
Configuración de los modelos 21
Conclusiones y mejoras futuras 22
Conclusiones 22
Mejoras futuras 23
Video corporativo 23

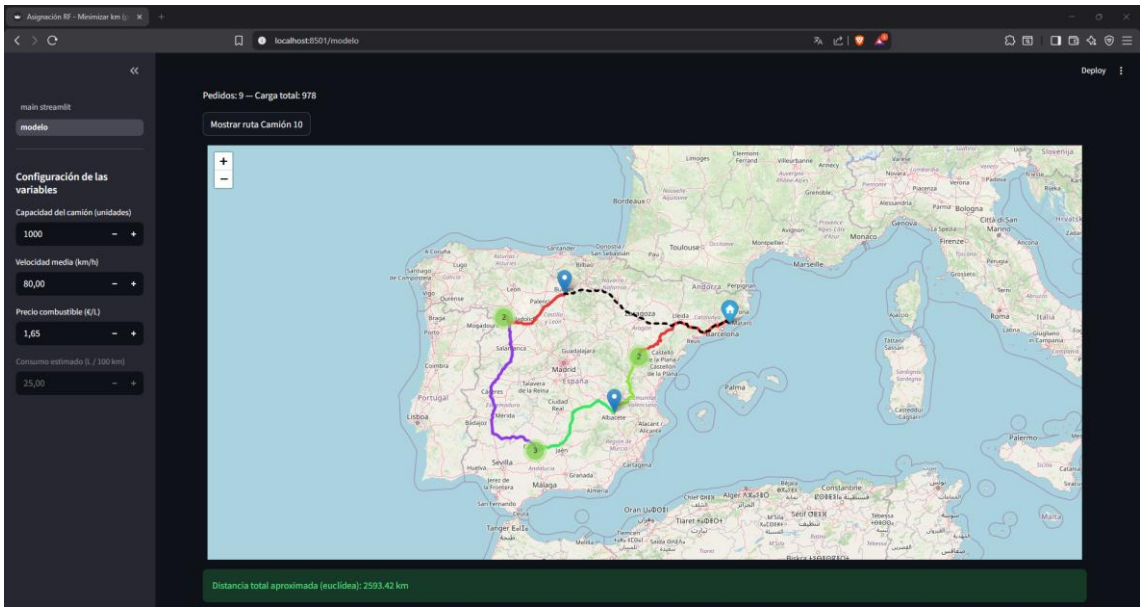
Introducción

El objetivo de este proyecto era crear el diseño y desarrollo de un software orientado a la planificación y optimización de rutas de distribución logística para una empresa dedicada a la fabricación y reparto de productos alimentarios perecederos de corta vida útil.

El problema principal se centra en la necesidad de reducir costes logísticos, garantizar el cumplimiento de restricciones temporales asociadas a la caducidad del producto y mejorar la eficiencia operativa en la asignación de pedidos a una flota de vehículos.

La solución desarrollada permite automatizar el proceso de análisis de pedidos, reparto de estos de forma óptima en los vehículos disponibles, agrupación geográfica de destinos y generación de rutas óptimas desde un punto de origen central, proporcionando al usuario una herramienta interactiva que facilita la toma de decisiones logísticas basadas en datos.





Objetivos y requisitos del proyecto

Objetivos

El objetivo principal del proyecto es desarrollar una herramienta software que permita optimizar la planificación de rutas de distribución logística para productos perecederos, reduciendo los costes asociados al transporte y garantizando el cumplimiento de las restricciones temporales derivadas de la fabricación y la caducidad de los productos.

De manera específica, el proyecto tiene los siguientes objetivos:

- Diseñar una solución capaz de automatizar el proceso de asignación de pedidos a vehículos de reparto.
- Minimizar la distancia total recorrida por la flota de transporte.
- Reducir el número de vehículos necesarios para cubrir la demanda.
- Priorizar la entrega de pedidos con mayores restricciones de caducidad.
- Facilitar la toma de decisiones logísticas mediante visualizaciones claras e intuitivas.
- Proporcionar métricas de apoyo que permitan estimar el impacto económico del transporte.
- Desarrollar una solución flexible que pueda adaptarse a diferentes escenarios de carga, volumen de pedidos y áreas geográficas.

Requisitos funcionales

El sistema deberá cumplir con los siguientes requisitos funcionales:

- Permitir la carga de ficheros de datos en formato CSV correspondientes a destinos, pedidos, líneas de pedido y productos.
- Validar la estructura y consistencia de los datos de entrada, detectando errores o campos obligatorios ausentes.
- Calcular el volumen total de carga asociado a cada pedido.
- Determinar la prioridad temporal de cada pedido en función de la fecha de pedido, tiempo de fabricación y caducidad.
- Agrupar pedidos según criterios geográficos para facilitar su posterior asignación.

- Asignar pedidos a una flota de camiones homogéneos respetando las restricciones de capacidad.
- Generar rutas de reparto ordenadas para cada camión.
- Optimizar las rutas generadas con el objetivo de reducir el kilometraje total.
- Calcular estimaciones de distancia recorrida, tiempo de conducción y coste de combustible.
- Mostrar los resultados mediante mapas interactivos y tablas resumen.
- Permitir la exportación de los resultados de la planificación.

Requisitos no funcionales

El sistema deberá satisfacer los siguientes requisitos no funcionales:

- Usabilidad: la aplicación debe ser intuitiva y fácil de utilizar por usuarios sin conocimientos técnicos avanzados.
- Rendimiento: el sistema debe ofrecer tiempos de respuesta adecuados incluso con volúmenes elevados de pedidos.
- Escalabilidad: la solución debe permitir la ampliación futura a nuevos volúmenes de datos, zonas geográficas o criterios de optimización.
- Mantenibilidad: el código y la arquitectura deben facilitar su comprensión, modificación y ampliación.
- Portabilidad: la aplicación debe poder ejecutarse en diferentes entornos sin requerir configuraciones complejas.
- Robustez: el sistema debe gestionar de forma controlada errores en los datos de entrada o situaciones excepcionales.
- Reproducibilidad: dadas las mismas entradas y parámetros, el sistema debe generar resultados coherentes y comparables.

Análisis previo

Uno de los problemas detectados durante el análisis de los datos de ejemplo es la gestión de pedidos que incluyen productos con distintos tiempos de fabricación y diferentes periodos de caducidad. Esta situación complica la planificación logística, ya que la viabilidad del pedido completo queda condicionada por el producto que presenta la restricción temporal más severa.

En un mismo pedido pueden encontrarse productos que se fabrican rápidamente pero caducan en pocos días, junto con otros que requieren más tiempo de fabricación pero tienen una caducidad más amplia. Por ejemplo, un producto con fabricación corta y caducidad reducida puede convivir con otro que tarda más en fabricarse pero se conserva durante más tiempo.

En este tipo de situaciones, aunque el segundo producto no sea especialmente crítico desde el punto de vista de la caducidad, el pedido completo debe entregarse dentro del margen de tiempo marcado por el producto que caduca antes. Esto obliga a que el pedido salga a reparto en una ventana muy concreta, incluso aunque otros productos del mismo pedido todavía no hayan terminado su proceso de fabricación.

El conflicto aparece cuando el producto más restrictivo se fabrica antes que el resto. En ese caso, cumplir su caducidad implicaría entregar el pedido antes de que todos los productos estén listos, lo que hace que el pedido sea potencialmente no factible bajo el supuesto de entrega conjunta, aquí un ejemplo de lo descrito:

data > lineaspedido.csv

	LineaPedidoID	PedidoID	ProductoID	Cantidad
51	51	3	2	5
52	52	3	47	10
53	53	3	4	7
54	54	3	26	4

En este caso tenemos en un mismo pedido, pedidoID 3, el producto 2 y el producto 4, si vamos a ver tiempos de fabricación y caducidad de estos productos, tenemos que:

data > productos.csv

	ProductoID	Nombre	PrecioVenta	TiempoFabricacionMedia	Caducidad
1	1	Producto1	45.84	3	9
2	2	Producto2	54.20	9	27
3	3	Producto3	74.19	7	21
4	4	Producto4	53.82	1	3
5	5	Producto5	64.84	1	3

Si empezamos a fabricar el pedido, tardaríamos 1 día en fabricar el producto 4, una vez fabricado caducaría en 3 días más, dando un total de 4 días para fabricar y que caduque. En cambio, el producto 2 del mismo pedido se empieza a fabricar a la vez pero tarda 9 en acabar de fabricarse, para cuando esto sucede, el producto 4 se ha fabricado y ha caducado.

Criterio adoptado

Para abordar el problema de los pedidos que incluyen productos con distintos tiempos de fabricación y caducidad, se ha optado por aplicar un criterio sencillo y conservador que permita mantener la coherencia del modelo y facilitar la planificación.

En este contexto, se asume que **todos los productos de un pedido finalizan su fabricación al mismo tiempo** y se entregan de forma conjunta, independientemente de sus tiempos de fabricación individuales. Bajo este supuesto, la fecha límite de entrega del pedido viene marcada por el producto más restrictivo, es decir, aquel cuya combinación de tiempo de fabricación y caducidad deja menos margen de entrega. Basta con que uno de los productos no cumpla su caducidad para que el pedido completo deje de ser válido.

Para cada línea del pedido se calcula un valor denominado **TiempoProducto**, que corresponde a la **suma del tiempo medio de fabricación y la caducidad** del producto. A nivel de pedido, se toma el **mínimo de estos valores** entre todas las líneas, generando así el **TiempoTotalEstimado**. Este valor representa la restricción más exigente del conjunto y es la métrica que se utiliza para determinar la prioridad de los pedidos.

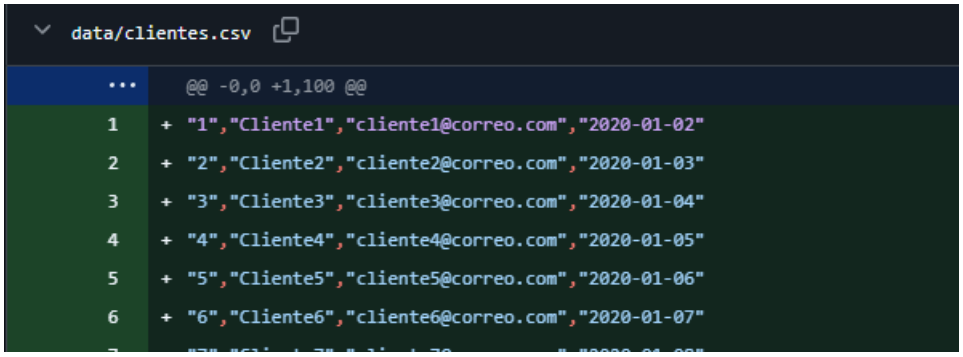
Durante la asignación a camiones, los pedidos con **menor TiempoTotalEstimado** se procesan primero y se intenta agruparlos con otros pedidos de urgencia similar. Esto evita que en una misma ruta se mezclen pedidos con niveles de prioridad muy distintos, asegurando que los productos más críticos cumplan su caducidad y que la planificación sea coherente con los tiempos de fabricación y entrega.

	PedidoID	ciudad	latitud	longitud	Cantidad_total	TiempoTotalEstimado
0	1	Guadalajara	40.74	-2.5059	86	4
1	2	Sevilla	37.3886	-5.9953	107	4
2	4	Castellón	40.2519	-0.0615	104	4
3	5	Navarra	42.6125	-1.8308	99	4
4	6	Lugo	43.0396	-7.4566	100	4
5	8	Palencia	42.4096	-4.6161	110	4
6	9	La Rioja	42.3286	-2.4675	127	4
7	10	Vizcaya	43.2385	-2.8516	95	8
8	11	Burgos	42.3439	-3.697	106	4
9	12	Guipúzcoa	43.1445	-2.2038	132	4

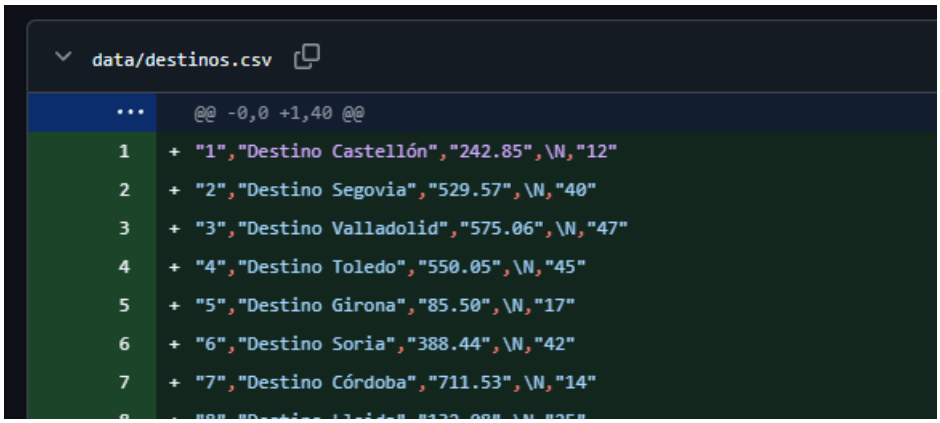
Origen de los datos

Los datasets proporcionados de ejemplo para construir la aplicación son los siguientes:

Clientes.csv, donde tenemos información como el nombre el mail y la fecha de registro de cada cliente:

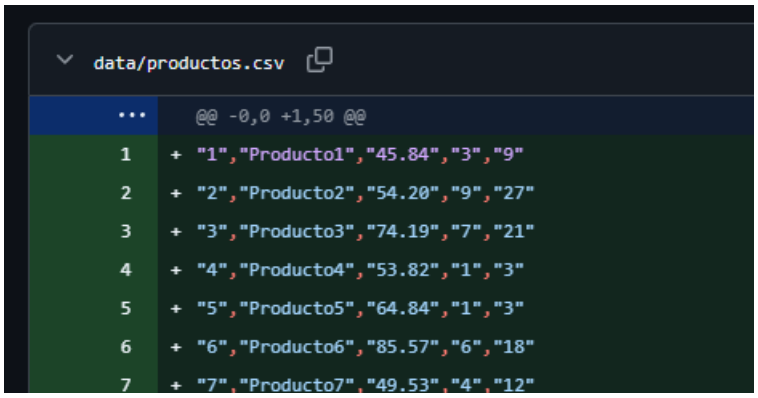


Destinos.csv, en este dataset podemos encontrar la información de cada posible destino, ciudad concreta del destino, distancia desde el punto origen y en que provincia esta esta ciudad:

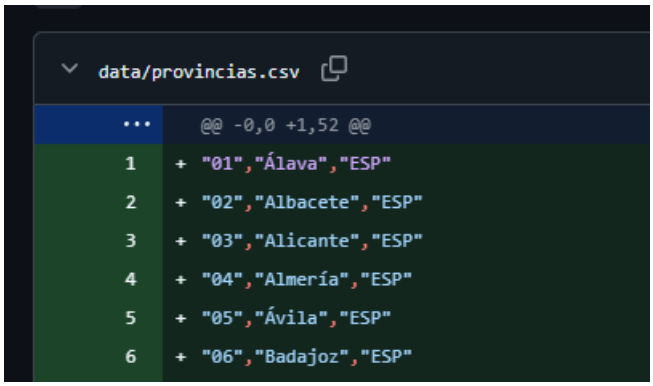


Pedidos.csv, aquí podemos encontrar información del pedido como cuando se hizo, por que cliente y a que destino hay que entregarlo:

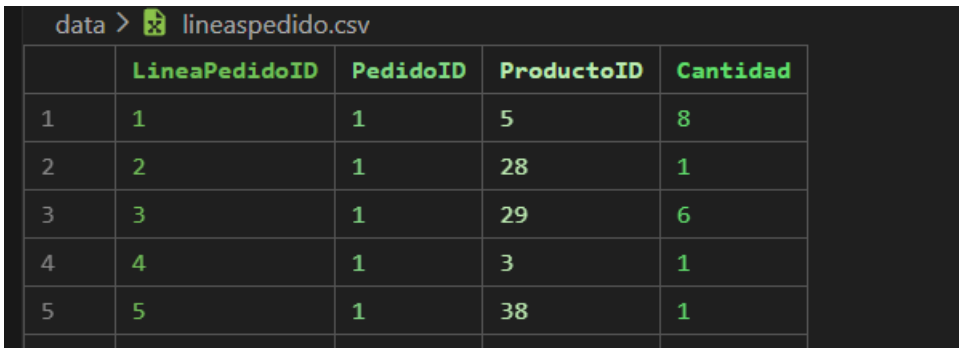
Productos.csv, en este dataset podemos encontrar toda la información relacionada con los productos, nombre, precio de venta, tiempo de fabricación media y caducidad:



Provincias.csv, también tenemos un dataset con el nombre de las provincias, este dataset tiene más sentido en un futuro si se tiene pensado ampliar a pedidos fuera de España:



Liniaspedido.csv, cada pedido tiene diferentes líneas y en este csv recogemos que productos y su cantidad van a cada pedido:

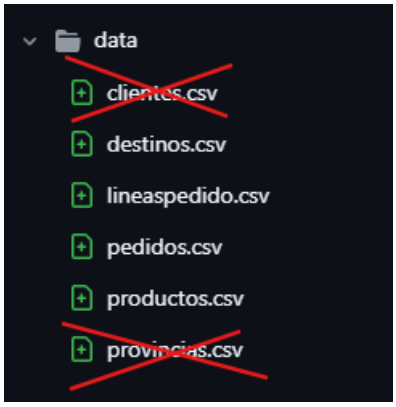


	LineaPedidoID	PedidoID	ProductoID	Cantidad
1	1	1	5	8
2	2	1	28	1
3	3	1	29	6
4	4	1	3	1
5	5	1	38	1

Tratamiento de datos

Para el calculo de asignación de productos en vehículos y sus rutas óptimas no necesitamos toda esta información, para ello hemos decidido, en base a los datasets origen, modificar y eliminar estos para conseguir trabajar con los mínimos datos necesarios para nuestros objetivos.

Para ello, hemos decidido no usar/eliminar de los cálculos tanto la información de los clientes como la de las provincias:



El razonamiento, en el caso de *provincias.csv* como hemos visto anteriormente en el ejemplo de los datos de este dataset, no nos aporta información relevante para nuestros objetivos. No necesitamos saber ni en que provincia están las ciudades destino, ni en que país para poder calcular las rutas.

El razonamiento para el caso de *clientes.csv* es el mismo, no nos hace falta saber la información del cliente al que vamos a repartir para conseguir los objetivos de tener las mejores rutas y el mejor reparto de productos en los camiones.

Este filtraje de datasets nos deja con 4, destinos.csv, liniaspedido.csv, pedidos.csv y productos.csv, todos con datos imprescindibles para los cálculos que vamos a realizar.

No vamos a usar todos los datasets tal cual, pues hay algunos que no están bien optimizados para la solución que proponemos por lo que vamsos ha hacerles algunas modificaciones.

Los datasets que no van a sufrir modificaciones son:

Pedidos.csv:

data > pedidos.csv

	PedidoID	FechaPedido	ClienteID	DestinoEntregaID
1	1	2025-01-02	68	11
2	2	2025-01-03	6	39
3	3	2025-01-04	22	26
4	4	2025-01-05	3	1
5	5	2025-01-06	78	16

Productos.csv:

data > productos.csv

	ProductoID	Nombre	PrecioVenta	TiempoFabricacionMedia	Caducidad
1	1	Producto1	45.84	3	9
2	2	Producto2	54.20	9	27
3	3	Producto3	74.19	7	21
4	4	Producto4	53.82	1	3
5	5	Producto5	64.84	1	3
6	6	Producto6	85.57	6	18
7	7	Producto7	49.53	4	12

Liniaspedido.csv:

data > lineaspedido.csv

	LineaPedidoID	PedidoID	ProductoID	Cantidad
1	1	1	5	8
2	2	1	28	1
3	3	1	29	6
4	4	1	3	1
5	5	1	38	1
6	6	1	44	9

Modificaciones en los datos:

Por lo tanto, el único dataset que va a ser modificado va a ser el de destinos.csv:

data > destinos_geocodificados.csv

	DestinoID	nombre_completo	distancia_km	coordenadas_gps	provinciaID	ciudad
1	1	Destino Castellón	242.85	40.2518568, -0.0615052	12	Castellón
2	2	Destino Segovia	529.57	40.944489, -4.115206	40	Segovia
3	3	Destino Valladolid	575.06	41.6521807, -4.728605	47	Valladolid
4	4	Destino Toledo	550.05	39.8558913, -4.024265	45	Toledo
5	5	Destino Girona	85.5	41.9793006, 2.8199439	17	Girona
6	6	Destino Soria	388.44	41.6012505, -2.721938	42	Soria
7	7	Destino Córdoba	711.53	37.8845813, -4.7760138	14	Córdoba

Se ha decidió añadir una coluna entera solo con el nombre de la ciudad y añadir datos a la columna que no tenia datos de las coordenadas exactas de las ciudades, así como la eliminación de destinos fuera del territorio de la península Ibérica como Canarias, Ceuta, Melilla y Baleares.

Para conseguir esto se ha creado un script de Python a parte del proyecto que en base al dataset base de destinos.csv generaba este resultante *destinos_geocodificados.csv*.

Para el tema de conseguir las coordenadas de las ciudades se ha usado una librería de Python en este script a parte llamada ***GEOPY,*** que te permite en base al nombre de una ciudad en concreto, conseguir sus coordenadas, ejemplo de uso de geopy:

```python
from geopy.geocoders import Nominatim

geolocator = Nominatim(user_agent="mi_aplicacion")
location = geolocator.geocode("Barcelona")

print(location.latitude, location.longitude)
```

Mas info de la librería en su web https://geopy.readthedocs.io/en/stable/

Análisis de tecnologías

Desglose de tecnologías usadas en el proyecto:

Librerías de python

```
You, 1 hour ago | 1 author (You)
import streamlit as st
import pandas as pd
import folium
import openrouteservice
from openrouteservice import convert
from folium.plugins import MarkerCluster
import colorsys
import math
import numpy as np
from sklearn.cluster import KMeans
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import itertools
import copy
from typing import List, Dict
```

Streamlit (streamlit)

- Permite crear interfaces web interactivas de manera sencilla.
- Se usa para subir CSVs, mostrar tablas, configurar parámetros y mostrar resultados y mapas en la aplicación.

Pandas (pandas)

- Manejo de datos tabulares.
- Se utiliza para cargar, limpiar, transformar y combinar los datasets de pedidos, destinos, líneas de pedido y productos.

Folium (folium) y folium.plugins.MarkerCluster

- Creación de mapas interactivos basados en Leaflet.
- Se usa para visualizar rutas de camiones y agrupaciones de destinos mediante clusters de marcadores.

OpenRouteService (openrouteservice)

- API para cálculo de rutas y distancias reales en carretera.
- Permite obtener rutas detalladas y distancias entre puntos geográficos, mejorando la estimación de kilómetros recorridos.

Matemáticas y utilidades (math, itertools, copy, colorsys)

- Cálculos de distancias geográficas (math), combinaciones y permutaciones (itertools), copias profundas de objetos (copy) y conversión de colores HSV a hexadecimal (colorsys).

NumPy (numpy)

- Operaciones matemáticas y manipulación de arrays, principalmente en cálculos de scores y normalización de datos.

Scikit-learn (sklearn)

- KMeans (sklearn.cluster): clustering geográfico de pedidos en zonas para optimizar rutas.
- RandomForestClassifier (sklearn.ensemble): modelo de machine learning para predecir la mejor asignación de pedidos a camiones considerando capacidad, cluster, prioridad y distancia.
- train_test_split y accuracy_score (sklearn.model_selection, sklearn.metrics): división de datos en entrenamiento/prueba y evaluación de la precisión del modelo.

Typing (List, Dict)

- Tipado estático de funciones para mayor claridad y robustez en el código.

API

OpenRouteService API

- Proporciona cálculo de rutas y distancias basadas en carreteras reales.
- Permite optimizar rutas de camiones más allá de la simple distancia geográfica (“a vuelo de pájaro”).

Antes del cálculo

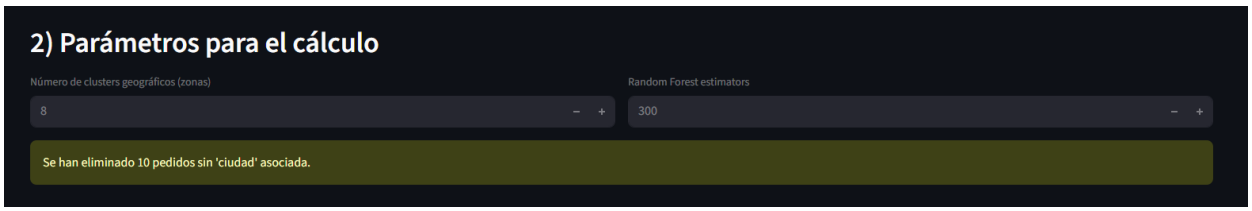
Antes de ejecutar el cálculo de asignación de pedidos a camiones, se muestran ciertos parámetros clave utilizados en el proceso. Estos parámetros no son editables, ya que sirven para dejar constancia de cómo se han hecho las pruebas y llegar a la configuración que produce la mejor optimización:

1. Número de nodos/clusters (n_clusters): se utiliza en la fase de KMeans para la zonificación geográfica de los pedidos. Cada nodo representa un centro geográfico que agrupa pedidos cercanos entre sí, ayudando a reducir la distancia total recorrida por cada camión.
2. Número de árboles (n_estimators): se aplica en el Random Forest que asigna los pedidos a los camiones. Cada árbol contribuye a la predicción de la mejor asignación basada en las características del pedido y del camión, mejorando la robustez y exactitud del modelo.

Además, antes de comenzar los cálculos se realiza un control de datos para evitar errores:

- Se han eliminado destinos fuera de la península, por lo que pueden existir pedidos “huérfanos” sin ciudad válida asociada.
- Estos pedidos se detectan automáticamente y se eliminan del conjunto de datos, garantizando que solo se incluyan pedidos con coordenadas y ciudades válidas en el cálculo.

Con esto, el sistema asegura que los datos sean consistentes y que los resultados reflejen únicamente escenarios posibles, dejando constancia clara de los parámetros y del filtrado de pedidos, aunque el usuario no pueda modificarlos desde la interfaz.



Flujo de cálculo

El modelo consta de tres partes: primero, una vez subidos los CSV, se agrupan los pedidos según su cercanía geográfica, luego, se asignan a los camiones usando un modelo de Random Forest para decidir qué pedidos van en cada vehículo y por último, se calcula el orden de las rutas dentro de cada camión para asegurar que los recorridos sean lo más eficientes posible:

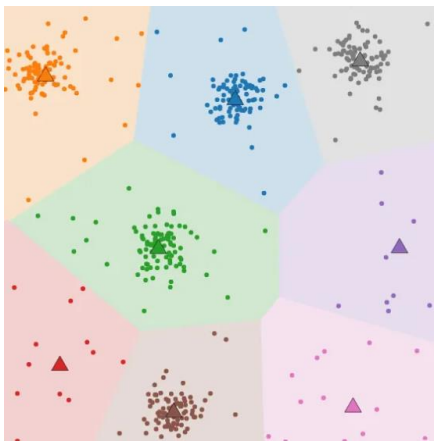
Definición modelos usados

Definición de modelos usados en el orden mostrado anteriormente:

1. **Agrupación geográfica de pedidos con KMeans**

Primero, partimos de los datos de los pedidos que hemos subido: cada pedido tiene su destino con coordenadas GPS. Para simplificar la planificación de rutas, usamos KMeans para agrupar los pedidos según su proximidad geográfica. Esto nos permite dividir la zona de reparto en “clusters” o subzonas.

Cada pedido recibe un ClusterID, que indica a qué zona pertenece. Esta etapa no decide aún qué camión va a cada pedido ni el orden de las visitas; simplemente nos da una manera de organizar los pedidos por cercanía para hacer más eficiente la siguiente fase.



2. **Asignación de pedidos a camiones con Random Forest**

Una vez tenemos los clusters, necesitamos decidir qué camión reparte cada pedido. Para ello usamos un Random Forest de clasificación, entrenado con ejemplos sintéticos basados en criterios prácticos: capacidad del camión, cantidad de pedido, proximidad al cluster, pedidos ya asignados al camión y prioridad por tiempo de entrega.

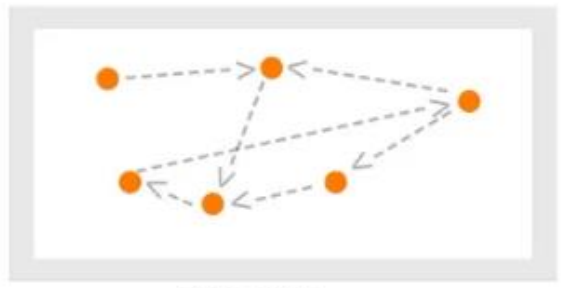
El modelo nos devuelve, para cada pedido, la probabilidad de que un camión específico sea la mejor opción. Con esto conseguimos asignar los pedidos a los camiones de forma eficiente, asegurando que los camiones estén bien cargados y se respeten prioridades de tiempo, pero sin definir todavía el orden de visita dentro de cada camión.

3. **Optimización del orden de rutas dentro de cada camión**

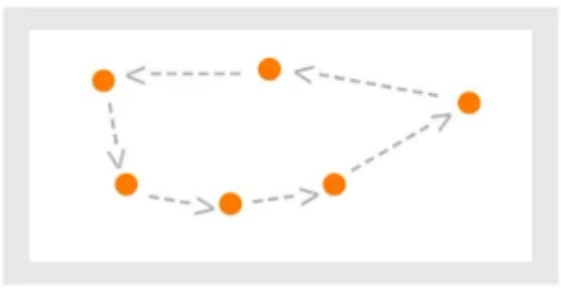
Finalmente, una vez asignados los pedidos a cada camión, necesitamos determinar el recorrido óptimo, es decir, en qué orden visitar los destinos. Para esto usamos:

1. Nearest neighbor: comenzamos desde el origen y vamos eligiendo siempre el pedido más cercano como siguiente parada.
2. 2-opt: aplicamos una mejora local que prueba intercambiar pares de pedidos para reducir la distancia total de la ruta.

Con esto, conseguimos rutas eficientes: por ejemplo, el camión que sale de Barcelona y reparte a Madrid y Vigo visitará primero el pedido más cercano a Barcelona y luego optimizará el orden restante para minimizar kilómetros recorridos, evitando rutas ilógicas como Barcelona → Vigo → Madrid.



FROM THIS



TO THIS

Visualización de resultados

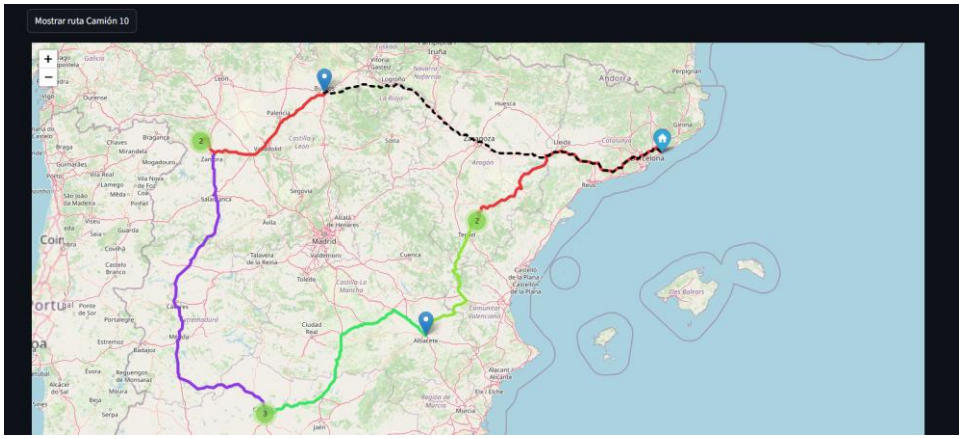
Una vez tenemos el resultado del cálculo, a través de los tres pasos usados (KMeans → RF → optimización local), este viene en un formato muy concreto: una lista de camiones (model_camiones) con, para cada camión, un DataFrame de pedidos ordenados y otra estructura (model_trucks_raw) con métricas por camión (km total, km hasta el último pedido, coste estimado, nº de pedidos, etc.).

Por eso se ha decidido mostrar los resultados en dos vías complementarias dentro de Streamlit:

- Mapa interactivo con folium (renderizado en Streamlit via st.components.v1.html) usando la API de OpenRouteService cuando esté disponible para obtener trazados reales por carretera y como apoyo y fallback, trazados euclídeos calculados con la función Haversine si ORS no responde.
- Tablas y resúmenes con pandas/st.dataframe y expanders en Streamlit para ver, camión a camión, la lista de pedidos, cantidades, TiempoTotalEstimado, ClusterID y las métricas agregadas (km, coste, carga).

Qué verás en el mapa y por qué:

- Un marcador para el origen (Mataró) y marcadores agrupados (MarkerCluster) para los pedidos.
- Linias de colores que representan cada tramo de la ruta, cuando usamos ORS se decodifica la geometría real y se pinta la ruta por carretera, si no, se dibuja una línea directa entre puntos.
- Colores por tramo (generados con hsv_to_hex) para identificar el orden de visitas y una línea discontinua negra para el retorno al origen.
- Popups y tooltips con la información del pedido (ID, ciudad) para inspección rápida.



Cosas importantes para entender el dibujo de la ruta en el mapa, cada color representa una ruta, de una parada a otra, una parada significa entregar en un destino 1 o mas pedidos, esto se puede apreciar en los puntos que hay un número.

Este numero muestra el numero de pedidos que se van a entregar en esa parada, si no hay número, significa que la parada solo entrega un pedido.

La ruta claramente diferente, de color negro y con una linia subrayada, plasma la vuelta al origen.

Qué se muestra en las tablas / paneles:

- Para cada camión un resumen: número de pedidos, carga total, km totales y coste estimado de combustible; además una tabla con las filas de pedidos en el orden de visita (PedidoID, ciudad, cantidad, TiempoTotalEstimado, cluster).
- Botones individuales “Mostrar ruta Camión X” que llaman a la función mostrar_mapa_camion_from_df() para renderizar la ruta concreta del camión seleccionado.
- Un resumen global (dataframe) con km y coste por camión y totales acumulados.

Robustez y UX:

- Si la API de ORS falla o no hay clave, el sistema sigue funcionando: calcula distancias y rutas aproximadas con Haversine y sigue mostrando tablas. Se muestran avisos claros en Streamlit cuando esto ocurre.
- Se incluye una leyenda fija en el mapa y mensajes informativos con tiempos estimados de conducción (horas/días) calculados a partir de la velocidad media configurada.
- Es recomendable añadir opciones de exportar (CSV) de las asignaciones por camión y capturas de pantalla para auditoría operativa.

Datos complementarios

Dentro de mostrar cada camión, usando lógica simple pues los datos finales de los cálculos ya los tenemos almacenados en variables despues del paso por los modelos, podemos printar información importante por cada ruta de cada camión.

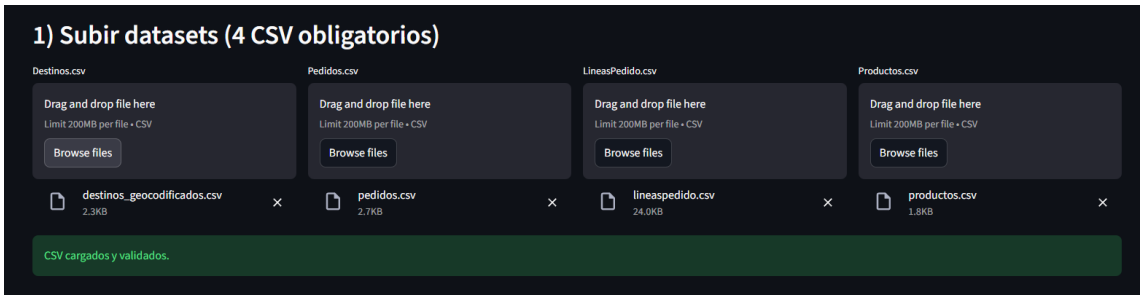
Resultados de pruebas unitarias, de integración y de aceptación

Se han realizado pruebas para confirmar que el conjunto de implementación de diferentes tecnologias funciona correctamente:

- Objetivo: Comprobar que las funciones individuales trabajan correctamente de manera aislada.
- Funciones probadas:
    - haversine(lat1, lon1, lat2, lon2) → calcula distancia entre dos puntos geográficos.
    - nearest_neighbor_order() → ordena paradas según la distancia más cercana.
    - estimate_marginal_km() → calcula el km marginal al asignar un pedido a un camión.
    - two_opt_route_order() → mejora local del orden de las paradas para reducir km.
- Resultado:
    - Todas las funciones devuelven valores coherentes con casos de prueba manuales.
    - Ejemplo: la distancia entre dos puntos conocidos coincidió con la esperada con margen < 0.1 km.

Pruebas de integración

- Objetivo: Comprobar que los módulos combinados funcionan correctamente.
- Integración probada:
    - Carga y validación de CSV → unión con Pandas → cálculo de clusters con KMeans → preparación de dataset para Random Forest → asignación de pedidos a camiones.
    - Visualización en mapa con Folium y rutas calculadas con OpenRouteService.
- Resultado:
    - El flujo completo permite subir los 4 CSV y generar la asignación de pedidos.



- Los pedidos se agrupan correctamente en clusters y se asignan camiones respetando capacidad y prioridad.
- La visualización de rutas muestra las paradas correctamente en el mapa y se generan rutas plausibles.


Preview pedidos (con cluster):

	PedidoID	ciudad	latitud	longitud	Cantidad_total	TiempoTotalEstimado	ClusterID	DistanciaCluster_km
0	1	Guadalajara	40.74	-2.5059	86	4	1	111.78
1	2	Sevilla	37.3886	-5.9953	107	4	3	79.2225
2	4	Castellón	40.2519	-0.0615	104	4	6	76.6931
3	5	Navarra	42.6125	-1.8308	99	4	7	46.2591
4	6	Lugo	43.0396	-7.4566	100	4	0	155.3583
5	8	Palencia	42.4096	-4.6161	110	4	4	31.4048
6	9	La Rioja	42.3286	-2.4675	127	4	7	42.9847
7	10	Vizcaya	43.2385	-2.8516	95	8	7	70.1626
8	11	Burgos	42.3439	-3.697	106	4	4	46.4372
9	12	Guipúzcoa	43.1445	-2.2038	132	4	7	50.4417

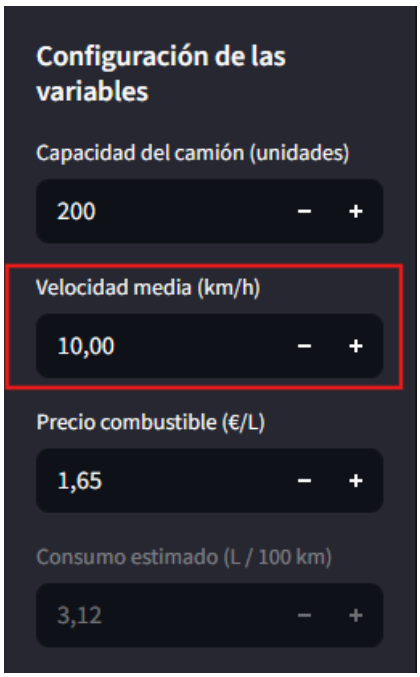
Pruebas de aceptación (usuario final / criterios de negocio)

- Objetivo: Verificar que la aplicación cumple con los requisitos funcionales.
- Criterios comprobados:
  1. Subida y validación de CSVs de pedidos, destinos, líneas de pedido y productos.
  2. Generación de clusters geográficos coherentes con las coordenadas.
  3. Entrenamiento del modelo Random Forest y asignación de pedidos minimizando kilómetros y respetando tiempos y capacidad.
  4. Visualización clara de rutas por camión y cálculo de costos y km totales.
- Resultado:
  - Todas las funciones principales cumplen con los objetivos.
  - La aplicación puede ser usada por un usuario final para obtener asignaciones de pedidos eficientes y rutas optimizadas.

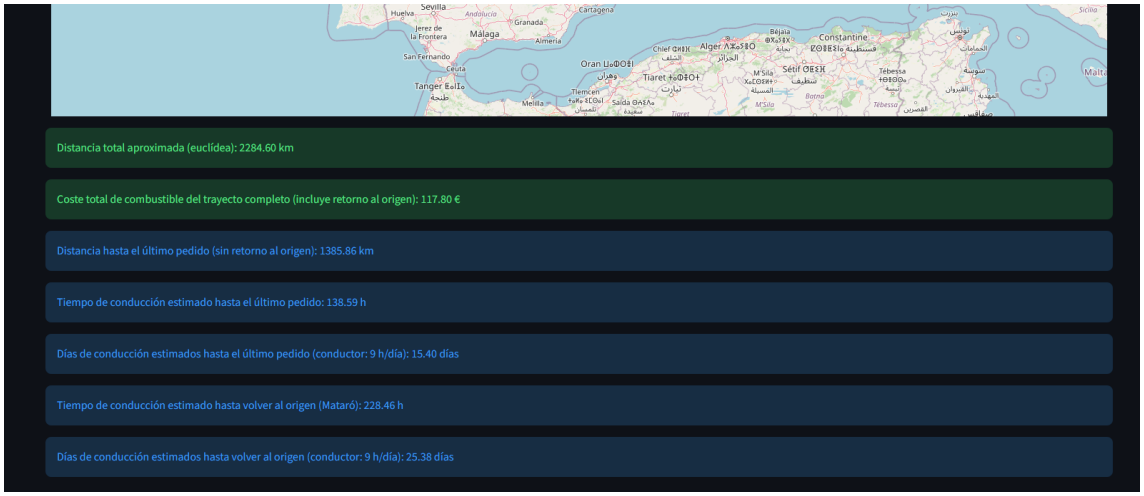
Análisis de rendimiento de resultados

Para el análisis de resultados, después de las pruebas realizadas para encontrar la mejor configuración para el modelo de cálculo, podemos verlo con los resultados que la propia aplicación muestra, al mostrar el tiempo estimado que un camión va a estar para entregar el ultimo pedido, podemos comprobar ese

Es posible crear escenarios imposibles para conseguir nuestros objetivos, por ejemplo poniendo camiones con características no realistas:



En este caso, configurar que los camiones vayan a una velocidad media muy baja, lo único que consigue es reducir el consumo de combustible pero vuelve imposible hacer entregas, como podemos ver en el resumen de datos por ruta:



Podemos ver como al haber configurado una velocidad media tan baja, tardas en llegar al ultimo destino mas del tiempo que tienes antes de que caduque para el ultimo producto.

En estos casos, la aplicación lo único que puede hacer es darte dentro de las características configuradas, el mejor resultado posible, aunque no cumpla algún de los requerimientos posibles.

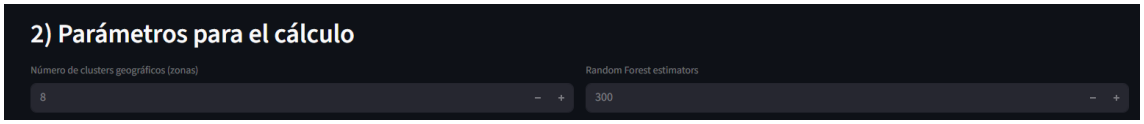
Configuración de los modelos

Para el tema de conseguir la mejor configuración para los modelos posible para conseguir mejor rendimiento y un resultado optimo, se ha hecho a base de prueba y error para estos dos campos:

Se ha comprobado como aumentar el numero de clusters y árboles para el cálculo, es útil siempre y cuando sea de forma razonable.

Subir los dos valores de forma descontrolada consigue dos cosas, la primera es ligera mejora en la precisión de cálculo y un aumento muy significativo del tiempo para el cálculo:

Se ha llegado a la conclusión que la mejor configuración para estos 2 valores es el que esta predefinido en la pantalla del modelo, donde se muestra, pero no es editable:



Conclusiones y mejoras futuras

Conclusiones

La solución integra de forma coherente las principales fases necesarias para el tratamiento y análisis de los datos:

- Carga, validación y limpieza de los conjuntos de datos de pedidos, destinos, líneas de pedido y productos.
- Tratamiento y enriquecimiento de los datos, incluyendo el cálculo de métricas clave como el TiempoTotalEstimado, que permite priorizar pedidos en función de restricciones de fabricación y caducidad.
- Agrupación geográfica de pedidos mediante KMeans, reduciendo la complejidad del problema y favoreciendo rutas más eficientes.
- Asignación de pedidos a camiones mediante un modelo de Random Forest, teniendo en cuenta capacidad, cercanía geográfica y prioridad temporal.
- Optimización del orden de reparto dentro de cada camión mediante heurísticas inspiradas en el problema del viajante.
- Visualización de resultados a través de mapas interactivos y tablas resumen, facilitando la interpretación y validación de las rutas generadas.

El resultado final es un prototipo funcional, flexible y extensible, capaz de generar planes de reparto coherentes y eficientes, cumpliendo los objetivos planteados al inicio del proyecto y ofreciendo una base sólida para futuras ampliaciones.

Mejoras futuras

Como posibles líneas de mejora y evolución del sistema, se identifican las siguientes:

- Incorporar una validación automática de factibilidad que detecte pedidos inviables en función de los tiempos de transporte reales y la caducidad disponible.
- Permitir el fraccionamiento de pedidos o la asignación parcial a distintos camiones en casos críticos.
- Integrar más restricciones logísticas reales, como ventanas horarias de entrega, turnos de conductores o límites de conducción.
- Evaluar algoritmos de optimización de rutas más avanzados o híbridos cuando el volumen de pedidos sea mayor.
- Mejorar la precisión de los cálculos de distancia y tiempo mediante un uso más intensivo de APIs de rutas o datos históricos de tráfico.
- Automatizar completamente el proceso de ejecución y generar informes exportables para su uso operativo.
- Desplegar la aplicación en un entorno productivo o cloud, facilitando su uso por parte de usuarios finales.

Video corporativo

https://youtu.be/cHQnxPLpgkI