ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN BIG DATA: TECNOLOGÍA Y ANALÍTICA AVANZADA

DESARROLLO DE MODELO DE ANALÍTICA AVANZADA PARA OPTIMIZACIÓN EN TRANSICIÓN DE FLOTA

Autor: Jorge Gómez Berenguer

Director: Domingo Miguel Guinea García-Alegre

Madrid

Junio 2021

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

DESARROLLO DE MODELO DE ANALÍTICA AVANZADA PARA OPTIMIZACIÓN EN TRANSICIÓN DE FLOTA

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2020/21 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Jorge Gómez Berenguer Fecha: 18/ 06/ 2021

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Domingo Miguel Guinea García-Alegre Fecha: 18/ 06/ 2021

Vº Bº del Coordinador de Proyectos

Fdo.: Carlos Morrás Ruiz-Falcó Fecha: ……/ ……/ ……

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Jorge Gómez Berenguer

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: DESARROLLO DE MODELO DE ANALÍTICA AVANZADA PARA OPTIMIZACIÓN EN TRANSICIÓN DE FLOTA, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

- El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 18 de junio de 2021

ACEPTA

Fdo Jorge Gómez Berenguer

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN BIG DATA: TECNOLOGÍA Y ANALÍTICA AVANZADA

DESARROLLO DE MODELO DE ANALÍTICA AVANZADA PARA OPTIMIZACIÓN EN TRANSICIÓN DE FLOTA

Autor: Jorge Gómez Berenguer

Director: Domingo Miguel García Guinea-Alegre

Madrid

Junio 2021

Agradecimientos

DESARROLLO DE MODELO DE ANALÍTICA AVANZADA PARA OPTIMIZACIÓN EN TRANSICIÓN DE FLOTA

Autor: Gómez Berenguer, Jorge.
Director: Guinea García-Alegre, Domingo Miguel.
Entidad Colaboradora: Ozone Drive S.L.

RESUMEN DEL PROYECTO

El presente proyecto se ha creado bajo marco de desarrollo de una aplicación de tipo *SAAS*, que busca optimizar las transiciones energéticas de flotas de vehículos empresariales a través de técnicas de Machine Learning, y utilización de Big Data

Palabras clave: Flotas, consumo, optimización de costes

1. Introducción

Hasta el momento, los cambios en las flotas empresariales se han realizado sin ningún criterio técnico que las respaldase. Estas decisiones se basaban en criterios subjetivos sin corroborar la viabilidad técnica de la misma, algunos de estos eran la creencia de la adecuación de un vehículo u otro, u ofertas comerciales recibidas.

La startup tecnológica Ozone Drive ha desarrollado una aplicación que pretende dar soporte a los gestores de flotas empresariales en esta decisión para realizar una transición energética de flota óptima, es decir, que se ajuste lo máximo posible a las necesidades particulares de cada conductor, reduciendo así las emisiones y costes totales anuales.

2. Definición del proyecto

El proyecto busca solucionar las necesidades de negocio detectadas por la empresa entre los potenciales clientes. Serán necesarias técnicas de Machine Learning para la predicción de consumos, y de extracción de datos de diferentes sitios web.

El fin último del proyecto será la creación de un informe interactivo mediante la utilización de la herramienta Power Bi, para dotar al cliente la flexibilidad de interactuar con la solución.

3. Descripción del modelo/sistema/herramienta

El presente proyecto se encuentra englobado en el desarrollo de esta aplicación, tratando la cadena de valor del dato de forma completa. Se crearán por lo tanto los procesos de extracción, transformación y almacenamiento de datos, así como los modelos de predicción y cálculo de costes.

Dado que se tratan técnicas muy diversas, se dividirá el completo en pequeños desarrollos modulares que posteriormente se conectarán dando lugar a una herramienta que funcione como un único ente.

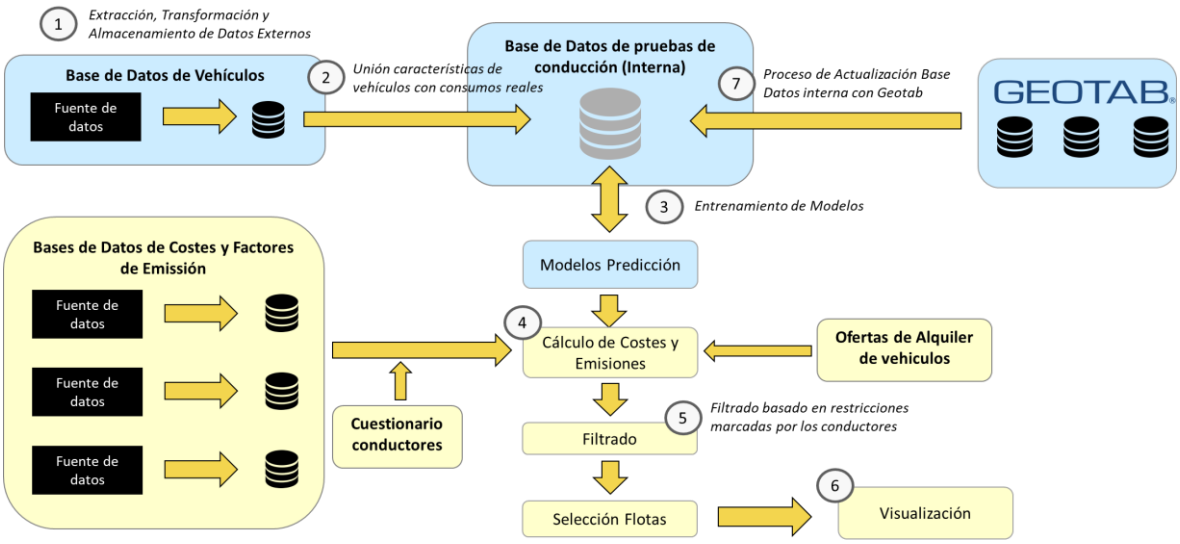

Ilustración 1.1: Arquitectura Básica de la solución

4. Resultados

El resultado final de la herramienta se presentará a través de un informe Power Bi, en el que, de una forma clara el cliente puede conocer qué flotas son las más adecuadas para una transición óptima en costes y emisiones.

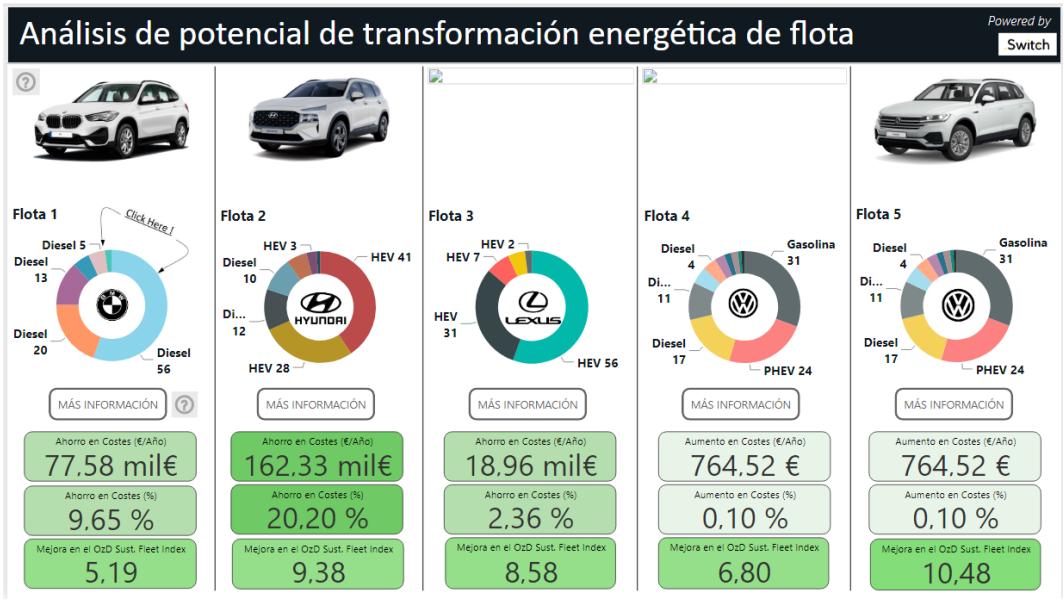

Ilustración 1.2: Primera pestaña informe Power Bi

5. Conclusiones

Dado que las predicciones de consumo generadas presentan unas métricas cercanas a la máxima precisión posible, el cálculo de costes de consumo será bastante preciso, por lo que el resultado final se traducirá en un Mínimo Producto Viable válido para la comercialización.

En futuros desarrollos se desarrollarán nuevas funcionalidades que doten a la herramienta de unas mejores capacidades.

6. Referencias

[1] Asociación Empresarial para el Desarrollo e Impulso del Vehículo Eléctrico (AEDIVE) (septiembre 3, 2020). Ozone Drive, startup tecnológica especializada en servicios avanzados de movilidad eléctrica. https://aedive.es/ozone-drive-startup-tecnologica-novilidad -electrica/

[2] ElReferente (mayo 2, 2016). Ozone Drive busca financiación para crear más vehículos eléctricos. https://elreferente.es/tecnologicos/ozone-drive-busca-financiacion-para-crear-mas-coches-electricos/

[3] Indiegogo (mayo 21, 2015). Ozone Drive, the Silent Revolution of Electric Cars. https://www.indiegogo.com/projects/ozone-drive-connecting-islands-to-electric-cars#/

[4] Bulut, O. (enero 11, 2021). Effective Feature Selection: Recursive Feature Elimination Using R. https://towardsdatascience.com/effective-feature-selection-recursive-feature-elimination-using-r-148ff998e4f7

[5] Brownlee, J. (agosto 28, 2020). Recursive Feature Elimination (RFE) for Feature Selection in Python. Machine Learning Mastery. https://machinelearningmastery.com/rfe-feature-selection-in-python/

[6] Badr, W. (enero 5, 2019). 6 Different Ways to Compensate for Missing Values In a Dataset. https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779

[7] Gelman, A. y Hill, J. (septiembre 2012). “Missing data imputation”. Data Analysis Using Regression and Multilevel/Hierarchical Models. New York, NY, USA pp 529-544

[8] Amat, J. (mayo, 2020). Detección de anomalías: Isolation Forest. cienciadedatos.net. https://www.cienciadedatos.net/documentos/66_deteccion_anomalias_isolationforest.html#Introducci%C3%B3n

[9] Lewinson, E. (julio 2, 2018). Outlier Detection with Isolation Forest. https://towardsdatascience.com/outlier-detection-with-isolation-forest-3d190448d45e

[10] Lewinson, E. (marzo 10, 2019). Outlier Detection with Extended Isolation Forest. https://towardsdatascience.com/outlier-detection-with-extended-isolation-forest-1e248a3fe97b

[11] Tony, F., Ming, K. y Zhi-Hua, Z. Isolation Forest

[12] Amat, J. (abril, 2020). Machine Learning con R y mlr3. cienciadedatos.net. https://www.cienciadedatos.net/documentos/60_machine_learning_con_r_y_mlr3#An%C3%A1lisis_exploratorio

[13] Muñoz, A., Sánchez E. y Portela J. (octubre, 2020). Chapter 2 – Classification. Machine Learning I. Universidad Pontificia de Comillas

[14] Amat, J. (abril, 2020). Machine Learning con R y Caret. cienciadedatos.net. https://www.cienciadedatos.net/documentos/41_machine_learning_con_r_y_caret#Conclusi%C3%B3n_an%C3%A1lisis_exploratorio

[15] Muñoz, A., Sánchez E. y Portela J. (octubre, 2020). Chapter 3 – Regression. Machine Learning I. Universidad Pontificia de Comillas

[16] Sánchez, E. (enero 2021). Ensemble Learning. Machine Learning II. Universidad Pontificia de Comillas

[17] Amat, J. (febrero, 2017). Árboles de decisión, random forest, gradient boosting C5.0. cienciadedatos.net. https://www.cienciadedatos.net/documentos/33_arboles_decision_random_forest_gradient_boosting_c50#%C3%81rboles_de_regresi%C3%B3n

[18] Brownlee, J. (abril 20, 2020). How to develop a Random Forest Ensemble in Python. Machine Learning Mastery. https://machinelearningmastery.com/random-forest-ensemble-in-python/

[19] Kuhn, M. (marzo 27, 2019). 5 – Model Training and tuning. The Caret Package. https://topepo.github.io/caret/model-training-and-tuning.html

[20] Dive Into Deep Learning. 4.1 Multilayer Perceptron. http://d2l.ai/chapter_multilayer-perceptrons/mlp.html

[21] Baumbah, L. (julio 2020). Neural Network – Activation Function. https://morioh.com/p/21b55ba475f9

[22] Stöttner, T. (mayo 16, 2019). Why Data should be Normalized before Training a Neural Network. https://towardsdatascience.com/why-data-should-be-normalized-before-training-a-neural-network-c626b7f66c7d

[23] Heim, R. (diciembre 26, 2019). How to Train a Multilayer Perceptron Neural Network. All About Circuits. https://www.allaboutcircuits.com/technical-articles/how-to-train-a-multilayer-perceptron-neural-network/

[24] Pizarro, J., Portela, J. y Muñoz, A. (febrero 2021). NeuralSens: Sensitivity Analysys of Neural Networks.

[25] JJ (marzo 23, 2016). MAE and RMSE — Which Metric is Better?. https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d

[26] Sab (junio 2, 2019). MAE vs MSE vs RMSE. http://zerospectrum.com/2019/06/02/mae-vs-mse-vs-rmse/

MODELO DE PREDICCIÓN DE PRECIOS EN TIEMPO REAL PARA EL MERCADO ELÉCTRICO RESIDENCIAL

Author: Gómez Berenguer, Jorge.
Supervisor: Guinea García-Alegre, Domingo Miguel.
Collaborating Entity: Ozone Drive S.L.

ABSTRACT

This project has been created under the framework of the development of a *SAAS* type application, which seeks to optimize the energy transitions of fleets of business vehicles through Machine Learning techniques and the use of Big Data.

Keywords: Vehicles Fleet, Comsuption, Costs optimization

1. Introduction

Until now, changes in company fleets have been made without any technical criteria to back them up. These decisions were based on subjective criteria without corroborating the technical feasibility of it, some of these were the belief of the suitability of one vehicle or another, or commercial offers received.

The technology startup Ozone Drive has developed an application that aims to support business fleet managers in this decision to make an optimal fleet energy transition, i.e., that fits as much as possible to the needs of each driver, thus reducing emissions and total annual costs.

2. Project definition

The project seeks to solve the business needs detected by the company among potential customers. Machine Learning techniques will be necessary for the prediction of consumption, and data extraction from different websites.

The ultimate goal of the project will be the creation of an interactive report using the Power Bi tool, to give the client the flexibility to interact with the solution.

3. Model/Tools/System description

The present project is included in the development of this application, dealing with the entire data value chain. Therefore, the extraction, transformation and storage processes will be created, as well as the prediction and cost calculation models.

Since it deals with very different techniques, the whole will be divided into small modular developments that will later be connected giving rise to a tool that works as a single entity.

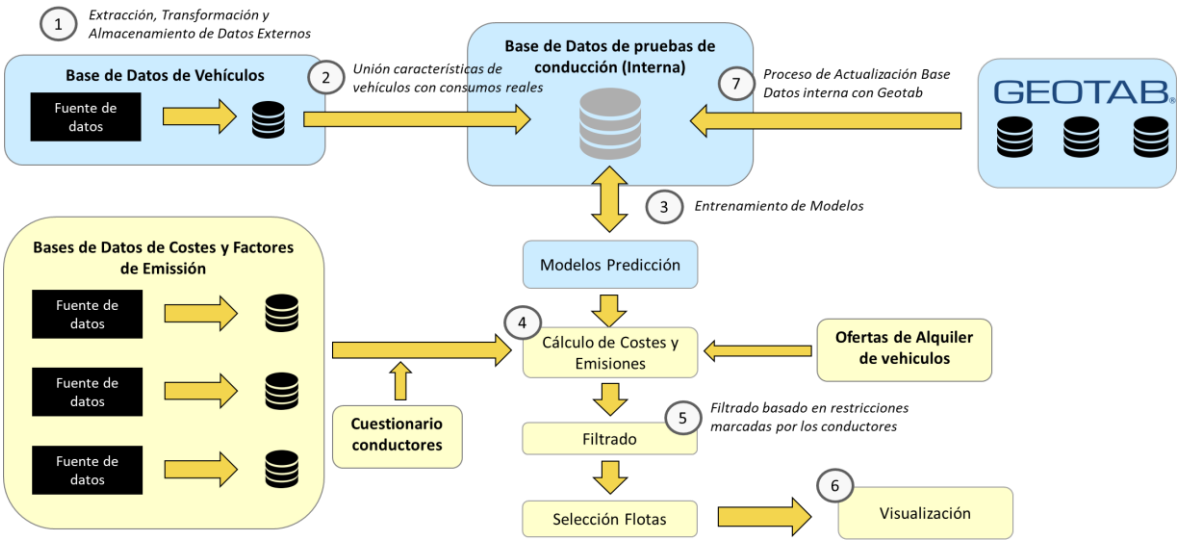

Ilustración 1.3: Basic Architecture of the solution

4. Results

Results will be presented through a Power Bi report, in which the customer can clearly know which fleets are the most suitable for an optimal transition in terms of costs and emissions.

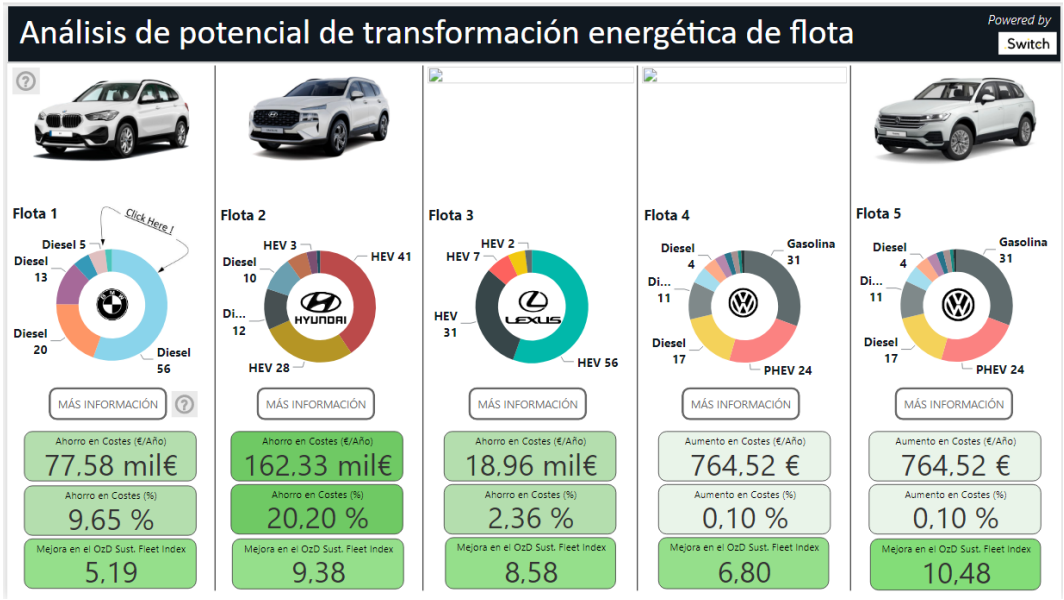

Ilustración 1.4: Power Bi Report First Sheet

5. Conclusions

It has been obtained a Minimum Viable Product valid for Ozone Drive commercial purposes, since the generated consumption predictions present metrics close to the highest possible accuracy.

In future developments, new functionalities will be developed to provide the tool with better capabilities.

6. References

[1] Asociación Empresarial para el Desarrollo e Impulso del Vehículo Eléctrico (AEDIVE) (septiembre 3, 2020). Ozone Drive, startup tecnológica especializada en servicios avanzados de movilidad eléctrica. https://aedive.es/ozone-drive-startup-tecnologica-novilidad -electrica/

[2] ElReferente (mayo 2, 2016). Ozone Drive busca financiación para crear más vehículos eléctricos. https://elreferente.es/tecnologicos/ozone-drive-busca-financiacion-para-crear-mas-coches-electricos/

[3] Indiegogo (mayo 21, 2015). Ozone Drive, the Silent Revolution of Electric Cars. https://www.indiegogo.com/projects/ozone-drive-connecting-islands-to-electric-cars#/

[4] Bulut, O. (enero 11, 2021). Effective Feature Selection: Recursive Feature Elimination Using R. https://towardsdatascience.com/effective-feature-selection-recursive-feature-elimination-using-r-148ff998e4f7

[5] Brownlee, J. (agosto 28, 2020). Recursive Feature Elimination (RFE) for Feature Selection in Python. Machine Learning Mastery. https://machinelearningmastery.com/rfe-feature-selection-in-python/

[6] Badr, W. (enero 5, 2019). 6 Different Ways to Compensate for Missing Values In a Dataset. https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779

[7] Gelman, A. y Hill, J. (septiembre 2012). "Missing data imputation". Data Analysis Using Regression and Multilevel/Hierarchical Models. New York, NY, USA pp 529-544

[8] Amat, J. (mayo, 2020). Detección de anomalías: Isolation Forest. cienciadedatos.net. https://www.cienciadedatos.net/documentos/66_deteccion_anomalias_isolationforest.html#Introducci%C3%B3n

[9] Lewinson, E. (julio 2, 2018). Outlier Detection with Isolation Forest. https://towardsdatascience.com/outlier-detection-with-isolation-forest-3d190448d45e

[10] Lewinson, E. (marzo 10, 2019). Outlier Detection with Extended Isolation Forest. https://towardsdatascience.com/outlier-detection-with-extended-isolation-forest-1e248a3fe97b

[11] Tony, F., Ming, K. y Zhi-Hua, Z. Isolation Forest

[12] Amat, J. (abril, 2020). Machine Learning con R y mlr3. cienciadedatos.net. https://www.cienciadedatos.net/documentos/60_machine_learning_con_r_y_mlr3#An%C3%A1lisis_exploratorio

[13] Muñoz, A., Sánchez E. y Portela J. (octubre, 2020). Chapter 2 – Classification. Machine Learning I. Universidad Pontificia de Comillas

[14] Amat, J. (abril, 2020). Machine Learning con R y Caret. cienciadedatos.net. https://www.cienciadedatos.net/documentos/41_machine_learning_con_r_y_caret#Conclusi%C3%B3n_an%C3%A1lisis_exploratorio

[15] Muñoz, A., Sánchez E. y Portela J. (octubre, 2020). Chapter 3 – Regression. Machine Learning I. Universidad Pontificia de Comillas

[16] Sánchez, E. (enero 2021). Ensemble Learning. Machine Learning II. Universidad Pontificia de Comillas

[17] Amat, J. (febrero, 2017). Árboles de decisión, random forest, gradient boosting C5.0. cienciadedatos.net. https://www.cienciadedatos.net/documentos/33_arboles_decision_random_forest_gradient_boosting_c50#%C3%81rboles_de_regresi%C3%B3n

[18] Brownlee, J. (abril 20, 2020). How to develop a Random Forest Ensemble in Python. Machine Learning Mastery. https://machinelearningmastery.com/random-forest-ensemble-in-python/

[19] Kuhn, M. (marzo 27, 2019). 5 – Model Training and tuning. The Caret Package. https://topepo.github.io/caret/model-training-and-tuning.html

[20] Dive Into Deep Learning. 4.1 Multilayer Perceptron. http://d2l.ai/chapter_multilayer-perceptrons/mlp.html

[21] Baumbah, L. (julio 2020). Neural Network – Activation Function. https://morioh.com/p/21b55ba475f9

[22] Stöttner, T. (mayo 16, 2019). Why Data should be Normalized before Training a Neural Network. https://towardsdatascience.com/why-data-should-be-normalized-before-training-a-neural-network-c626b7f66c7d

[23] Heim, R. (diciembre 26, 2019). How to Train a Multilayer Perceptron Neural Network. All About Circuits. https://www.allaboutcircuits.com/technical-articles/how-to-train-a-multilayer-perceptron-neural-network/

[24] Pizarro, J., Portela, J. y Muñoz, A. (febrero 2021). NeuralSens: Sensitivity Analysys of Neural Networks.

[25] JJ (marzo 23, 2016). MAE and RMSE — Which Metric is Better?. https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d

[26] Sab (junio 2, 2019). MAE vs MSE vs RMSE. http://zerospectrum.com/2019/06/02/mae-vs-mse-vs-rmse/

Índice

Índice de figuras

Índice de tablas

1. INTRODUCCIÓN

1.1 OZONE DRIVE

Ozone Drive es una startup que arranca operaciones en 2011, buscando un hueco en el sector de la automoción sostenible mediante un servicio de *carsharing* de vehículos eléctricos a empresas (Modelo de negocio *business to business*). Además de ofrecer este servicio a compañías orientadas a utilizar una flota energéticamente más eficiente, buscaron de la mano de grandes cadenas hoteleras proponer un servicio de alquiler de vehículos eléctricos a sus clientes, instalando puntos de recarga eléctrica en los propios hoteles.

Ilustración 1.1: Servicio de Renting de Vehículos eléctricos en cadena hotelera

En varias ocasiones ha organizado *Road Shows* junto con empresas del sector turístico, ofreciendo la experiencia a turistas, de conducir sus vehículos eléctricos.

En 2014 junto con la colaboración de BMW, implantó uno de los primeros servicios de *carsharing* totalmente eléctrico de Europa, recibiendo en 2015 como premio por la iniciativa, el Sello de Excelencia de Innovación de la Unión Europea.

En 2017 desarrolló su tecnología *UnPlug&Drive*, una tecnología pionera de conectividad para la gestión de flotas de *carsharing* sin necesidad de hardware a bordo.

En 2019 fue elegida como una de las startups tecnológicas más importantes de Europa en el Programa de Aceleración *Impact Connected Car* de la UE, por el desarrollo de herramientas basadas en Inteligencia Artificial, BigData y Analítica Avanzada en el campo de la movilidad eléctrica y eficiencia energética de flotas.

Tras más de 6 años trabajando al lado de grandes empresas, encontraron que el gran hándicap de este servicio de *carsharing* era la descentralización de la carga de los vehículos una vez había finalizado la jornada laboral. Por regla general, los trabajadores se desplazaban desde sus hogares con el vehículo de empresa hasta los diferentes puntos de trabajo y al finalizar realizaban el mismo recorrido a la inversa, haciendo que la instalación de un punto de carga en el domicilio fuera un requisito indispensable.

Las dificultades que encontraron en la instalación de puntos de carga en los domicilios particulares de los empleados hicieron que en más de una ocasión la transición de flota fuera prácticamente inviable.

Pese a las dificultades halladas, el modelo de negocio les permitió conocer de primera mano cómo gestionan las flotas las empresas, descubriendo que de forma generalizada en todas las compañías cada 4 – 5 años se producía una renovación prácticamente completa del parque móvil.

Aunque de esta decisión dependían una buena parte de los costes operativos (hasta 10-15% dependiendo del sector en el que se encontrara la empresa) de las compañías, descubrieron que se tomaba en base a ofertas comerciales, vehículos semejantes y consumos teóricos, sin estudiar la idoneidad de los vehículos para las necesidades particulares de cada uno de los trabajadores.

Ante esto, surgió la idea de desarrollar una aplicación que, basándose en modelos de analítica avanzada y Big Data, pudiera dar soporte a las decisiones relacionadas con la transformación de la flota en aquellas empresas que tuvieran un número relativamente alto de vehículos.

Esta idea fue apoyada por el grupo automovilístico Bergé Auto, quien a través de su *Venture Lab* (*B4Motion*), decide participar en el capital de Ozone Drive para acelerar la aplicación hasta convertirla en un producto viable para la comercialización.

1.2 SWITCHFLEET

SwitchFleet es la aplicación que pretende optimizar la transición de flota, mediante la analítica avanzada y el Big Data, ofreciendo soporte a los gestores de flota en la toma de decisiones.

Mediante esta aplicación, se procedería a estudiar las principales variables que afectan al consumo de los vehículos (independientemente del tipo de tecnología que permita el movimiento). Para que, a través de modelos de predicción calcular el consumo de los conductores con todos los vehículos potencialmente utilizables (presentes en el mercado español, u ofertas de renting proporcionadas por los clientes).

Con los consumos previamente obtenidos, el cálculo de costes es prácticamente inmediato. Bastará con conocer el valor aproximado del precio del carburante correspondiente en la localización (provincia) donde se encuentra el conductor, o el valor del precio de la electricidad, en el caso de los vehículos eléctricos e híbridos enchufables.

Además del cálculo de costes también es importante tener en cuenta ciertas restricciones que el conductor indica a través de un cuestionario. Dado que el objetivo de Ozone Drive es generar una movilidad más sostenible, la gran mayoría de estas preguntas van destinadas a saber si el vehículo eléctrico o híbrido puede presentar problemas en alguno de los casos en concreto. Si no existe ninguna restricción, siempre se propone como primera opción la que genere un menor impacto medioambiental a través de las Emisiones de CO_2 y NO_X.

Algunas de las preguntas del cuestionario son las siguientes:

- Posibilidad de instalación de un cargador en el domicilio personal
- Media aproximada de los km diarios realizados
- Valor máximo aproximado de km diarios realizados anualmente
- Necesidad de aparcamiento en el centro de alguna ciudad
- Coste de aparcamiento en zona regulada mensualmente
- Marcas preferentes (Existen situaciones en las que las compañías solicitan incluir en el análisis de transformación de flota alguna determinada marca, por acuerdos comerciales, necesidad de diferenciar puestos ejecutivos de operativos, imagen corporativa...)

Con toda la información generada hasta este punto, es posible elegir el vehículo que cumpliendo todas las restricciones optimice los costes para las necesidades/requisitos particulares de cada uno de los conductores.

SwitchFleet ha sido diseñada como una aplicación tipo *SAAS* (*Software as a Service*), que busca proponer flotas más eficientes y económicas mediante la parametrización de la conducción, así como las características de sus vehículos actuales y potenciales, a través de técnicas de Machine Learning y una visualización sencilla en la que se explican los ahorros en costes y emisiones de la nueva flota propuesta.

Para los primeros clientes, se decidió desarrollar un archivo Excel totalmente manual en el que se aplicaran todas estas iteraciones. El principal problema se encontraba en la integración de diferente tipología de vehículos, así como la actualización de este al mercado actualizado del automóvil, y la sencillez de los modelos de predicción de consumos. Los problemas se acrecentaban a medida que la flota estudiada y la diversidad de los vehículos aumentaba.

La componente de visualización se encontraba desarrollada mediante la herramienta Power Bi. Pese a estar ya desarrollada, se ha decidido renovarla para lograr que la información que captaba la atención de los gestores de flota quedara mucho mejor reflejada, obteniendo de

un simple vistazo los principales *insights* del informe para las diferentes flotas propuestas. Para la nueva aplicación también se utilizará la herramienta Power Bi.

En las Figuras 1.2 y 1.3 se puede observar la diferencia entre la apariencia de una de las pestañas del primer informe desarrollado, y el actual.

Ilustración 1.2: Primer desarrollo de la aplicación SwitchFleet

Ilustración 1.3: Nuevo desarrollo de la aplicación SwitchFleet

Ante la creciente demanda de proyectos de transición de flota, se decidió construir una aplicación totalmente automatizada que fuera capaz de tratar toda la cadena de valor del dato en procesos independientes de forma modular, pero interconectados entre sí.

Mediante la utilización de analítica avanzada y Big Data, se podrá además de facilitar el proceso de actualización de vehículos, analizar un espectro mucho más amplio de variables, generando por un lado una mejora en la precisión de los resultados, y por el otro, propuestas de flota más eficientes en costes y emisiones.

A comienzos de 2021, Ozone Drive llega a un acuerdo con la empresa Geotab para ser proveedor oficial de sus dispositivos de análisis en real time de múltiples variables.

Mediante esta alianza, Ozone Drive busca mejorar la cantidad y la calidad de los datos en los que basa sus cálculos predictivos, ofreciendo estos dispositivos a los clientes interesados en el producto *SwitchFleet* como una ventaja estratégica clave a la hora de conocer de primera mano cómo conducen sus conductores y de esta forma entender buena parte de los costes operativos.

La medición de los parámetros se realiza mediante un dispositivo de seguimiento que se acopla al vehículo a través del puerto OBD.

Ilustración 1.4: Dispositivos GO9 de Geotab

El proyecto se encuentra englobado en el desarrollo de esta aplicación, abarcando desde la definición de la arquitectura de los módulos, extracción de los datos, su tratado y almacenamiento, modelado mediante algoritmos de Machine Learning y creación de un informe automatizado en el que los clientes puedan ver de forma clara toda la información obtenida en diferentes informes interconectados.

2. ARQUITECTURA Y PIPELINE

La arquitectura del proyecto, definida a alto nivel tendría la forma representada en la Figura 2.1

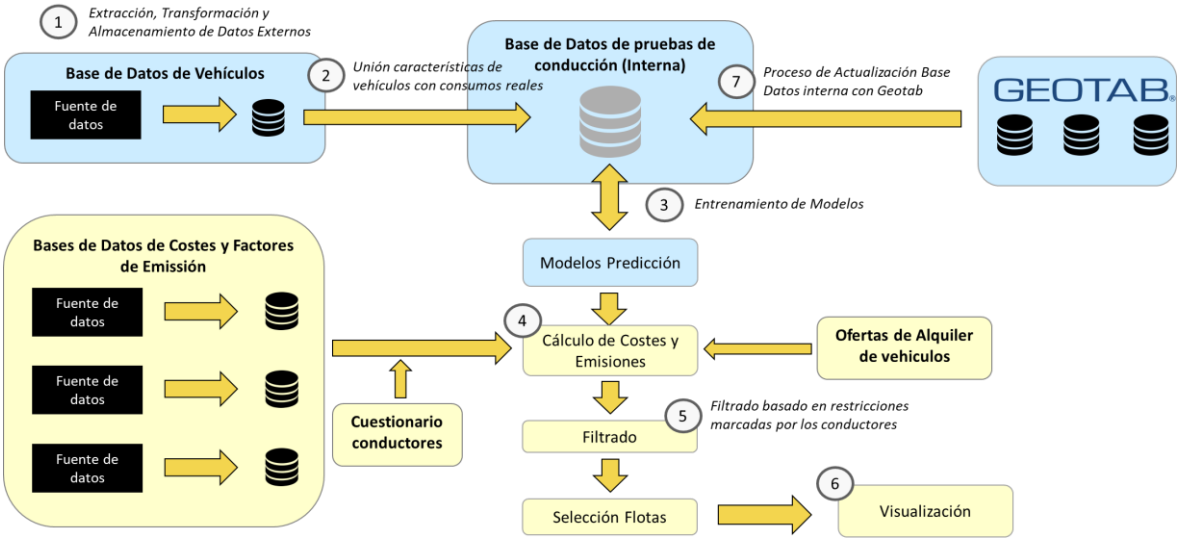

Ilustración 2.1: Esquema Resumen Arquitectura SwitchFleet

Se distinguen 2 tipos de procesos diferenciados por los colores de fondo

- Los procesos azules son aquellos que aportan datos para el entrenamiento de los modelos que posteriormente generarán predicciones de consumo.
- Los procesos amarillos son los que se activan una vez estén diseñados los modelos. Traducen los valores de consumo predichos en costes y emisiones totales

A continuación, se procede a detallar de forma breve cada uno de ellos:

1- En primer lugar, se pueden observar los procesos de extracción, almacenamiento y transformación de los datos externos a Ozone Drive. Mediante estos procesos se extraerán bases de datos como la de precios de combustibles, características de vehículos o factores de emisión.

2- El segundo paso será unir la base de datos de características de vehículos (representada en azul) extraída en el paso previo con la base de datos interna de Ozone Drive que contiene consumos reales obtenidos en pruebas de conducción en diferentes entornos y con diferentes estilos. Esta unión se realiza con el motivo de generar una base de datos de entrenamiento con variables inputs (que provienen de la de Características de Vehículos) y outputs (que provienen de la Base de Datos interna de Ozone Drive)

3- Una vez se encuentre la base de datos interna con los consumos obtenidos en pruebas de conducción enriquecida con la de características, se tiene el conjunto de datos que sirva como entrenamiento para los modelos de predicción. Por lo que se entrenarán diversos modelos para así poder comparar entre las métricas que arroje cada uno.

4- Con los modelos entrenados, se realizan las predicciones de consumo. Éstas se transforman en costes y emisiones mediante los datos extraídos en el paso 1, y el Formulario de Conductores. Además de los costes generados por los consumos, habrá que incluir aquellos provenientes del alquiler de la flota de vehículos.

5- Posteriormente llega la fase de filtrado, en la que se recoge toda la información proveniente de los cuestionarios enviados a los conductores, y se aplican las restricciones necesarias caso a caso. Con los vehículos que cumplen los requisitos, se procede a hacer una ordenación (de menor a mayor) de los costes y emisiones. Tras esto, se seleccionan los primeros n (hasta ahora n está definida por los requisitos del cliente, se espera que más adelante exista un número fijo para facilitar la visualización).

6- Por último, se crea la visualización del *dataset* generado con el filtrado. En ésta, debe quedar clara la desviación (negativa si se consigue reducir, positiva en caso contrario) en los costes y emisiones. Para facilitar la comprensión de lo que supone el ahorro o aumento en las emisiones de la flota propuesta, existe una tabla soporte que ayuda a calcular el equivalente del delta de emisiones en número de árboles plantados (o eliminados), y hectáreas de bosques.

7- En próximos desarrollos se introducirán datos de los dispositivos de Geotab en la base de datos interna de Ozone Drive con consumos.

Siguiendo con el último punto previamente señalado, se espera que, para próximas implementaciones, los dispositivos Geotab se encuentren instalados en múltiples vehículos proporcionando datos a la base de datos interna de Ozone Drive que hagan mejorar la precisión en los algoritmos. Estos datos se obtendrán por medio de un servicio API REST que pone a disposición Geotab para todos sus clientes.

El proceso de extracción de datos debería ser ejecutado al menos una vez al día, y los modelos recalibrados con los nuevos datos al menos una vez mensual.

Es importante entender que, pese a que el *pipeline* del proyecto se está explicando de forma lineal, al tratarse de un proceso analítico que soporta una aplicación comercial, debe verse como un proceso cíclico. Es decir, se generarán nuevas bases de datos cada cierto intervalo de tiempo en función de las necesidades específicas de la demanda de proyectos.

Toda esta estructura modular se explicará al detalle a lo largo del presente proyecto, en los diferentes apartados que se encuentran a continuación.

3. OBTENCIÓN Y PROCESAMIENTO DE DATOS

A lo largo del presente capítulo se procede a describir las diferentes fuentes y procedimientos seguidos para obtener, los datos que se utilizarán en el proyecto. Posteriormente, se detallarán los pasos realizados para procesamiento, limpieza y ordenado de los datos, y su posterior almacenamiento.

Dado que hasta ahora la decisión de la transición de flota se realizaba en base a pocos datos sin analizar de forma profunda el valor de estos, tener múltiples fuentes enriquecerá los informes entregados al cliente, pudiendo dar respuesta a numerosas cuestiones difícilmente resueltas previamente.

También es importante además de la cantidad de fuentes de datos, la calidad de estas. Realizar una buena selección en las fuentes, permitirá a la aplicación aportar información veraz dotándola de un rigor analítico superior.

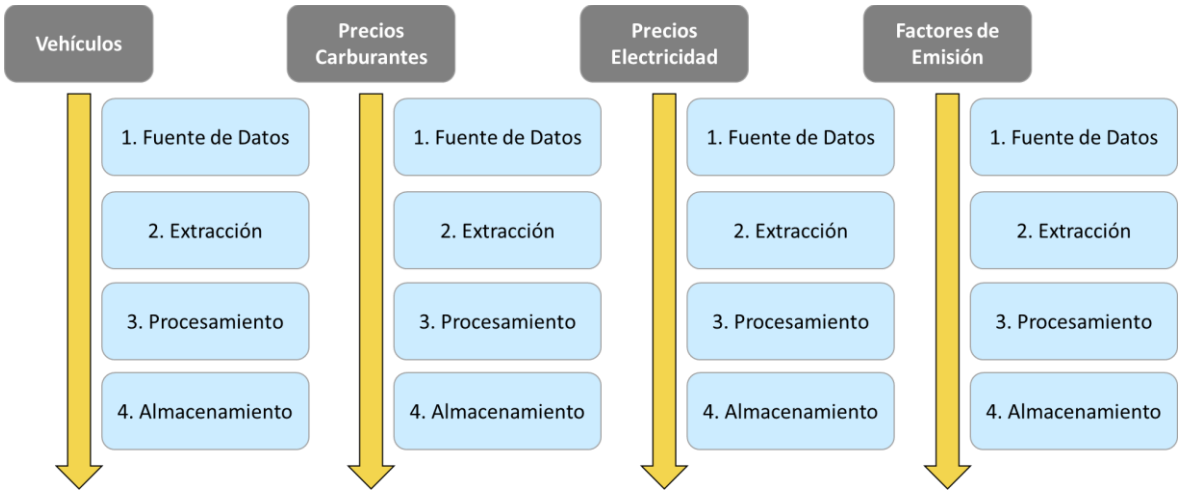

Ilustración 3.1: Proceso ETL para cada fuente de datos

Las principales características que deben de cumplir las fuentes de datos seleccionadas son:

- Información actualizada
- Robustez y veracidad de los datos
- No más de un 40% de valores nulos o huecos en blanco
- Granularidad fina de los datos para poder posteriormente agregar si se considera necesario
- *Primary keys* iguales/semejantes para permitir uniones entre bases de datos y así enriquecer los análisis (posteriormente se resolverá un caso de *join* con diferentes *primary keys*)

Uno de los principales problemas hallados en este apartado ha sido la elección de la fuente de datos que proporcione un número relativamente elevado y fiable de características para cada uno de los vehículos. A pesar de que el mundo del motor es una industria relativamente abierta para un buen porcentaje de la población, existen pocas bases de datos disponibles en las que aparezcan más variables que las homologadas que aparecerían en cualquier catálogo de concesionario.

A continuación, se pretende desarrollar en apartados separados, los procesos ETL de creación de cada una de las bases de datos. El procedimiento para la descripción de las diferentes bases de datos utilizadas se encuentra detallado en el esquema de la Figura 3.1. Se comienza partiendo de la descripción de la fuente de datos, analizando las principales características. Posteriormente se procede a detallar los procesos de Extracción y Transformación de los datos y su almacenamiento final.

3.1 CARACTERÍSTICAS DE VEHÍCULOS

Contar con una base de datos de vehículos con un elevado número de variables (características) permitirá desgranar mejor el mecanismo de funcionamiento de las diferentes categorías de estos (combustión, híbridos y eléctricos), y por lo tanto ofrecer predicciones más precisas en los consumos.

3.1.1 FUENTES DE DATOS

Como se comentaba en la introducción a este mismo apartado, antes de seleccionar una fuente de datos válida se realizó una labor de investigación para asegurar que éstas presentaban más variables que las homologadas que suelen aparecer en cualquier fuente de información del motor (velocidad máxima, aceleración de 0 a 100 km/h máxima, potencia máxima…)

Para este caso, fueron dos fuentes las seleccionadas por la gran cantidad de variables potencialmente extraíbles:

Tabla 3.1: Fuentes de datos para Extracción de Características de Vehículos

Nombre Fuente de Datos	*url*
Km77	https://www.km77.com/
ElPeriódico - Motor	https://www.elperiodico.com/es/motor/

Prestaciones y consumos homologados	
Velocidad máxima	171 km/h
Aceleración 0-100 km/h	10,8 s
Consumo WLTP	
Combinado batería cargada	1,3 l/100 km
Autonomía eléctrica WLTP	50 km
Emisiones de CO_2 WLTP	29 gr/km
Normativa de emisiones	Euro 6
Distintivo ambiental DGT	0 emisiones
Dimensiones, peso, capacidades	
Tipo de Carrocería	Turismo familiar
Número de puertas	5
Longitud	4.605 mm
Anchura	1.800 mm
Altura	1.465 mm
Batalla	2.650 mm
Vía delantera	1.565 mm
Vía trasera	1.573 mm
Peso	1.608 kg
Depósito de combustible	

Ilustración 3.2: Extracto de tabla con características de un vehículo en Km77

Carrocería y Dimensiones
Tipo: **Turismo**
Carrocería: **Familiar**
Calificación energética: **A**
Distintivo ambiental: **C**
Número de puertas: **5 p**
Número de Plazas: **5**
Longitud: **4.866 mm**
Alto: **1.460 mm**
Ancho: **1.871 mm**
Batalla: **2.835 mm**
Peso: **1.425 Kg**
Peso Máximo Admitido: **1.980 Kg**
Arrastre con/sin Freno: **1.900 kg / 710 kg**
Capacidad Mínima del Maletero: **565 litros**
Capacidad Máxima del Maletero: **1.632 litros**

Ilustración 3.3: Extracto de tabla con características de un vehículo en ElPeriódico

Los motivos de la elección de estas dos fuentes son los siguientes:

- *Km77* proporciona información contrastada en sus propias pruebas. Realizan siempre pruebas de conducción sobre el mismo circuito, por lo que todas las variables que se seleccionen serán comparables.
- Pese a que *ElPeriódico* no se trata de una fuente de información ligada al mundo de la automoción, en su sección de motor se encuentra una de las más extensas fuentes de datos públicas de España. Este medio de información obtiene estos datos a través de la plataforma *JATO*, plataforma creada en los años 80 como solución al problema de la creciente globalización en la industria automotriz y la escasez de datos

completos y precisos para todos los vehículos existentes. *JATO* ofrece un servicio de acceso a sus datos mediante suscripción anual.

- Ambas ofrecen datos de modelos que no se encuentran hoy en el mercado. Esto es crucial para conocer las características de los vehículos que actualmente se encuentran entre la flota de los clientes (vehículos de 4 o 5 años que pueden haberse descatalogado muy rápido), o para asociar las características a los vehículos utilizados en las pruebas de conducción.
- Información bien estructurada, facilitando el procesado y almacenamiento, además de la imputación de valores no nulos en aquellos huecos que pueda haber en la base de datos.

El objetivo principal de utilizar dos fuentes de datos es teniendo una de las dos como base, poder enriquecer esta con la información de la segunda. Además, puede servir de gran ayuda a la hora de la imputación de valores no nulos en huecos en blanco.

Para poder realizar la operación que permita obtener información de ambas bases de datos es necesario que al menos una de las columnas de ambas tablas sea común. Lo lógico en este caso, es que dicha columna sea la del nombre del vehículo, puesto que es identificador clave cuando se habla de las variables. Además, es la única que, en principio, no debería de tener valores duplicados, facilitando así la unión de ambas.

El problema hallado es la falta de uniformidad existente en la nomenclatura a la hora de nombrar vehículos. Resultando que, el mismo en ambas fuentes de datos, se encuentra con un nombre ligeramente diferente. Como se puede observar en las siguientes imágenes, hay ciertos caracteres que pueden o no aparecer reflejados en el nombre (potencia en KW o CV, Fecha de Lanzamiento…)

Ilustración 3.4: Nombre de un vehículo concreto en las dos fuentes de datos

Posteriormente se tratará en detalle la metodología utilizada para resolver el caso de unión de ambas fuentes de datos con diferentes nombres de vehículos, utilizando esta columna como nexo entre ambas.

En cuanto al nombre completo, para generar una base de datos ordenada por niveles, se ha decidido dividir la cadena de caracteres completa en las siguientes categorías

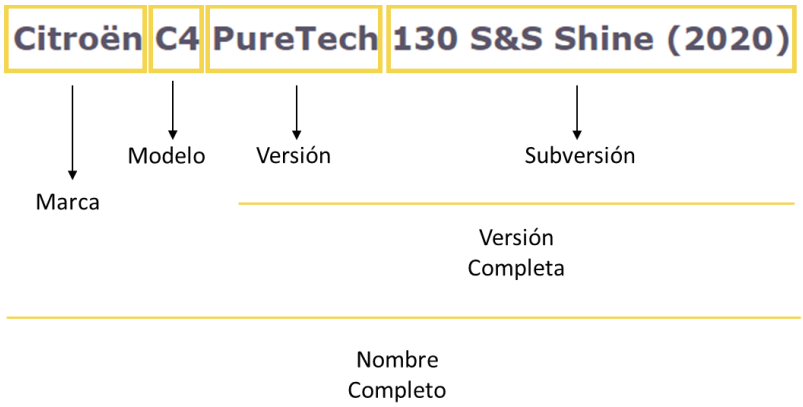

Ilustración 3.5: Descomposición del nombre completo de un vehículo

3.1.2 EXTRACCIÓN

Como ambas fuentes se encuentran en una página web, existen dos métodos principalmente aplicables a este tipo de datos:

- Llamada a *API Request*
- *Web Scraping*

Dado que ninguna de las dos fuentes presenta un servicio de *API REST* la única opción viable para extraer los datos es mediante el *scraping* de las tablas que se encuentran en la página web.

El proceso para obtener los datos es semejante para ambas fuentes de datos. Por lo que el esquema que se muestra a continuación y todo el desarrollo es aplicable a ambos orígenes de datos.

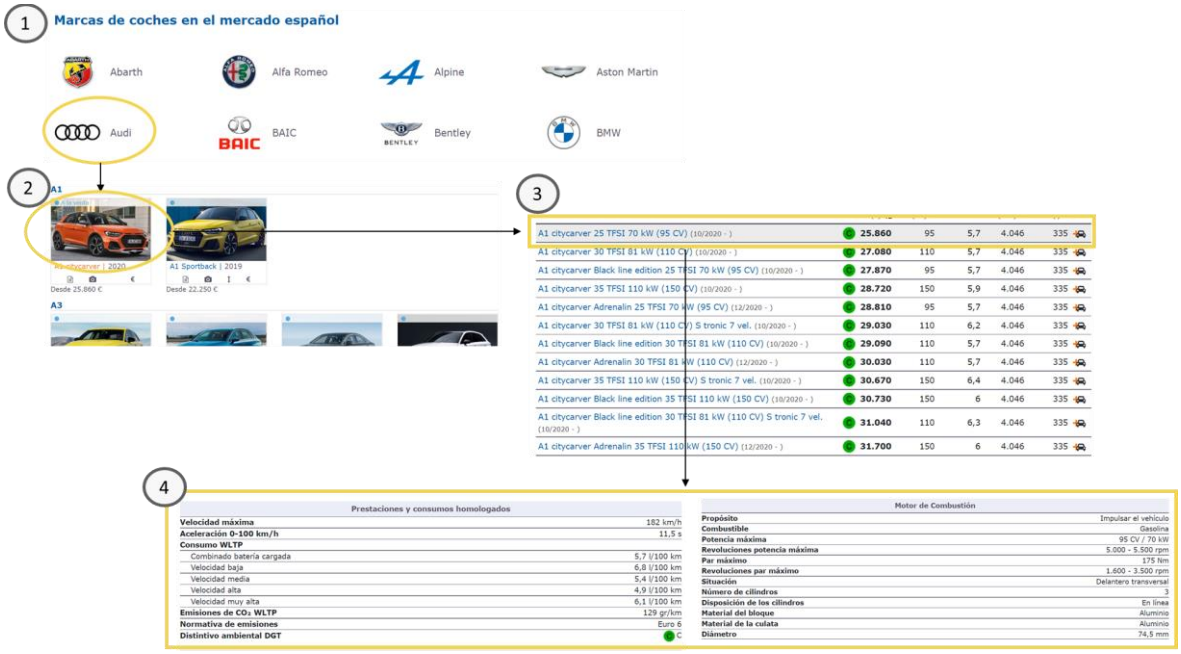

Ilustración 3.6: Esquema representativo del proceso de extracción de datos

1. Iteración por todas las marcas que presentan vehículos en el mercado español actualmente.
2. Una vez dentro de una marca, se almacena y se procede a iterar entre todos los modelos.
3. Al igual que en el paso anterior, se almacena el modelo y se accede a todos los submodelos formados por las versiones y subversiones existentes.
4. El nivel subversión identifica de forma unívoca el vehículo, incluyendo la información particular que se está buscando. Se accede al vehículo concreto, y se extraen las diferentes tablas que contienen la información buscada.

La *url base* del *scraper* es https://www.km77.com/coches/. A través de esta dirección se irán añadiendo los caracteres que forman marca, modelo y versión completa para formar la *url final* que contiene la información deseada.

Ilustración 3.7: Proceso de formación de la url para el scraper de vehículos

Habrá que tener especial precaución con aquellos casos en los que el *Web Scraper* esté recogiendo información y alcance alguna versión que contenta un prototipo y por lo tanto no sea capaz de encontrar las tablas que se buscan. Este caso se resolverá añadiendo excepciones al código que hacen saltar el bucle a la siguiente iteración.

3.1.3 PROCESAMIENTO Y ALMACENAMIENTO

El procesamiento que genera esta base de datos es el más complejo de los 4 que se describirán en los apartados de este Capítulo. Esto es así, por un lado, debido a la no uniformidad en los nombres de los vehículos entre fuentes de datos, y por el otro, a la necesidad de generar una base de datos sin huecos en blanco, para hacer que el entrenamiento de los algoritmos sea lo más eficiente posible. A continuación, se detallarán las técnicas utilizadas para hacer frente a estos dos problemas:

3.1.3.1 Unión de Bases de Datos con matcheo parcial entre columnas

Como se comentaba en la descripción de las fuentes de datos, el utilizar dos orígenes diferentes para obtener una mayor robustez en los valores fuerza a unir ambos mediante una columna. Esta columna debe ser común a ambas fuentes de datos, y además ser unívoca (para evitar errores en la unión). Por lo que la única opción posible es realizar esta unión mediante la columna que presenta el nombre completo del vehículo, es decir, la *Primary Key*.

La falta de un consenso oficial a nivel mundial para designar el nombre de los vehículos hace que encontrar dos formas exactamente iguales de referirse a un mismo automóvil por toda la web sea una misión prácticamente imposible.

Ante esto, se plantearon dos formas de tratar el problema:

- Generar un algoritmo que sea capaz de reconocer aquellos atributos que no son intrínsecos al nombre del modelo en sí. Es decir, este algoritmo buscaría entre todas las características cuáles se encuentran en el nombre del modelo, y las retiraría.

  Por ejemplo, para el caso de la Figura 3.4 en el segundo nombre, detectaría que los 104KW que aparecen en el nombre indican la Potencia Máxima, por lo que debería de retirarlos (al igual ocurriría con los 141 CV). Esto mismo ocurriría con el primer ejemplo y el año 2020 que aparece.

  El problema que puede surgir con este método es la lentitud al ir a buscar para cada nombre qué cadenas de palabras forman parte de él, o son características añadidas.

- Utilizar un matcheador parcial que sea capaz de *tokenizar* las cadenas de caracteres que aparecen en el nombre, y puntúen el match entre dos nombres en función de lo que se parezcan dichos caracteres. Este método se basa en la *Distancia de Levenshtein*, medida que representa cómo de lejos se encuentran dos cadenas de palabras, es decir, mide el número mínimo de veces que hay que editar una palabra para que sea exactamente igual que la otra.

La solución implementada en este caso ha sido la segunda por la rapidez a la hora de obtener resultados y la trazabilidad que aporta para encontrar posibles errores posteriormente.

Existe una buna parte de veces en las que, mediante este método, el problema pueda ser resuelto. Puesto que con pocos cambios podremos obtener el nombre completo de la fuente de datos contraria. Sin embargo, puede ocurrir que dentro de un nombre aparezca una cadena de caracteres y dentro del otro no. Los cambios que habría que hacer en el segundo nombre pueden ser tan elevados que de forma aleatoria exista otro vehículo cuyo nombre tenga una cadena de caracteres más parecida al primero, y por lo tanto haga el match con este tercero, en lugar del segundo vehículo.

Para resolver este problema, se propone utilizar el paquete *FuzzyWuzzy*. Capaz de encontrar similitudes en subcadenas de palabras dentro del nombre completo.

Siguiendo con el ejemplo de la Figura 3.4, se obtendría el siguiente resultado:

Tabla 3.2: Comparativa para Algoritmo de la Distancia de Levenshtein y Librería FuzzyWuzzy

Distancia de Levenshtein	*Librería FuzzyWuzzy*
65.9574	93

La diferencia entre ambas medidas es más que sustancial, la librería *FuzzyWuzzy* asegura en un alto valor de probabilidad de que se trata del mismo vehículo.

Para prevenir posibles errores a la hora de *matchear* los nombres de los vehículos, y dotar a la solución de aún mayor fiabilidad, se ha diseñado la siguiente implementación

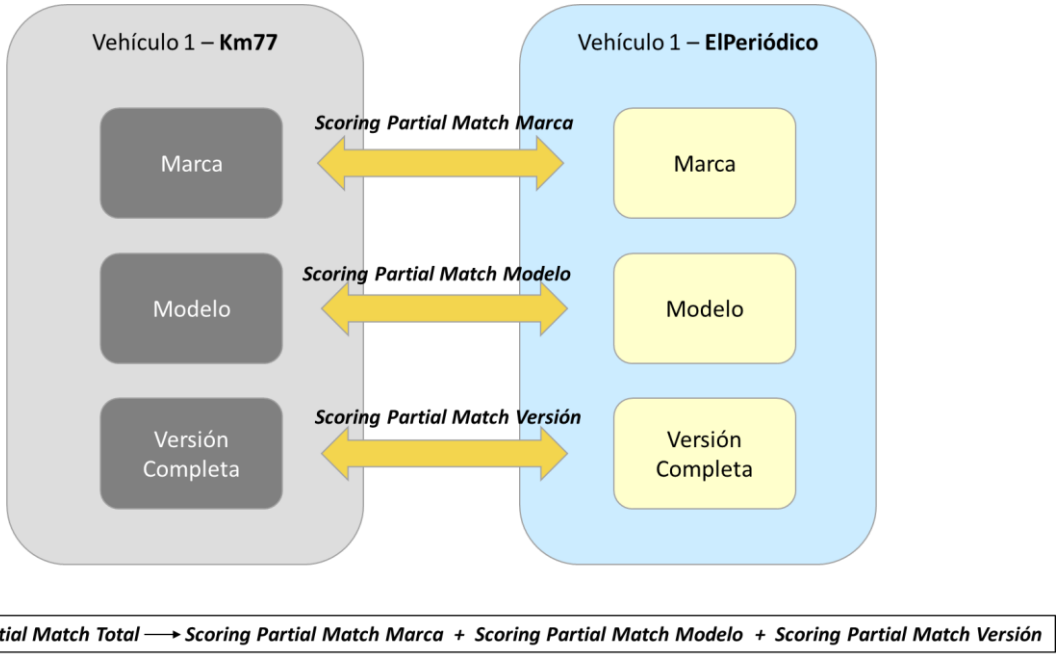

Ilustración 3.8: Esquema representativo del modelo de matching parcial

De esta forma se consigue asegurar que, si el mismo vehículo recibe nombres de versión diferentes para cada una de las fuentes, tendrá un Score de Match muy alto al coincidir prácticamente al 100% tanto para Marca como para Modelo.

Unir ambas fuentes de datos permitirá, como se comentaba previamente enriquecer las variables elegidas con información proveniente de la fuente complementaria. Pese a ello, sigue cabiendo la posibilidad de encontrar valores nulos en alguno de los campos de la base de datos resultante.

3.1.3.2 Separación de los datos por Tipo de Combustible

Si nos fijamos en la Figura 3.9, las clases por Tipo de Combustible se encuentran enormemente desbalanceadas. Suponiendo los Vehículos de Combustible Fósil más del 60% de la muestra total.

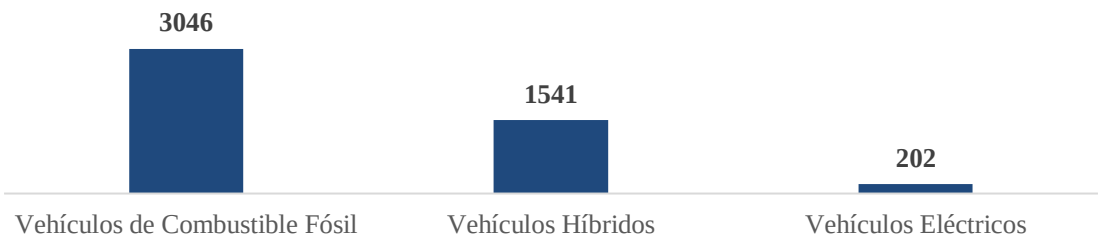

Ilustración 3.9: Número de vehículos por tipo dentro de la muestra

Por la razón anterior, el filtrado de las variables con más de un 60% de valores no nulos, impactará negativamente en los Vehículos Híbridos y Eléctricos eliminando prácticamente todas las puramente asociadas a esta tipología de vehículos.

Si nos vamos al caso de los Vehículos Eléctricos, es fácil de observar. Este tipo de vehículos representan únicamente el 4% de la base de datos, por lo tanto, aquellas variables inherentes a ellos (como son el alcance o el tiempo de recarga de la o las baterías) tendrán como máximo número de observaciones ese 4% (quizás haya casos en que compartan variable con los Híbridos y la densidad de observaciones aumente), por lo que el 96% restante serán huecos en blanco. Quedando totalmente descartada por el filtrado de no más de 40% de valores nulos dentro de una columna.

Por esta razón, se ha considerado útil realizar una división de la base de datos por Tipo de Combustible utilizado. De esta forma, el anterior 4% de las observaciones de variables

propias de Vehículos Eléctricos sobre el total de las observaciones, pasan a ser el 100% sobre el total relativo de los mismos.

Esta división además será útil posteriormente para el entrenamiento de los modelos de predicción, ya que las variables output también son diferentes dependiendo del tipo de vehículo que se esté conduciendo. Para Vehículos de Combustión e Híbridos será *l/km* mientras que en los Eléctricos será *kWh/km.*

3.1.3.3 Imputador de huecos en blanco

Entrenar un modelo con valores en blanco en sus predictores, puede afectar de forma drástica al rendimiento de este. Incluso existen modelos que no son capaces de trabajar con nulos dentro de su base de datos de entrenamiento. Por este motivo, es importante realizar este paso antes de que esta base de datos pase a ser la base de entrenamiento de los modelos.

Al unir las bases de datos *scrapeadas* de ambas fuentes, muchos de los huecos en blanco de que trataremos como base (la de *km77*) se habrán rellenado con información proveniente de la segunda (la de *ElPeriódico*). Pero lo más probable es que sigan existiendo campos no hallados en ninguna de las dos fuentes en variables que consideremos importantes para el entrenamiento posterior de los modelos.

Ante esto, las dos opciones que existen son:

- Eliminar aquellas observaciones que tengan al menos un hueco en blanco en alguna de sus variables. El inconveniente de utilizar este método es que a medida que aumenta la muestra de variables o columnas en la base de datos, más probable será encontrar al menos una observación en blanco, forzando a eliminar demasiados vehículos.
- Crear un imputador que rellene estos campos mediante técnicas que se basen en otras observaciones de la misma u otras variables.

Para solucionar este problema, se ha recurrido a la segunda opción. Existen múltiples técnicas válidas para generar valores que rellenen los campos vacíos. Entre las que destacan:

- **Imputación de valores a través de la media o mediana**. Como su propio nombre indica, esta técnica se basa en utilizar el valor del cálculo de la media o mediana de las variables (sólo para los campos que no son nulos), para rellenar los huecos en el conjunto de datos. Como principales ventajas de esta técnica destacan la sencillez y rapidez con la que se realiza la técnica. Como inconvenientes hay que destacar que sólo actúa con valores de una misma variable, no es utilizable para variables categóricas y no es un método muy preciso.
- **Imputación de valores utilizando los más frecuentes.** De esta técnica destaca lo bien que trabaja con variables categóricas, pero puede introducir sesgos en los datos y no trabaja tomando referencias de otras variables.
- **Imputación de valores utilizando k-NN.** El algoritmo *k-nearest neighbours* (*k-NN*) utiliza la similitud entre variables para predecir cualquier dato. Por lo tanto, esto puede ser muy útil para realizar predicciones sobre valores nulos a través de la búsqueda de los k vecinos más próximos a la observación sin dato. Como ventaja principal es que este algoritmo sí se basa en otras variables, resultando mucho más preciso que los otros dos métodos anteriormente descritos. Como principales inconvenientes se encuentra que es computacionalmente costoso, y puede resultar sensible a la presencia de *Outliers*.
- **Imputación a través de Métodos Estocásticos de Regresión.** Método que trata de predecir los valores nulos mediante la regresión, utilizando variables que se encuentren relacionadas con la que presenta huecos.

Dado que se trata de un modelo que resuelve el problema basándose en el resto de las variables del conjunto de datos sin ser un algoritmo excesivamente complejo, el método utilizado será el de imputación de valores utilizando K-NN.

Pese a que computacionalmente es costoso, al no tratarse de un problema recurrente a resolver de forma diaria, no generaría unos sobrecostes muy abultados.

Para tratar los posibles problemas ante la presencia de *outliers,* se realizará de forma posterior la técnica de *Isolation Forest* (explicada en el Capítulo de Análisis Exploratorio)

3.2 PRECIOS CARBURANTES

A la hora del cálculo de los costes, es fundamental conocer la tendencia que siguen los precios de los diferentes carburantes en una determinada zona geográfica. El proceso a seguir para obtener una base de datos con la que trabajar es semejante a cualquier otro de los 4 que se detallan a lo largo del presente trabajo.

3.2.1 FUENTE DE DATOS

Se estudiaron diversas propuestas como fuentes para obtener los datos de los precios de los carburantes. Entre ellas, múltiples páginas web y aplicaciones que obtenían en tiempo real los precios que se buscaban.

Si se profundiza hasta llegar a la raíz de la fuente de los datos de las diferentes aplicaciones, se observaba que todas estaban conectadas por medio de una API al servicio web que ofrece el Gobierno para la consulta de datos oficiales https://datos.gob.es/es/apidata. En este servicio se puede encontrar un catálogo de conjuntos de datos, entre los que se encuentra el de *Precio de Carburantes en las gasolineras españolas* (REF e04990201).

Esta aplicación, creada en 2013, dispone de los precios actualizados de todos los carburantes existentes en los puntos de repostaje a lo largo y ancho de la geografía española y se actualiza diariamente.

Además de los precios, también tiene información tan valiosa como la dirección en la que se encuentra el punto de repostaje, horario, municipio provincia y comunidad a la que pertenece. Información que se podría almacenar para posibles desarrollos futuros (visualización de puntos de repostaje próximos, densidad de dichos puntos por provincia/comunidad autónoma, estimación de la tendencia en los precios por provincia/comunidad…)

Dado que se trata de una fuente totalmente oficial, actualizada y que cuenta con la información muy estructurada será la elegida para obtener los precios de los carburantes.

3.2.2 EXTRACCIÓN

Dicha fuente ofrece una forma inmediata de extracción de datos a través de un servicio *API REST*. Por lo tanto, se creará una *request* a este servicio con la *url* que se muestra a continuación.

url ⟶ https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/EstacionesTerrestres/

Ilustración 3.10: Url base para scraper de precios de carburantes

Siguiendo con la periodicidad de la actualización de los datos, la extracción también se realizará de forma diaria.

También es posible extraer históricos filtrando por provincias, que se encuentran a su vez asociadas a un ID único. De esta forma, se podría extraer el listado de provincias con su identificador, e iterar a lo largo de este archivo para obtener el precio de cualquier día que se encuentre almacenado en la base de datos de esta fuente.

A continuación, se muestra un extracto del archivo *json* que genera la *request* con la *url* mostrada en la Figura 3.11


```
244363          "C.P.": "30107",
244364          "Dirección": "AVENIDA JERONIMOS  EN  GUADALUPE, 160",
244365          "Horario": "L-D: 07:00-22:00",
244366          "Latitud": "37,995639",
244367          "Localidad": "GUADALUPE DE MACIASCOQUE",
244368          "Longitud (WGS84)": "-1,176611",
244369          "Margen": "D",
244370          "Municipio": "Murcia",
244371          "Precio Biodiesel": "",
244372          "Precio Bioetanol": "",
244373          "Precio Gas Natural Comprimido": "",
244374          "Precio Gas Natural Licuado": "",
244375          "Precio Gases licuados del petróleo": "",
244376          "Precio Gasoleo A": "1,259",
244377          "Precio Gasoleo B": "",
244378          "Precio Gasoleo Premium": "1,309",
244379          "Precio Gasolina 95 E10": "",
244380          "Precio Gasolina 95 E5": "1,399",
244381          "Precio Gasolina 95 E5 Premium": "",
244382          "Precio Gasolina 98 E10": "",
244383          "Precio Gasolina 98 E5": "",
244384          "Precio Hidrogeno": "",
```


Ilustración 3.11: Extracto json de precios de combustible extraído mediante el scraper

3.2.3 PROCESAMIENTO Y ALMACENAMIENTO

Debido a que esta aplicación busca optimizar los costes de flota para la empresa durante los próximos 4 – 5 años, es difícil conocer con exactitud qué puntos de repostaje utilizará el conductor cuando sea necesario. Por lo que, almacenar del orden de 12000 registros (número de gasolineras en España) de forma diaria con los precios actualizados, requeriría de un dimensionamiento del almacenamiento muy superior al que realmente es necesario.

La solución propuesta será utilizar una granularidad más gruesa que la proporcionada por los datos de origen, agregando por el ID de la Provincia en la que se encuentra localizado el punto de repostaje. Los valores de los precios se agregarán por la media aritmética de todos ellos.

Debido a esto, habrá campos que se perderán al estar asociados a registros únicos, como son por ejemplo el Código Postal, la Dirección o el horario de apertura.

Para generar campos únicos en la base de datos creada, se generará una columna que funcione como clave primaria en la base de datos. Esta columna se formará con los campos ID de la Provincia y Fecha de actualización de los datos

3.3 PRECIOS ELECTRICIDAD

Al igual que ocurría en el punto anterior, el precio de la electricidad es fundamental para calcular la componente de costes formada por el consumo de aquellos vehículos eléctricos e híbridos enchufables.

Los usuarios pueden elegir entre contratar la tarifa eléctrica en el mercado libre, a través de alguna oferta con una Comercializadora, o regulado (PVPC – Precio Voluntario al Pequeño Consumidor). Puesto que el mercado libre depende de la Comercializadora contratada por la empresa (o trabajador que recarga el vehículo eléctrico en su domicilio particular), y de la posible oferta que haya acordado con el consumidor, para el cálculo de los costes de la electricidad se tomará el del mercado regulado.

La tarifa PVPC se caracteriza por ofrecer un precio del Kilowatio por hora (kWh) para cada hora del día (24 precios) fijando este precio en función de la oferta y la demanda existente en el mercado eléctrico. Los precios de esta tarifa pueden conocerse el día anterior.

Hasta el 1 de junio de 2021, el PVPC ofrecía 3 tarifas diferenciadas:

- Tarifa por defecto, 2.0 A – Ofrece un único período de facturación
- Tarifa eficiencia en dos períodos, 2.0 DHA – Ofrece 2 períodos de facturación para el término de energía. Presentando un período punta y un período valle.
- Tarifa para vehículos eléctricos, 2.0 DHS – Limitada a potencias contratadas superiores a 10 kW. Ofrece 3 períodos de facturación. Período punta, período valle y período supervalle (orientado a cargar el vehículo eléctrico por las noches, durante el período supervalle)

Será esta última tarifa la elegida para el cálculo de los costes de recarga de los vehículos de la flota.

Desde el día 1 de junio de 2021, el mercado regulado ofrecerá una única tarifa PVPV 2.0 TD que presenta 3 períodos (valle, llano y punta)

Dado que el proyecto ha sido realizado de forma previa al cambio de tarifas, el desarrollo de los siguientes apartados se realizará teniendo en cuenta que existen 3 tarifas diferentes dentro del PVPC.

3.3.1 FUENTE DE DATOS

La fuente de datos oficial que ofrece estos precios es Red Eléctrica de España, a través de su página web (https://www.esios.ree.es/es) ofrece un servicio *API REST* a través del cual se pueden acceder a los precios diarios. Previamente, se requiere autorización a través de token que facilitan una vez solicitado.

3.3.2 EXTRACCIÓN

Como la fuente ofrece el servicio *API REST*, los datos se extraerán mediante una *request*, cuya *url* estará formada por la *url base*, más los parámetros que indiquen el código de la tarifa que se quiere obtener, el intervalo de tiempo deseado y el identificador geográfico (en España 3)

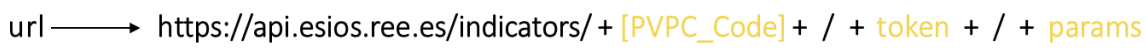

Ilustración 3.12: Proceso de creación de la url para el scraper de precios de electricidad

Una vez extraídas las 3 tarifas se obtiene un archivo *json* con un precio por tarifa para cada hora del día del intervalo que se ha seleccionado. A continuación, se puede observar un extracto del archivo *json* generado para la Tarifa para vehículos eléctricos, 2.0 DHS del PVPC.

Ilustración 3.13: Extracto del json de precios de electricidad extraído mediante el scraper

3.3.3 PROCESAMIENTO Y ALMACENAMIENTO

Debido a que la aplicación busca optimizar los costes de flota para el cliente que lo solicite, para los próximos 4 – 5 años, es difícil estimar los precios de la electricidad con exactitud.

Por esta razón, almacenar el precio horario de cada día conllevará sobredimensionar el almacenamiento sin un objetivo justificado.

Por lo que, utilizando la columna de hora y fecha de cada uno de los precios como clave primaria, las 3 tarifas se unirán en un mismo archivo. Una vez unidas, para reducir la granularidad de los datos, se ha propuesto agregarlos de forma mensual a través de la media aritmética de los precios.

De esta forma, pese a que se perderá algo de precisión en las estimaciones en costes de recarga (puesto que obviamos que los vehículos muy probablemente se recarguen durante la noche, con *Tarifa Supervalle*), se reducirá en un 96,67% el espacio de almacenamiento que ocupará la base de datos (en lugar de tener 720 registros mensuales, sólo habrá 24)

Para que las observaciones en cada uno de los precios sean únicas una vez agregadas, se ha generado el campo clave primaria formado por el valor del mes y año que agrega la media de los precios.

3.4 FACTORES DE EMISIÓN

Para el cálculo de las emisiones anuales totales de la flota, es imprescindible el cálculo de los factores de emisión. De forma anual, el Ministerio para la Transición Ecológica y el Reto Demográfico publica dichos factores asociados a la generación de energía eléctrica y proveniente de combustibles fósiles.

La solución propuesta para generar esta base de datos es un *scraper* automatizado que descargue anualmente el reporte creado por el Ministerio, y convierta el PDF en un archivo CSV, con el que se pueda trabajar de forma posterior.

Debido a la estructura no estructurada de los datos, ha sido necesario el desarrollo de un *script* que fuera capaz de identificar columnas útiles y necesarias de las que no lo fueran.

Estos valores extraídos se utilizarán posteriormente para conocer las emisiones ligadas al consumo (independientemente del tipo de tecnología motriz del vehículo) que se calculará a través de los modelos de predicción.

3.5 CONSUMO EN PRUEBAS DE CONDUCCIÓN

Dado que esta base de datos se encuentra en los servidores de Ozone Drive, la estructura del punto será ligeramente diferente. No habrá que extraer, procesar o almacenar la información.

Ozone Drive cuenta con pruebas en más de 100 y 30 vehículos no eléctricos (de combustión e híbridos) y eléctricos respectivamente. Estas pruebas se han agregado parametrizando el estilo de conducción y el tipo de trayecto en 3 y 3 niveles respectivamente. De esta forma se facilita el que la muestra sea comparable.

Por lo tanto, para cada vehículo se encontrarán 9 registros de consumos diferentes como se comprobar a continuación:

Tabla 3.3: Valores de consumo para un vehículo extraídos en pruebas de conducción

Consumo Real (l/km) - Audi A3 Sportback 30 TDI 85 kW (116 CV) (2020)	*Entorno*	*Estilo*
0,054184832	Ciudad	Tranquilo
0,035557951	Carretera	Tranquilo
0,0338452	Autovía	Tranquilo
0,054223117	Ciudad	Normal
0,042660899	Carretera	Normal
0,046514289	Autovía	Normal
0,057884996	Ciudad	Deportivo
0,045583148	Carretera	Deportivo
0,056011224	Autovía	Deportivo

Es en este punto donde se demuestra la importancia de utilizar fuentes de datos para las características de los vehículos que incluyeran vehículos ya descatalogados.

Para entender estos consumos en pruebas reales, es necesario tener una base de datos robusta con las principales características de los diferentes vehículos. Dado que Ozone Drive no contaba con ello, fue necesario unir esta base de datos de pruebas (algunas con vehículos algo más antiguos) con la base de datos que proporciona el valor de las diferentes variables para cada vehículo.

Como se comenta, alguno de estos vehículos se encuentra ya descatalogado, por lo tanto, si la fuente de información elegida para proporcionar las características de estos no incluyera históricos, sería necesario eliminar dichos registros pues no habría variables con las que entrenar el algoritmo de predicción.

Esta base de datos estará a su vez dividida en 2, en función de la columna de salida. Para aquellos vehículos de combustión e híbridos, la columna *output* será *l/km*, mientras que para los vehículos eléctricos será *kWh/km.*

3.5.1 FUENTE DE DATOS

Los datos de las pruebas de conducción se obtienen o bien a través de un dispositivo conectado al puerto *OBD* (funcionamiento semejante al dispositivo de seguimiento de Geotab, ver Figura 1.4), o a través de una alianza con la compañía *BotOn*. Esta empresa posee una plataforma que ofrece estadísticas sobre diferentes recorridos y tipología de vehículos, por lo que es una fuente fiable de información útil a la hora de construir la base de datos.

Conocer la orografía sobre la que se está realizando el recorrido es fundamental para entender posibles desviaciones en el consumo. Pero los datos que aportan ambas fuentes no contienen la altura sobre el nivel del mar de cada uno de los puntos. Para ello, se desarrolló un *script* capaz de asociar la longitud y latitud de cada coordenada geográfica a la altura sobre el nivel del mar que les corresponde.

Finalmente, para asegurar que todos los valores de consumos son comparables y evitar así que existan desviaciones por la diferente orografía en cada uno de los recorridos, se ajustan a un recorrido estándar previamente definido.

4. ANÁLISIS EXPLORATORIO

El paso natural previo a la creación de los algoritmos de Machine Learning, es estudiar mediante una exploración descriptiva, el conjunto de datos. A través de este proceso se pretende entender mejor qué información contiene cada variable, así como calidad y forma de los datos.

Además, resulta muy útil para decidir qué variables trabajarán como predictores dentro del algoritmo de Machine Learning que se creará posteriormente.

La base de datos sobre la que interesa realizar este análisis exploratorio es la generada a través de la unión de la de Características de Vehículos (detallada en el Apartado 3.1) y la de Consumos en Pruebas de Conducción de Ozone Drive (detallada en el Apartado 3.5)

Como se describió en el Apartado 3.1.3.2 de procesamiento de la Base de Datos de Vehículos, el último paso realizado antes del almacenamiento es dividir los datos en 4 conjuntos diferenciados por el tipo de combustible utilizado por el vehículo. Esto se realizaba así para evitar eliminar columnas útiles para tecnologías minoritarias (eléctricos e híbridos especialmente) que no pasaran el filtrado de densidad de valores nulos.

A lo largo del presente capítulo se escogerá la que tiene por Tipo de Combustible Gasolina, pero todos los pasos son aplicables a los 3 restantes.

4.1 RESUMEN DE LOS DATOS

A continuación, se muestra una tabla resumen con las dimensiones del *dataset* cargado (observaciones 486 y variables input 52 más la variable output), así como el número de columnas numéricas y caracteres.

```
-- Data Summary ------------------------
                           Values
Name                       fdata
Number of rows             486
Number of columns          53
_____________________
Column type frequency:
  character                29
  numeric                  24
_____________________
Group variables            None
```

Ilustración 4.1: Tabla resumen de los datos utilizados

Como se puede observar en la siguiente Figura, no existen valores nulos en la base de datos. Esto significa que el imputador de huecos en blanco ha funcionado correctamente.

```
-- Variable type: numeric ---------------------------------------------------------------------------------------
# A tibble: 24 x 11
   skim_variable                                                                n_missing complete_rate
 * <chr>                                                                            <int>         <dbl>
 1 (l/km)                                                                               0             1
 2 (kWh/km)                                                                             0             1
 3 Potencia Máxima Combinada (CV)                                                       0             1
 4 Value.Prestaciones y consumos homologados - Velocidad máxima (km/h)                  0             1
 5 Value.Prestaciones y consumos homologados - . Combinado (l/100 km - kwh/100km)       0             1
 6 Value.Prestaciones y consumos homologados - Emisiones de CO₂ WLTP (gr/km)            0             1
```

Ilustración 4.2: Extracto de detección de huecos en blanco para algunas de las variables

Importante asegurarse que todas las variables tengan el formato correcto, para evitar así errores en el algoritmo posterior. A continuación, se muestra un extracto del listado de columnas acompañado de la tipología en la que se almacenan

```
 $ Potencia Máxima Combinada (CV)                                          : num [1:486] 140 140 140 140 140 1
40 140 140 140 72 ...
 $ Value.Prestaciones y consumos homologados - Velocidad máxima (km/h)     : num [1:486] 213 213 213 213 213 2
13 213 213 213 160 ...
 $ Value.Prestaciones y consumos homologados - . Combinado (l/100 km - kwh/100km): num [1:486] 5.7 5.7 5.7 5.7 5.7
 5.7 5.7 5.7 5.7 4.9 ...
 $ Value.Prestaciones y consumos homologados - Emisiones de CO2 WLTP (gr/km) : num [1:486] 129 129 129 129 129 1
29 129 129 129 105 ...
 $ Value.Prestaciones y consumos homologados - Distintivo ambiental DGT    : chr [1:486] "C" "C" "C" "C" ...
 $ Tipo Vehículo                                                           : chr [1:486] "Gasolina" "Gasolina"
 "Gasolina" "Gasolina" ...
 $ Value.Dimensiones, peso, capacidades - Tipo de Carrocería               : chr [1:486] "Turismo" "Turismo"
 "Turismo" "Turismo" ...
```

Ilustración 4.3: Extracto del listado de variables y tipología

4.2 ESTUDIO DE LA VARIABLE OUTPUT

Es interesante conocer qué forma presenta la variable de salida de los modelos predictivos, en este caso el consumo de combustible en l/km *(l/km)*.

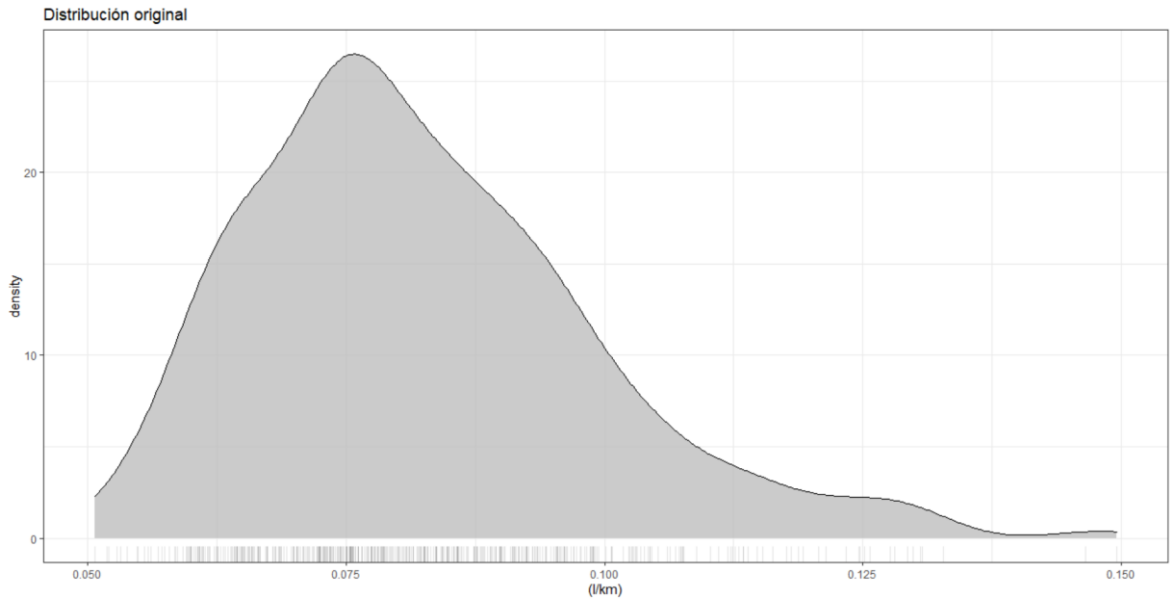

Ilustración 4.4: Distribución de la variable de salida (l/km)

Se trata de una distribución asimétrica con una cola positiva, debido a que existen ciertos vehículos con un consumo muy elevado con respecto a la media. Posteriormente se estudiará si se pudiera tratar de Outliers de medición (no se puede olvidar que los valores de salida se basan en mediciones reales, donde es posible que existan fallos).

4.3 ESTUDIO DE LAS VARIABLES INPUT

4.3.1 VARIABLES CONTINUAS

Este paso resulta interesante para estudiar si existen ciertas variables que pese a ser numéricas, toman valores discretos. Como es el caso de Número de Puertas (1ª columna y 3ª fila) o Número de Plazas (5ª columna y 2ª fila). En estos casos, habrá que tratar a la variable de forma cualitativa en lugar de cuantitativa, cambiando su formato de numérico a factor convirtiéndose estas en variables discretas.

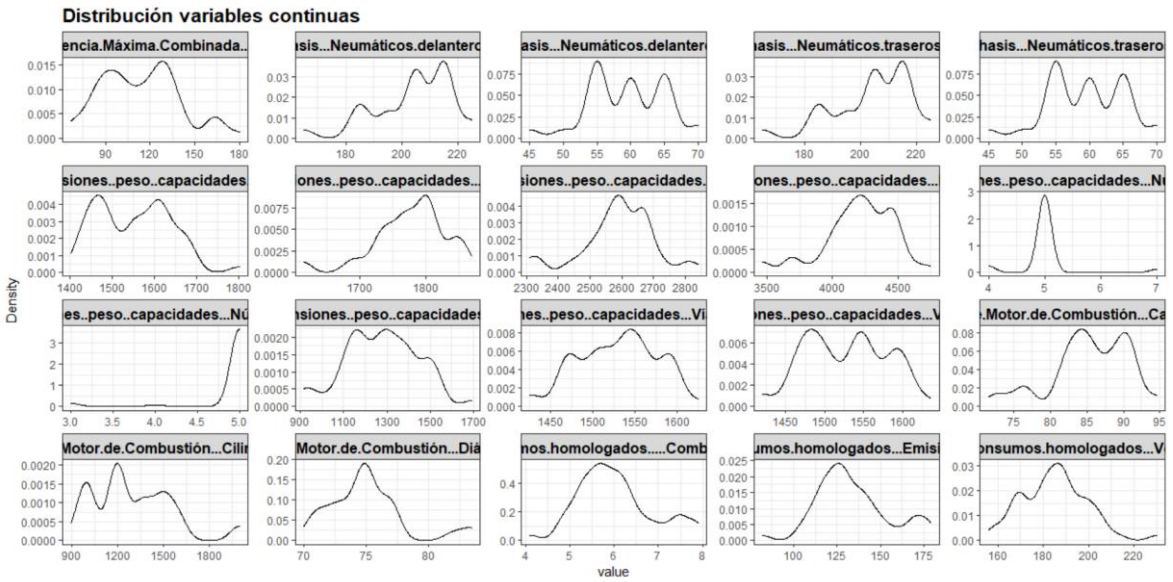

Ilustración 4.5: Distribuciones de las variables continuas de entrada

4.3.1.1 Estudio de la Correlación entre variables

Algunos modelos se ven muy perjudicados ante una correlación muy alta entre predictores. Por lo tanto, detectar correlaciones entre variables es una tarea fundamental previa a la creación del modelo.

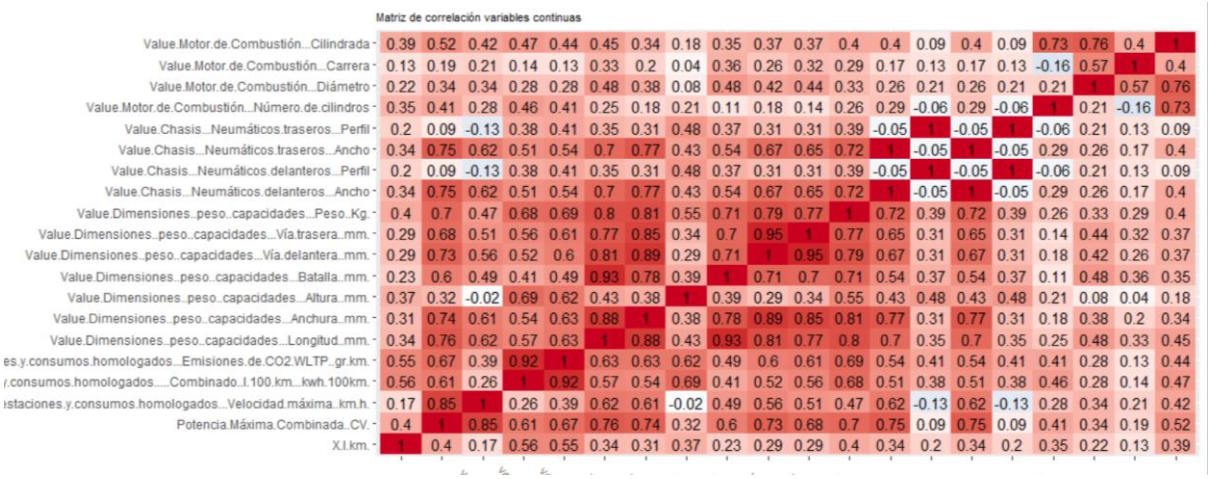

Ilustración 4.6: Matriz de correlaciones entre variables de entrada

Como se puede observar en la Figura 4.6 existen altas correlaciones entre múltiples predictores, pudiendo dar lugar como se comentaba, a irregularidades en el modelo. Aquellas variables que tengan una correlación exactamente 1 se eliminarán del conjunto

4.3.2 VARIABLES DISCRETAS

Realizar el estudio de la información que contienen las variables discretas es fundamental para detectar:

- Variables que contengan varianza nula entre sus observaciones (sólo presentan una clase). Éstas no aportan información al modelo, dando lugar a errores.
- Variables que contengan pocas clases (2 ó 3) y poco balanceadas. Durante la validación cruzada o *bootstrapping* puede ocurrir que algunas particiones tengan varianza nula, dando lugar al caso que se ha descrito en el punto anterior

Para el primer tipo de variables, se procederá a eliminar por completo la columna, puesto que no aporta información útil para diferenciar entre observaciones. Para el segundo tipo, debido a la escasez de datos, eliminar observaciones que formen parte de la clase minoritaria no es una opción válida, por lo que se procederá a agrupar las variables minoritarias en un único grupo.

En la siguiente imagen se puede observar una muestra de las variables entre las que se incluyen los dos tipos previamente expuestos

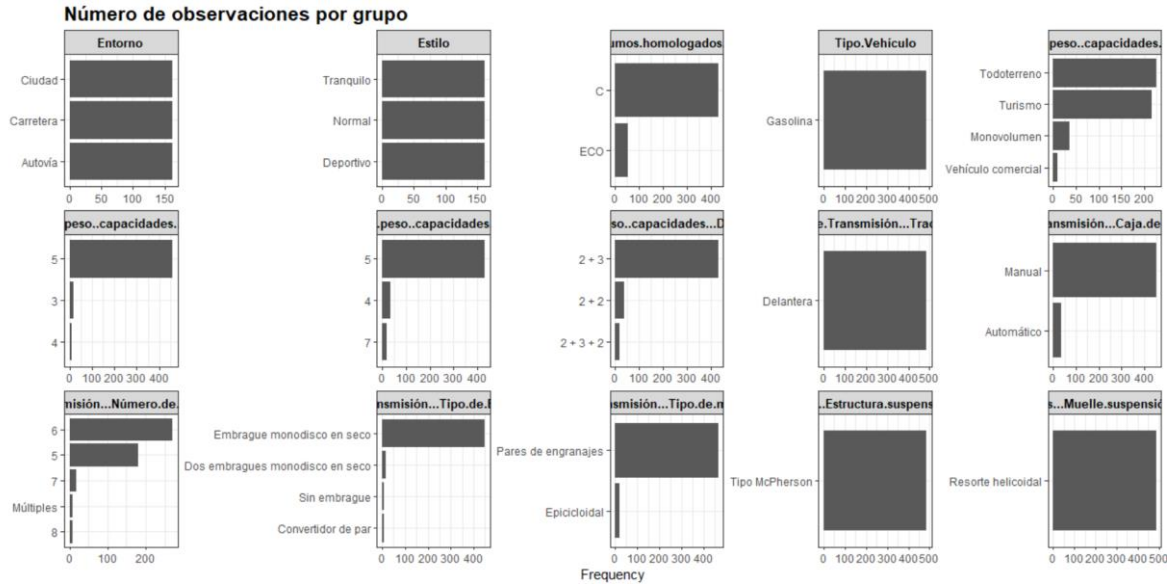

Ilustración 4.7: Distribuciones de variables de entrada categóricas

Puede ser útil, conocer la distribución de los consumos segmentada por alguna de las variables discretas que se consideren más importantes, como lo son el Estilo y el Entorno. Ambas generarán un impacto directo sobre el consumo de combustible.

En las Figura 4.8 y 4.9 ,se puede observar la distribución que siguen los consumos de combustible para los vehículos de Gasolina en función de las variables de Entorno y Estilo respectivamente.

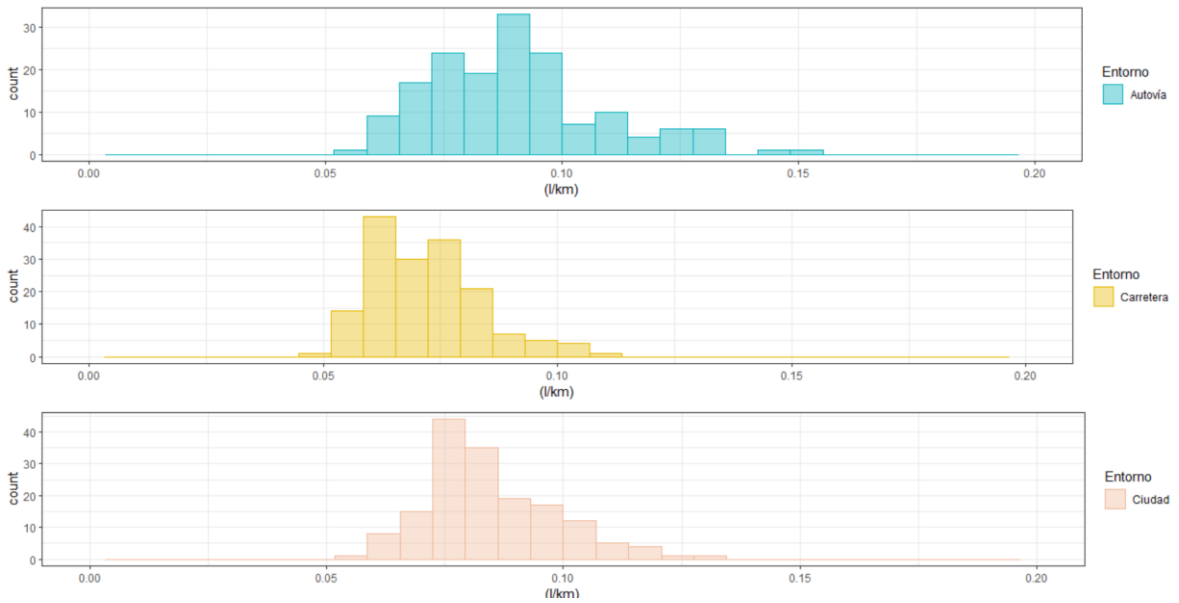

Ilustración 4.8: Histogramas para el consumo en l/km segmentados por Entorno

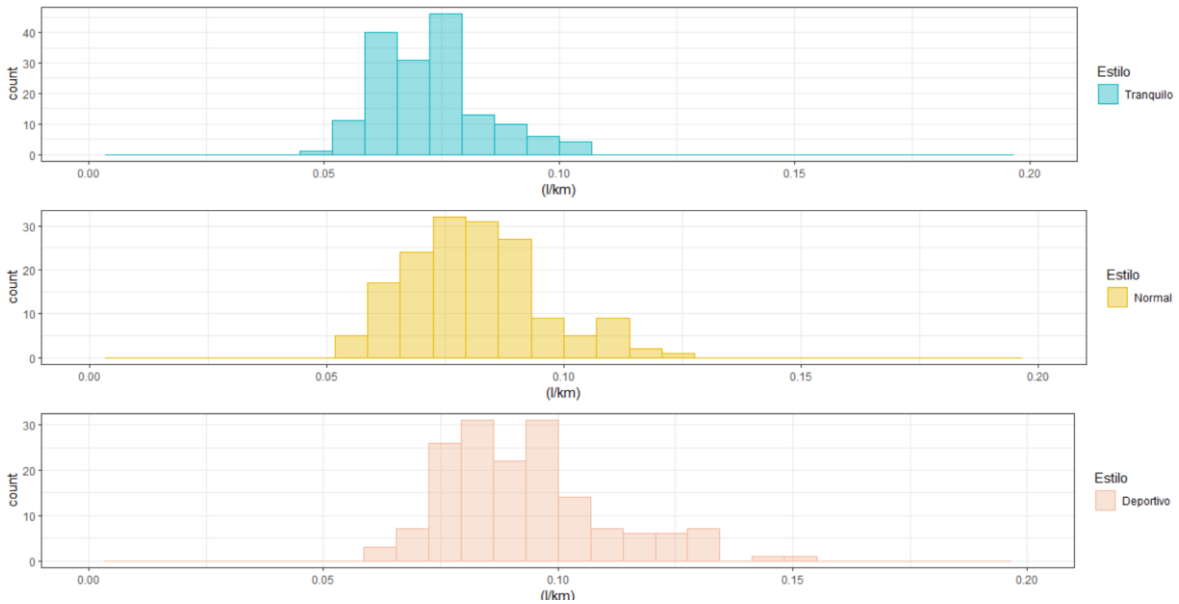

Ilustración 4.9: Histogramas para el consumo en l/km segmentados por Estilo

De ambas figuras se pueden extraer las siguientes observaciones:

- El consumo se ve más afectado por el Estilo de Conducción que por el Entorno (posteriormente esta observación se corroborará). Los histogramas se mantienen muy parecidos independientemente del Entorno en el que se esté conduciendo.

- Los consumos en Carretera presentan una distribución que se encuentra ligeramente más desviada a la izquierda que los de Autovía y Ciudad. Por lo que los consumos por este entorno serán en media, algo menores que en los otros dos.
- En Autovía se encuentran los valores de consumos (cercanos a 15L cada 100km) más alejados de la muestra (por la derecha)
- Si se analizan los histogramas correspondientes al Estilo, se puede apreciar de forma clara que la muestra correspondiente al Estilo Deportivo se encuentra más desviada a la derecha. Esto quiere decir que, en media, los consumos serán mayores con este estilo de conducción. Ocurre todo lo contrario, como era de esperar con el estilo Tranquilo.

4.4 RECURSIVE FEATURE ELIMINATION

Dado que tras realizar prácticamente todo el preprocesamiento aún siguen quedando 38 de las 53 variables con las que se empezó, se utilizarán técnicas de Machine Learning para conseguir medir la importancia de las variables, y así hacer modelos más sencillos que únicamente incluyan estas.

En concreto, se utilizará la técnica de *Recursive Feature Elimination* a través del algoritmo *Random Forest*. Éste genera numerosos árboles de predicción en paralelo, tomando subconjuntos de variables. De esta forma, si existen variables muy correladas, el efecto que generarían en el modelo desaparece. Este algoritmo se explicará de manera profunda en el próximo capítulo (pues es uno de los algoritmos que se construyen para predecir los consumos).

Para crear el algoritmo se utiliza la librería *Caret*, ésta presenta una función llamada *rfe* que automatiza el proceso de selección de las variables más importantes. Como método de validación se utilizará *Repeated Cross-Validation* con 5 repeticiones y el *dataset* subdividido en 10 conjuntos. El código tendrá la siguiente forma:

```
rfe_model <- rfe(x = x_train,
                 y = y_train,
                 sizes = c(1:(ncol(fTR) - 1)),
                 rfeControl = control)
```

Con esto se obtiene que de las 38 variables input iniciales, sólo se necesitan 6 para alcanzar las mejores métricas de *RMSE* y *Rsquared* en la predicción de la variable output. En las Figuras 4.10 y Figura 4.11 se puede observar lo comentado.

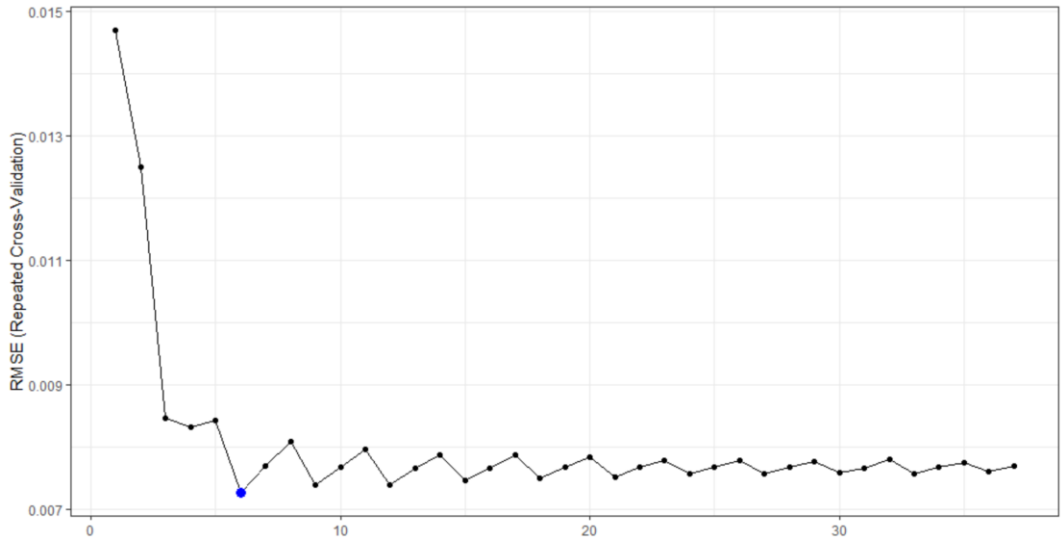

Ilustración 4.10: Valor del RMSE para el número de variables seleccionado

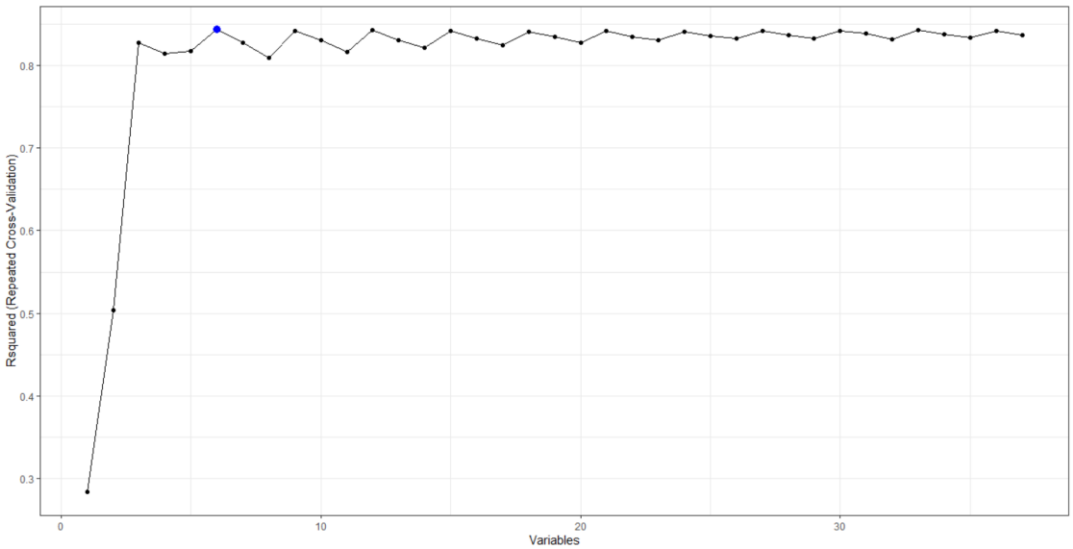

Ilustración 4.11: Valor de R2 para el número de variables seleccionado

A continuación, se muestran qué variables son las que forman el grupo de 6 predictores que optimizan las métricas de predicción, y el valor de la importancia de cada ellas, basado en el porcentaje de varianza que son capaces de explicar.

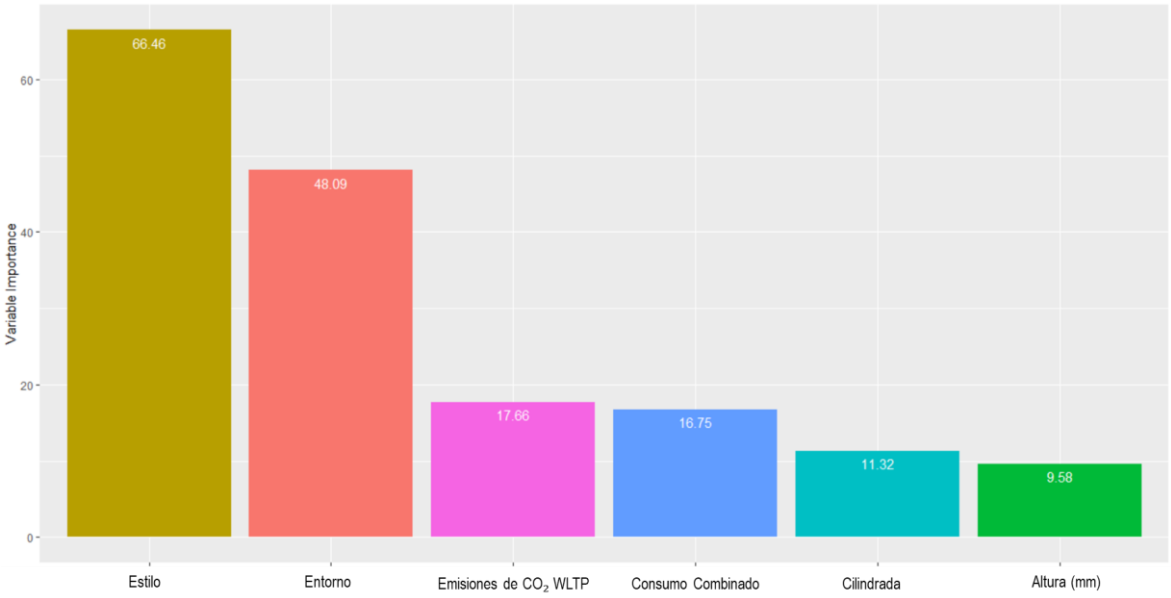

Ilustración 4.12: Variables más importantes y representación de la importancia

Tiene sentido que las variables Estilo y Entorno sean las que más afecten al consumo de combustible, y que las 2 siguientes (Emisiones y Consumo homologados) tengan una importancia semejante, ya que ambas se encuentran directamente relacionadas entre sí.

Por lo tanto, serán estas 6 variables las que entren como predictores en los algoritmos que se creen posteriormente.

Pese a que este paso ha supuesto un coste computacional alto (el algoritmo *Random Forest* tiene que comprobar todas las combinaciones de 1 a 30 variables de forma aleatoria), es clave a la hora de reducir tiempos de entrenamiento posteriores y generar un pipeline más eficiente.

4.5 ISOLATION TREE OUTLIERS DETECTION

Como último paso en el Análisis Exploratorio, se procede a estudiar y tratar los *Outliers* del conjunto de datos que ha quedado. Para ello se utilizará la técnica de *Isolation Forest.*

Esta técnica se basa en el hecho de que las observaciones anómalas son pocas y significantemente diferentes del resto de observaciones "normales". El bosque se construye mediante árboles de decisión, que tienen acceso a un subconjunto de datos. Se basa en la técnica de *Random Forest.*

El proceso selecciona una de las variables que forman el *dataset* y crea una división aleatoria entre el mínimo y el máximo del valor de las observaciones. Como los *Outliers* son muy diferentes al conjunto de la muestra genérica, deberían de quedar aislados en pocas divisiones, por lo que el número de nodos necesarios para llegar a esta observación será menor que para el resto. La diferencia entre la detección de un valor "normal" y "anómalo" se puede observar a continuación.

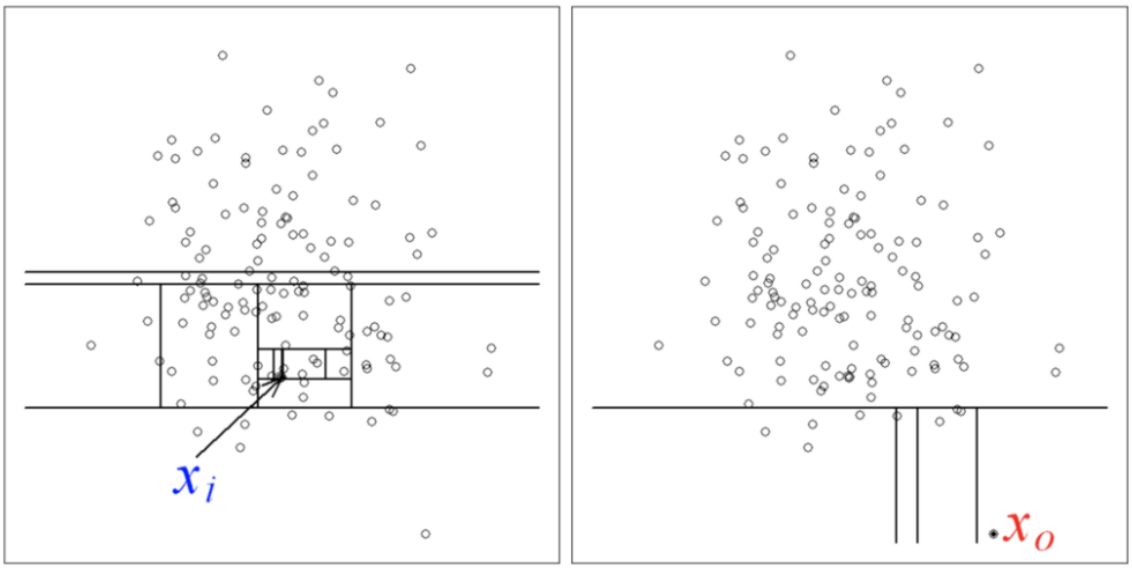

Ilustración 4.13: Proceso de selección de outliers seguido por Isolation Forest

Tal y como sucede con otras técnicas de identificación de *Outliers* se requiere de una puntuación que marque si se trata de un valor anómalo. En el caso del *Isolation Forest* se calcula de la siguiente manera

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

Donde $h(x)$ es la distancia hasta el punto x, $c(n)$ es la media de la distancia de las búsquedas fallidas, y n es el número de nodos.

Dado este valor puede suceder que:

- *Score* cercano a 1 indica anomalía
- *Score* mucho más pequeño que 0,5 indica que se trata de una observación normal
- Si todos los *Scores* son cercanos a 0,5 la muestra entera no presentaría ninguna anomalía distintiva.

A continuación, se puede observar cómo serían los árboles que forman el *Isolation Forest*. Como se comentaba previamente, los valores anómalos se encuentran aislados cerca de la raíz de estos, con puntuaciones superiores a 0,5

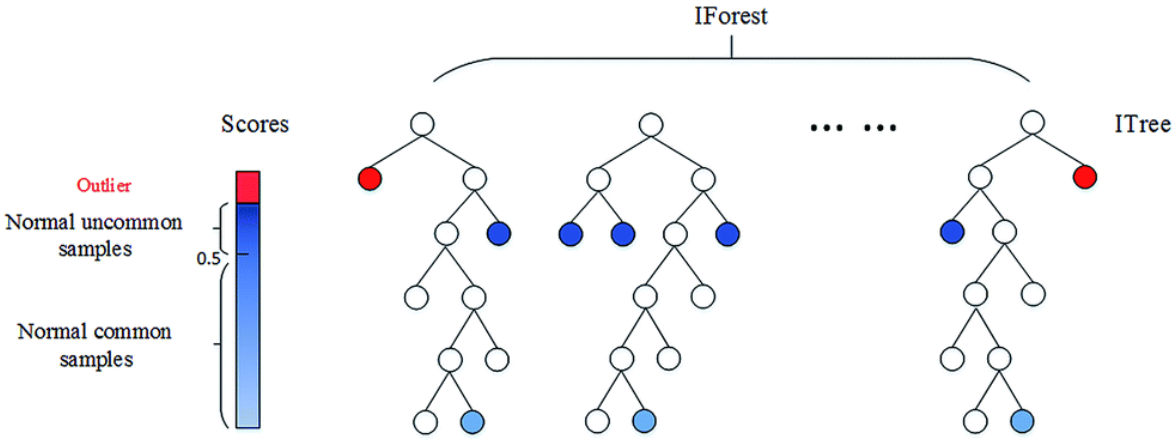

Ilustración 4.14: Representación gráfica del Algoritmo Isolation Forest

Mediante las siguientes líneas de código se implementa el algoritmo de *Isolation Forest.*

```
isoforest <- isolationForest$new(
  sample_size = as.integer(nrow(fdata)/2),
  num_trees   = 500,
  replace     = TRUE,
  seed        = 150)
isoforest$fit(dataset = fdata %>% select(-`(l/km)`))
```

Primero se crea el propio modelo, definiendo el tamaño de la muestra y el número de árboles en el bosque. Y posteriormente se aplica al conjunto de datos sin la variable de salida.

Si se realizan los cálculos del *Anomaly Score* se obtienen que el máximo valor asociado a una observación es 0,6530, muy lejos del 1 que aseguraría que se trata de una anomalía.

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.5808  0.5856  0.5892  0.5973  0.6031  0.6530
```

Ilustración 4.15: Valores del Anomaly Score para todas las variables

Por lo tanto, no se puede asegurar que el *dataset* contenga *Outliers.*

4.6 RESUMEN ANÁLISIS EXPLORATORIO

Ha quedado demostrado que la exploración de los datos previa a la creación de los algoritmos de predicción es un paso clave para entender mejor el conjunto de datos con el que se está trabajando. Invertir tiempo en cada uno de los pasos generará un impacto notable en la mejora de la eficiencia de los modelos.

Se partía de un *dataset* con 53 variables y 486 observaciones, sin huecos en blanco, con 24 columnas numéricas y el restante no numéricas.

Se observó, en el caso de las variables numéricas, muchas de ellas presentaban correlaciones altas, pudiendo posteriormente generar problemas en los modelos. Y en las no numéricas, se detectó algunas cuya varianza era 0 o tenían las clases muy desbalanceadas.

Finalmente, se analizó mediante la técnica de *Recursive Feature Elimination* cuáles eran las que generaban más impacto en las predicciones, eliminando el resto. Esto conllevó una simplificación significativa en el conjunto de datos, pasando de 52 variables explicativas a únicamente 6.

Por último, se analizó la presencia de anomalías en el conjunto de datos, llegando a la conclusión de que no se puede asegurar la presencia de estas entre las observaciones presentes.

5. MODELOS DE PREDICCIÓN

A lo largo del presente capítulo se desarrollará en profundidad cómo se han generado los modelos de predicción de consumos teniendo como base de datos de entrenamiento el conjunto generado previamente.

De nuevo, es importante reseñar que pese que a que se tomará como referencia el *dataset* de vehículos de Gasolina, se procedería exactamente igual con los otros 3 *datasets*.

Dado que los datos de entrada se encuentran segmentados por el tipo de vehículo, a los algoritmos les sucederá lo mismo. Se encontrarán otros 3 procesos de entrenamiento, asociados a cada conjunto de datos que generen algoritmos diferentes.

Generando algoritmos que se adapten a su *dataset* de entrenamiento hará que las predicciones sean mucho más exactas, puesto que los predictores mantendrán su importancia prácticamente constante dentro del conjunto de datos en el que se encuentren.

Para encontrar el mejor modelo a efectos de rendimiento y métricas alcanzadas, se han entrenado 4 diferentes. Del más sencillo al más complejo:

- Árbol de regresión
- Métodos de Ensamblaje (*ensemble*)
    - *Random Forest*
    - *Gradient Boosting*
- Perceptrón multicapa (*MLP*)

A continuación, se procede a realizar una introducción teórica de cada uno de ellos. Esto ayudará a entender de forma más clara los pasos realizados durante el entrenamiento.

5.1 INTRODUCCIÓN TEÓRICA

5.1.1 ÁRBOL DE REGRESIÓN

Se tratan de un subconjunto de modelos dentro de la familia de los árboles predictivos. Aplican cuando la variable de salida es continua.

La estructura del árbol se va generando a medida que las observaciones comienzan a repartirse por los nodos hasta alcanzar el nodo terminal. Las ramas representan cortes o bifurcaciones generadas en uno de los predictores, de tal forma que, a un lado de la bifurcación se encontrarán observaciones que cumplan una condición determinada, y al otro lado las que no.

Las variables más importantes son las que determinan los primeros cortes del árbol.

Como principal ventaja se encuentra que se trata del modelo más sencillo de todos, es fácil de entender, requiere de poca preparación de los datos, las variables pueden ser cuantitativas o cualitativas (como este caso en concreto, con el Estilo y Emisiones, por ejemplo) y funciona muy bien cuando no existen linealidades entre los datos.

La principal desventaja que presentan estos algoritmos es que presentan una varianza muy alta y poco poder predictivo. La alta varianza que presenta puede dar lugar a sobreajuste del modelo sobre los datos de entrenamiento. Si el árbol se deja crecer al final se acabará aprendiendo los datos ajustándose perfectamente a la muestra y generalizando mal.

5.1.2 MÉTODOS DE ENSAMBLAJE (*ENSEMBLE*)

La gran mayoría de modelos existentes en Machine Learning se ven afectados por el problema del equilibrio entre sesgo y varianza (como se ha señalado en el punto anterior, este último era el gran problema que presentaban los árboles de regresión)

El término varianza hace referencia a cuánto se ajusta el modelo a los datos utilizados durante la fase entrenamiento. En el mejor de los casos, un modelo no debería verse notablemente afectado ante mínimas variaciones en los datos de entrenamiento, si esto

ocurriera, es porque el modelo está memorizando los datos en lugar de aprender la relación entre las variables predictoras y la variable respuesta, generando un modelo sobreentrenado (*overfiting*). Por ejemplo, en un árbol de regresión o clasificación muy ramificado (que presenta muchos nodos) suele variar sus ramas y nodos con que apenas cambien unos pocos datos de entrenamiento.

El término sesgo hace referencia a cuánto se alejan las predicciones de un modelo respecto a los valores reales. Por ejemplo, si existe un patrón no lineal entre los variables explicativas y la variable de salida, por muchos datos de los que se disponga, un modelo de regresión lineal no podrá modelar correctamente la relación, por lo que presentará un sesgo alto.

A medida que aumenta la complejidad de un modelo, este dispone de mayor flexibilidad para adaptarse a las observaciones, reduciendo así el sesgo y mejorando su capacidad predictiva. Sin embargo, si se sigue aumentando la complejidad del modelo, se corre el riesgo de alcanzar un determinado grado de flexibilidad, apareciendo el ya mencionado problema de *overfitting*. El modelo se ajusta tanto a los datos de entrenamiento que es incapaz de predecir correctamente nuevas observaciones. El mejor modelo es aquel que optimiza el equilibrio entre sesgo y varianza.

Por regla general, loa árboles que presentan pocas ramas, tienen un sesgo alto pues no son capaces de modelar correctamente la relación entre los predictores (como el problema que se presentaba con el modelo lineal y la relación no lineal), pero presentan una varianza muy baja. Mientras que, por el contrario, los árboles muy ramificados presentarán una varianza muy alta y muy poco sesgo (se ajustan mucho a los datos de entrenamiento).

La forma de solucionar este problema son los métodos de ensamblaje (*ensemble*). Estos métodos buscan a través de la combinación de múltiples modelos en uno nuevo, lograr un equilibrio entre sesgo y varianza. Consiguiendo de esta forma, mejores predicciones que cualquiera de los modelos que lo forman. Los dos principales métodos de ensamblaje de modelos son:

- ***Bagging**:* Se ajustan múltiples modelos en paralelo (por regla general árboles de clasificación o regresión) cada uno con un subconjunto de entrenamiento distinto extraído mediante técnicas de muestreo *bootstrapping*. Para predecir, todos los modelos que forman el conjunto participan aportando su predicción. Como valor final, se toma la media de todas las predicciones (árbol de regresión) o la clase más frecuente (árbol de clasificación).

  En el proceso de *Bagging* no genera sobreajuste en el modelo, por lo que el número de árboles creados no es un hiperparámetro crítico, es decir, por mucho que se incremente dicho número no habrá riesgo.

  Los modelos *Random Forest* estarían dentro de esta categoría.
- ***Boosting***: Se ajustan secuencialmente múltiples modelos sencillos (por regla general árboles que tienen una única rama -tocón-) de forma que cada modelo aprende de los errores del anterior.

  Los modelos de *Gradient Boosting* se encuentran dentro de esta tipología.

Aunque el objetivo buscado por ambas técnicas es el mismo, reducir el error total en las predicciones, ambas toman caminos opuestos para lograrlo. Mientras que *Bagging* ataca la componente del error perteneciente a la varianza agregando modelos con muy poco sesgo y mucha varianza. *Boosting* se encarga de reducir la otra componente del error (la del sesgo), empleando para ello modelos con muy poca varianza y mucho sesgo, ajustando secuencialmente reduce de forma notable ese último, sin apenas modificar la varianza del resultado y generar *overfiting,*

Otra de las grandes diferencias que presentan ambas técnicas reside en la manera en la que introducen las variaciones en el modelo de ensamblaje. En *bagging* cada modelo es diferente porque entrena con una muestra diferente obtenida de forma aleatoria a través de la técnica de muestreo *bootstrapping.* Mientras que *Boosting* varía el peso de las observaciones en cada iteración (en función del error que tenga el resultado con respecto al árbol anterior) dando lugar a ajustes diferenciados.

5.1.2.1 Random Forest

El algoritmo de *Random Forest* es una modificación del proceso de *Bagging* que consigue aún mejores resultados mediante la *decorrelación* de los árboles generados en el proceso (ya se vio durante el Análisis Exploratorio, lo útil que puede resultar este modelo para conseguir evitar el efecto de variables altamente correladas)

Recordando el apartado anterior, los beneficios del *Bagging* se basan en el hecho de que, promediando un conjunto de modelos, se consigue reducir la varianza. Esto es cierto siempre y cuando los modelos agregados no presenten variables que mantengan una alta correlación entre sí. Si esto ocurre, la reducción de varianza que se puede lograr es pequeña.

Random Forest evita el problema de alta correlación entre variables mediante la selección aleatoria de predictores antes de evaluar cada división. De esta forma existirán árboles en los que los predictores muy correlacionados no aparezcan, dando lugar a resultados que no se vean afectados por este factor.

La única diferencia entre el algoritmo *Random Forest* y *Bagging* es que el primero selecciona aleatoriamente m predictores, mientras que el segundo no hace esta selección (toma las p variables). Por lo que si en *Random Forest* se escoge $m = p$ los resultados serían equivalentes a los de *Bagging*. Para problemas de regresión, el número de predictores recomendado es $m \approx \frac{p}{3}$, mientras que para clasificación $m \approx \sqrt{p}$. Si los predictores se encuentran muy correlacionados, serán los valores pequeños de m los que consigan optimizar el rendimiento del modelo.

Como era de esperar, al igual que sucedía con *Bagging*, *Random Forest* tampoco sufre problemas de sobreajuste al aumentar el número de árboles creados en el proceso. La reducción del error se mantiene estable a partir de un determinado valor de árboles.

5.1.2.2 Gradient Boosting

Se trata de uno de los algoritmos que pertenece a la familia del Método de Ensamblaje *Boosting*. Se caracteriza, como previamente se mencionaba, en ajustar de forma secuencial

múltiples árboles poco ramificados (tocones), para que de manera iterativa éstos vayan aprendiendo el error que ha generado el anterior, y poco a poco se vaya corrigiendo.

Los principales hiperparámetros a optimizar son:

- El número de iteraciones o árboles poco ramificados
- *Learning rate:* controla el ritmo al que aprende el modelo. Cuanto menor sea su valor, más árboles se necesitan para alcanzar mejores resultados, pero menor es el riesgo de sobreajuste. Valores entre 0,01 y 0,001
- Tamaño máximo permitido de los árboles que forman el modelo. Valores entre 1 y 5.

La idea generalizada del funcionamiento es la siguiente:

1- Se ajusta el primer árbol f_1 con el que se predice la variable respuesta y. Se calculan los residuos $y - f_1(x)$. A continuación, se ajusta un nuevo modelo f_2 que intenta predecir los residuos del modelo anterior, es decir, trata de corregir los errores que ha dejado el modelo f_1

$$f_1(x) \approx y$$
$$f_2(x) \approx y - f_1(x)$$

2- En la siguiente iteración, se calculan los residuos de los dos modelos de forma conjunta

$$f_3(x) \approx y - f_1(x) - f_2(x)$$

3- Este proceso se repite *M* veces, de forma que cada nuevo modelo minimiza los errores del modelo anterior.

Si el algoritmo siguiera iterando, tarde o temprano se empezaría a generar sobreajuste en el modelo. Para ello, se introduce un término llamado *learning rate* (λ), que limita la influencia de cada modelo en el conjunto. Quedando finalmente:

$$y \approx \lambda f_1(x) + \lambda f_2(x) + \cdots + \lambda f_m(x)$$

5.1.3 PERCEPTRÓN MULTICAPA (MLP)

El perceptrón multicapa se trata de un modelo genérico de red neuronal que combina numerosas capas de neuronas ocultas presentando arquitecturas relativamente sencillas. Cada neurona se encuentra conectada a todas las neuronas de la capa anterior y a las de la capa posterior.

De esta forma, a continuación, se procede a explicar en detalle qué son las Redes Neuronales, cómo funcionan, y sus principales componentes.

5.1.3.1 Redes Neuronales

Las redes neuronales son modelos creados al ordenar operaciones matemáticas sencillas siguiendo una determinada estructura. La forma más común de representar la estructura de una red neuronal es mediante el uso de capas, formadas a su vez por neuronas que se corresponderían con las unidades. Cada neurona, como se comentaba previamente, se encuentra conectada a la capa anterior y posterior a mediante pesos, cuya función es regular la información que se propaga a lo largo de la red.

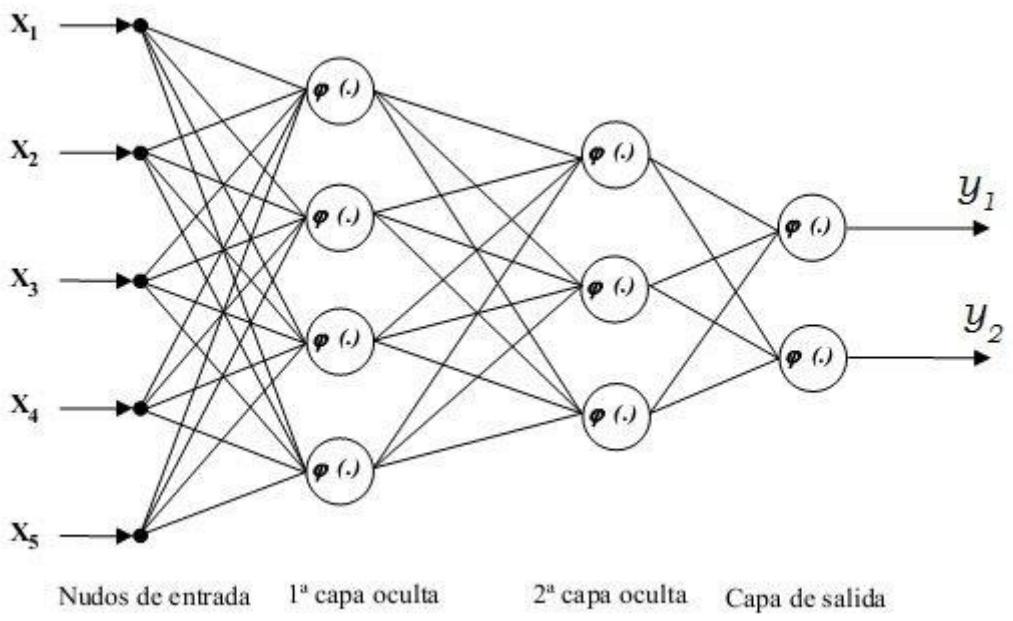

Ilustración 5.1: Representación de Red Neuronal

La primera capa presenta los nudos de entrada y recibe toda la información en bruto, es decir el valor de las propias variables. Las capas intermedias (1ª y 2ª capa oculta), reciben los valores de la primera capa ponderados por los pesos (líneas que unen unas capas con otras). La última capa será la que reciba los valores generados en las capas ocultas, combinándolos para generar la predicción.

Para entender mejor las redes neuronales, es necesario conocer en profundidad 3 componentes clave a la hora de construir la red neuronal que genere posteriormente las predicciones.

5.1.3.1.1 La Neurona

La neurona es la unidad funcional de las redes neuronales. Dentro de cada neurona, ocurren dos operaciones básicas: la suma ponderada de las entradas y la aplicación de una función de activación.

En la primera parte, se multiplica cada valor de entrada por su peso asociado. Este es el valor de entrada a la neurona. A continuación, la llamada función de activación, que se dedica a transformar el valor de entrada en un valor de salida.

Aunque el valor que llega a la neurona, multiplicación de los pesos por las entradas siempre es una combinación lineal, gracias a la función de activación se pueden generar salidas muy diversas. Es la función de activación la principal responsable de aprender relaciones más complejas que las lineales.

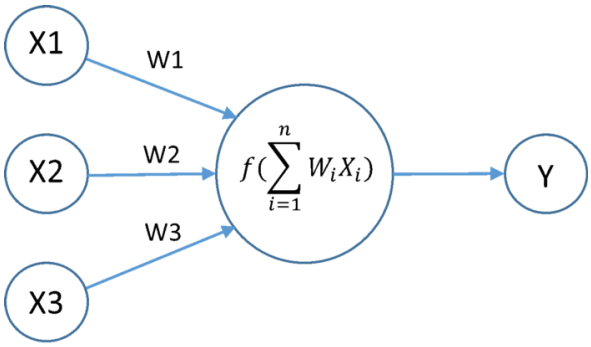

Ilustración 5.2: Representación de Neurona, Función de Activación y Pesos

5.1.3.1.2 Función de Activación

Las funciones de activación son las encargadas de controlar qué información pasa de una capa a otra. Estas funciones son las que convierten el valor de entrada a una neurona en un nuevo valor. La gran mayoría convierten el valor de entrada en un valor dentro del rango (0, 1) o (-1, 1).

Las dos funciones de activación más empleadas son las siguientes:

- **Rectified linear unit (ReLU)**

  Función de activación que sólo activa la neurona si el valor de entrada se encuentra por encima de 0. Si es así, el valor de salida aumenta de forma lineal con el de la entrada. De esta forma sólo retiene valores positivos y descarta los negativos dándole una activación de cero.

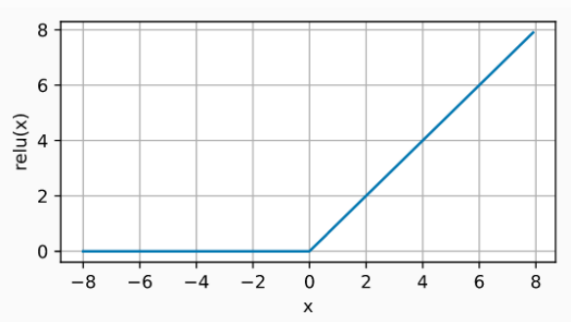

Ilustración 5.3: Representación de Función de Activación ReLu

- **Sigmoidal**

  Función de activación que transforma valores del rango (-inf, +inf) a valores en el rango (0, 1). Dado que sus resultados pueden interpretarse como probabilidades, tiene una gran utilidad en problemas de clasificación binaria

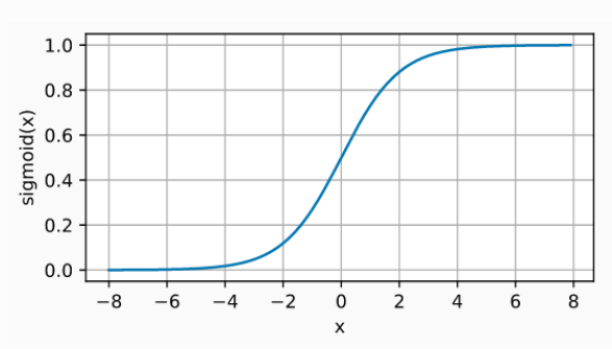

Ilustración 5.4: Representación de Función de Activación Sigmoidal

5.1.3.1.3 Función de pérdida

La función de pérdida es la encargada de medir la distancia entre el valor real y el valor predicho por la red. A medida que disminuye el valor de esta función, menor error habrá y por lo tanto mejores serán las predicciones.

En problemas de regresión, las más utilizadas son el error cuadrático medio y el error absoluto medio (esta última será la que se utilice en el presente trabajo para hacer la comparación final entre modelos)

5.1.3.2 Hiperparámetros

El gran problema a la vez que aliado de las redes neuronales es la gran flexibilidad que tienen a la hora de modelar problemas. Esto puede dar lugar a sobreajustes, incapacitando el modelo para hacer nuevas predicciones. La forma de evitar esto será realizando una apropiada configuración de hiperparámetros. A continuación, se explican los más importantes a tener en cuenta:

5.1.3.2.1 Tamaño de las capas

Este valor determinará en gran medida la capacidad de aprendizaje del algoritmo. Es directamente proporcional al número de neuronas que forma la red neuronal completa.

La capa de entrada tendrá tantas neuronas como número de predictores y la capa de salida tendrá una única neurona en problemas de regresión (variable output) y tantas como clases

en los problemas de clasificación. En cambio, el tamaño de las capas intermedias vendrá definido por las necesidades del usuario.

Cuanto mayor sea el número de neuronas, mayor será la complejidad de las relaciones que puede aprender el modelo. Sin embargo, los costes computacionales aumentarán de forma notable, así como el tiempo de entrenamiento.

5.1.3.2.2 Learning Rate

El *Learning Rate* establece cómo de rápido pueden cambiar los parámetros de un modelo a medida que aprende. Si es demasiado alto, el proceso de optimización irá saltando de una región a otra sin que el modelo aprenda realmente. Mientras que, si es demasiado pequeño, el proceso de entrenamiento tardará demasiado y quizás no llegue a completarse.

5.1.3.2.3 Weight Decay

El objetivo de este hiperparámetro es que los pesos tomen valores excesivamente elevados. Evitando así, que ciertas neuronas dominen el comportamiento de la red al completo y que los ruidos tengan pesos muy diferentes a 0.

5.2 ENTRENAMIENTO

Con la base teórica asentada para los 4 modelos, es momento de pasar a código y comenzar a generar resultados.

El primer paso será definir los conjuntos de entrenamiento y test. El 80% de los datos se quedan en el *dataset* de entrenamiento (*fTR*) mientras que el 20% restante se destinará para formar el de test (*fTS*).

Como método de validación genérico para todos los modelos se utilizará el método de *Cross-Validation* con k (número de subconjutos en los que se divide el *dataset* de entrenamiento) igual a 10. Para aquellos métodos que requieran de remuestreo *Bootstrapping*, se hará una ligera modificación en el método de validación utilizando *Repeated Cross-Validation*.

Los hiperparámetros pertenecientes a los algoritmos se optimizarán en base a la métrica de la raíz cuadrada del error cuadrático medio (*RMSE*) durante validación. De tal forma que, el algoritmo que resulte óptimo tendrá aquella combinación de hiperparámetros que minimice esta variable.

Durante el entrenamiento de los diferentes modelos se procederá de la misma forma, primero se entrenará un modelo que no esté ajustado, se mostrarán sus métricas, y posteriormente se obtendrá el óptimo mediante ajustes iterativos de sus hiperparámetros.

5.2.1 ÁRBOL DE REGRESIÓN

5.2.1.1 Árbol de Regresión Modelo 1

El árbol de regresión se genera mediante la librería *rpart*. Es semejante al árbol de clasificación que genera dicha librería, modificando el método por *anova.* Presentaría la siguiente forma:

```
set.seed(150) #For replication
tree.fit_1 <- rpart(form = `(l/km)`~.,
                    data = fTR,
                    method = "anova")
```

El árbol de regresión formado no utiliza la totalidad de las variables. Se obtiene mediante el uso de las que se muestran en la Tabla 5.1. Eliminando las variables *Altura del Vehículo* y *Cilindrada*.

Tabla 5.1: Variables utilizadas por el Árbol de Regresión

Nombre de la variable
Consumo Combinado
Emisiones CO2
Entorno
Estilo

Presentando el aspecto que se muestra a continuación

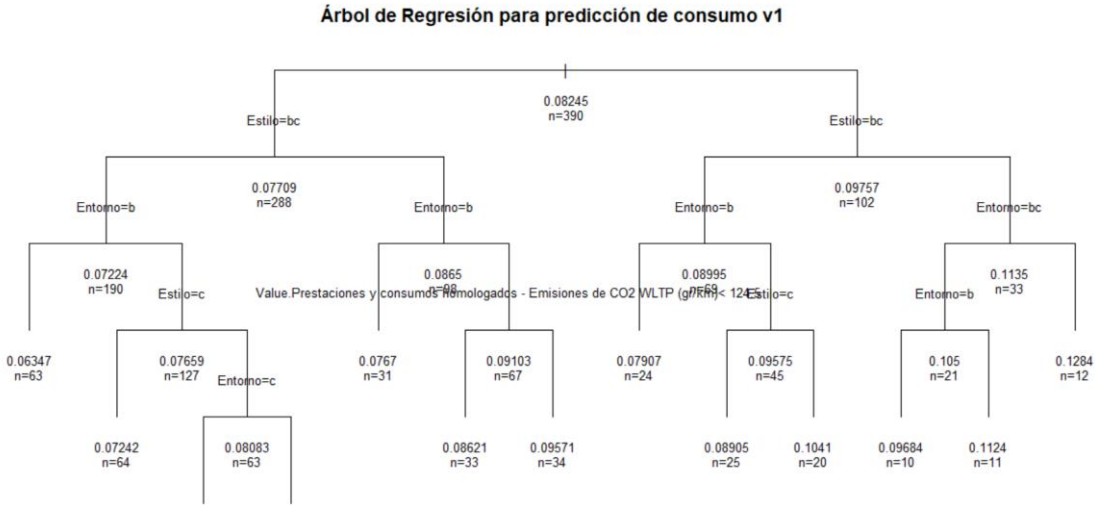

Ilustración 5.5: Representación gráfica del Árbol de Regresión generado

Mientras que el coeficiente de complejidad presenta la siguiente evolución a medida que varía el árbol.

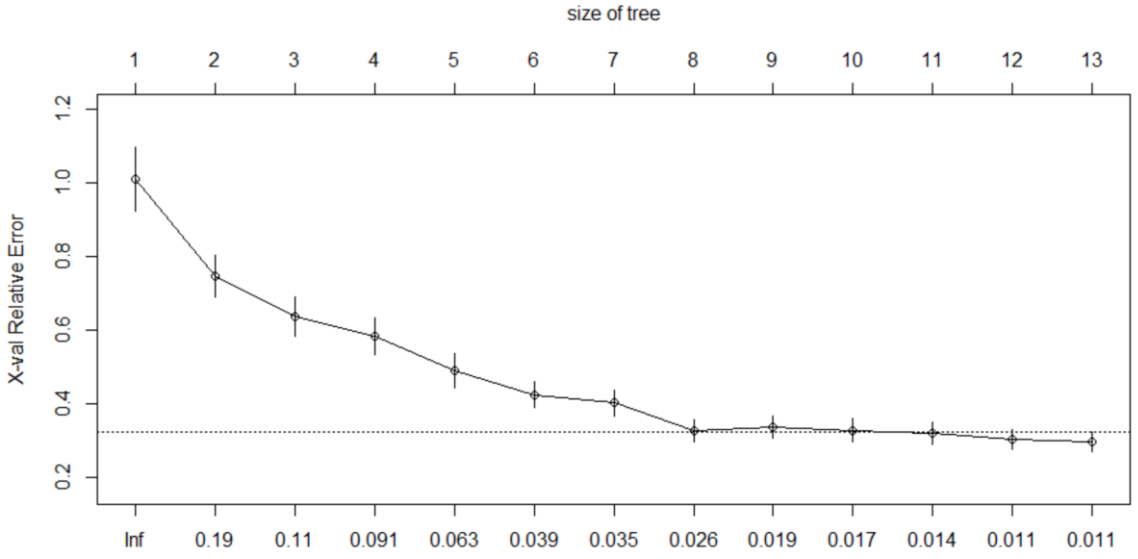

Ilustración 5.6: Evolución del error para el tamaño del árbol y coeficiente de complejidad Cp

Como era de esperar, según aumenta el tamaño del árbol se reduce este coeficiente, reduciendo a su vez el valor del Error Relativo (medida que se busca minimizar). También se observa que, a partir de un valor determinado en la complejidad de los árboles, el Error Relativo parece quedarse estabilizado.

5.2.1.2 Árbol de Regresión, optimización del modelo

Dado que el árbol se ha construido con las condiciones que impone por defecto la librería *rpart*, se procede a optimizarlo para analizar diferencias entre las métricas de ambos.

Los hiperparámetros configurables para un árbol de regresión son:

- Complejidad o penalización – A través del parámetro *cp*
- Número mínimo de observaciones necesarias para realizar un nuevo corte en el árbol *minsplit*
- Máxima profundidad *maxdepth*

Se buscará la pareja formada por el número mínimo de observaciones necesarias y la máxima profundidad que optimicen las métricas del árbol. Para ello se creará un *dataframe* con las posibles combinaciones de parámetros, y se iterará con los árboles formados hasta conseguir encontrar el mejor. A continuación, se muestra la forma que presentaría el árbol óptimo:

```
set.seed(150) #For replication
tree.fit_optimo <- rpart(form = `(l/km)`~.,
                         data = fTR,
                         method  = "anova",
                         control = list(minsplit = 16, maxdepth = 13 , cp =
0.01151443))
```

Dicho árbol presenta exactamente la misma estructura que el más básico, mostrado en Figura 5.5

Obteniendo las siguientes métricas en validación

Tabla 5.2: Métricas de Validación cruzada en entrenamiento para Árbol de Regresión Optimizado

Métrica	*Resultado*
Rel Error	0.22942
xerror	0.29588
xstd	0.026429

5.2.2 RANDOM FOREST

5.2.2.1 Random Forest Modelo 1

Para este caso, el paquete *Caret* será el elegido para realizar el entrenamiento. Este paquete trae predefinida la función que crea el Random Forest (*rf*). Antes habrá que tener en cuenta que el hiperparámetro a configurar será:

- Número máximo predictores por árbol m (*mtry* en el código)

Como método de validación para entrenamiento se utilizará *Repeated Cross-Validation* con $k = 10$ y el número de repeticiones igual a 3.

Para este primer modelo se utiliza el valor teórico para los modelos de regresión, teniendo $m \approx \frac{p}{3}$. El código se muestra de la siguiente forma:

```
set.seed(150) #For replication
mtry <- ((ncol(fTR) - 1)/3)
tunegrid <- expand.grid(.mtry=mtry)
rf.fit_1 <- train(`(l/km)`~.,
             data = fTR,
             method = "rf",
             tuneGrid=tunegrid,
             trControl=ctrl_tune_repeated)
```

Obteniendo los resultados de *resampling* que se muestran a continuación:

Tabla 5.3: Métricas de Validación cruzada en entrenamiento para Random Forest 1

Métrica	*Resultado*
Rsquared	0.8380299
RMSE	0.007552462
MAE	0.005727836

5.2.2.2 Random Forest, optimización del modelo

Random Forest permite optimizar sus resultados iterando hasta escoger el número máximo de predictores por árbol $mtry$ que las métricas del modelo completo alcancen el mínimo o el máximo correspondiente.

Si se realiza dicho cambio, se crearía un *tunegrid* que abarque todas las posibles combinaciones de predictores (desde 1 a 6) quedando el algoritmo de la siguiente forma

```
set.seed(150) #For replication
tunegrid <- expand.grid(.mtry=(1:6))
rf.fit_optimo <- train(`(l/km)`~.,
                   data = fTR,
                   method = "rf",
                   tuneGrid=tunegrid,
                   trControl=ctrl_tune_repeated)
```

Si la función entrena con diferentes combinaciones de variables explicativas se obtiene un valor de RMSE para cada valor del conjunto de dichas variables.

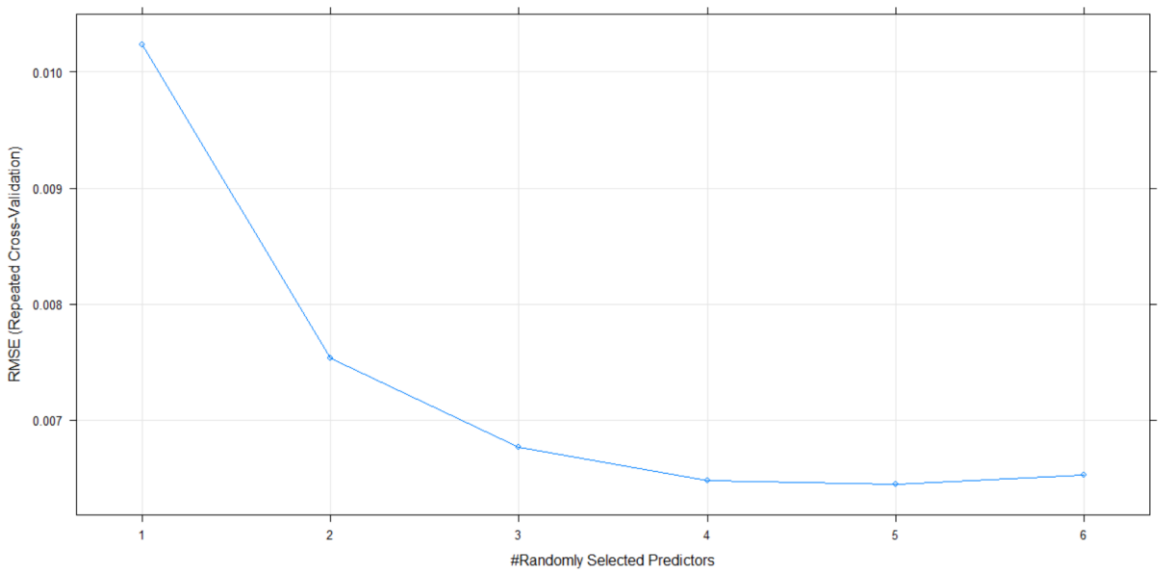

Ilustración 5.7: Variación del RMSE para número de variables seleccionadas

Tal y como se puede ver en la Figura 5.7 el mínimo valor del *RMSE* se alcanza para un valor $mtry = 5$. Siendo este hiperparámetro el que optimiza el rendimiento del algoritmo.

Por lo que para el *Random Forest* óptimo se obtendrían las siguientes métricas de *resampling*

Tabla 5.4: Métricas de Validación cruzada en entrenamiento para Random Forest Optimizado

Métrica	*rfe_optimo*
Rsquared	0.8616149
RMSE	0.006450275
MAE	0.004985589

5.2.3 GRADIENT BOOSTING

5.2.3.1 Gradient Boosting Modelo 1

El algoritmo se ha generado a través de la librería *Caret* utilizando *Repeated Cross-Validation* como técnica de control para el entrenamiento. Que al igual que en el modelo de *Random Forest*, subdivide el *dataset* de entrenamiento en $k = 10$ y el número de repeticiones igual a 3. Por lo tanto, el primer modelo de *Gradient Boosting* tendría la siguiente forma

```
set.seed(150) #For replication
gbm.fit_1 <- train(`(l/km)`~.,
                   data = fTR,
                   method = "gbm",
                   trControl = ctrl_tune_repeated,
                   verbose = FALSE)
```

Para este modelo *Caret* escoge mediante iteraciones los hiperparámetros que se han nombrado previamente, obteniendo:

- El número de iteraciones o árboles poco ramificados igual a 150 (*n.trees*)
- *Learning rate (shrinkage)* 0,1 (muy superior a los valores recomendables, el algoritmo termina de iterar muy pronto, pero hay riesgo de sobre entrenamiento)

- Tamaño máximo permitido de los árboles que forman el modelo igual 3 (*interaction.depth*)

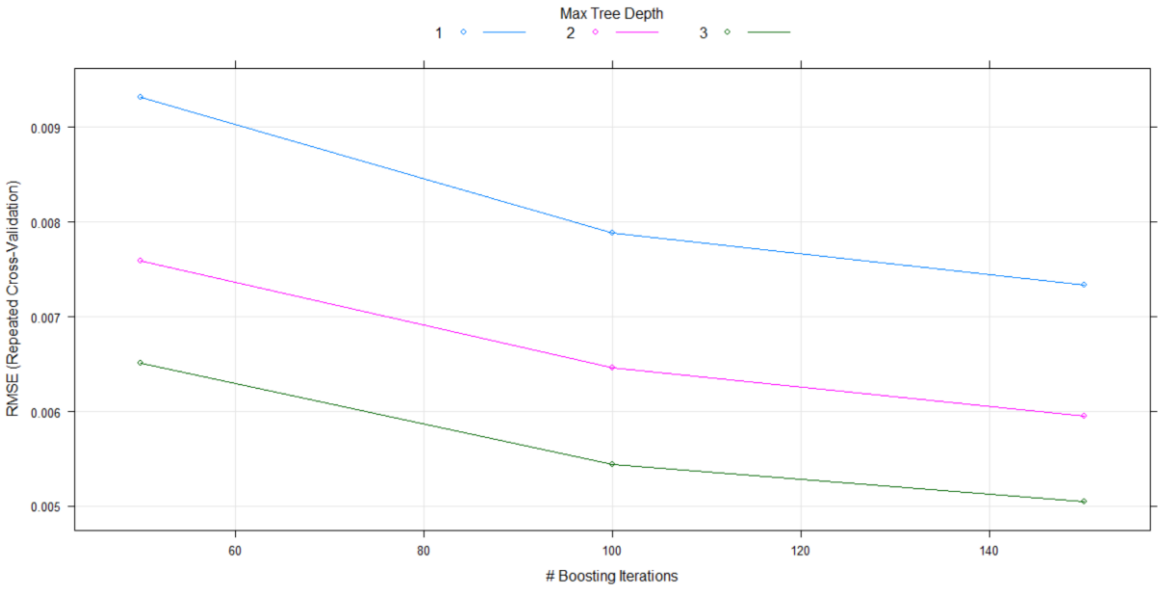

Ilustración 5.8: Vairación del RMSE por profundidad de los árboles creados e iteraciones Boosting

Obteniendo las gráficas mostradas en la Figura 5.8 en la que se puede ver que a medida que aumentan las iteraciones el RMSE de validación disminuye notablemente, y que los valores mínimos se alcanzan para aquellos árboles cuyo tamaño máximo es igual a 3

Con estos hiperparámetros se obtienen los siguientes resultados de *resampling*

Tabla 5.5: Métricas de Validación cruzada en entrenamiento para Gradient Boosting 1

Métrica	*gbm.fit_1*
Rsquared	0.9175178
RMSE	0.005045610
MAE	0.003654248

5.2.3.2 Gradient Boosting, optimización del modelo

Gradient Boosting permite optimizar su rendimiento de la misma forma que lo hacía *Random Forest*, creando un set de datos con valores posibles para cada uno de sus parámetros e iterando a lo largo y ancho de conjunto hasta llegar al mínimo del *RMSE*.

```
gbmGrid <-  expand.grid(interaction.depth = c(1, 5, 9),
                        n.trees = (1:30)*100,
                        shrinkage = c(0.01, 0.001))

set.seed(150) #For replication
gbm.fit_optimo <- train(`(l/km)`~.,
                    data = fTR,
                    method = "gbm",
                    trControl = ctrl_tune_repeated,
                    verbose = FALSE,
                    tuneGrid = gbmGrid)
```

Se ha creado un nuevo *dataset* llamado *gbmGrid* que contiene los 3 hiperparámetros que necesita *Gradient Boosting*. Estos 3 hiperparámetros estarán distribuidos en 4 columnas, siendo cada fila una posible combinación de estas 3 variables. Los intervalos utilizados para cada uno de ellos son:

- El número de iteraciones o árboles poco ramificados irá de 50 a 3000
- *Learning rate* igual a 0,01 y 0,001
- Tamaño máximo permitido de los árboles que forman el modelo igual 1, 5 y 9

Con este modelo se obtiene el siguiente gráfico. De él se pueden extraer múltiples observaciones:

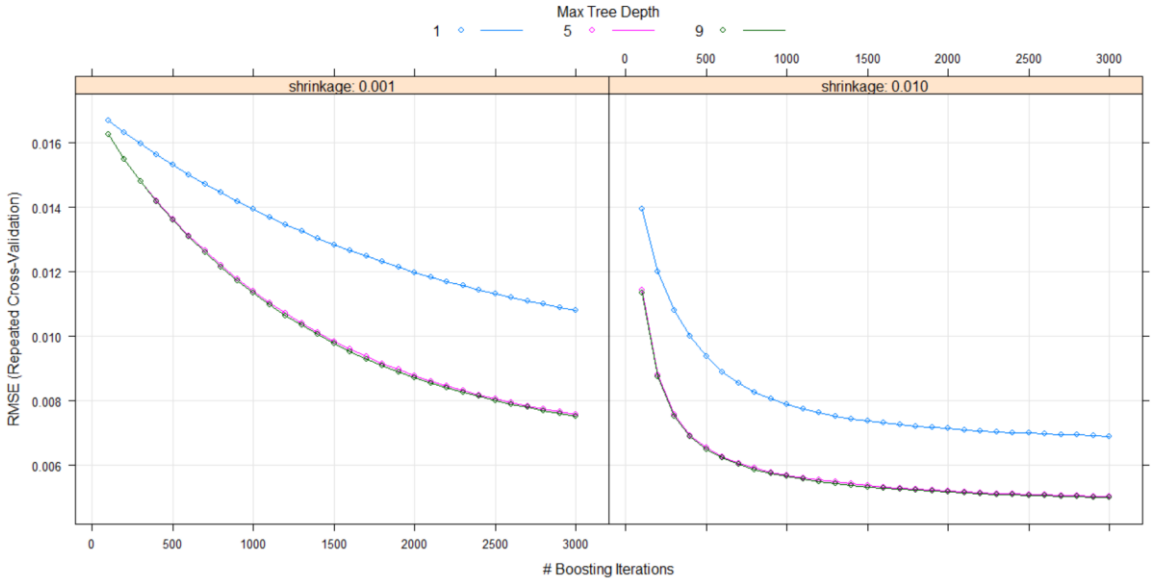

Ilustración 5.9: Variación del RMSE por profundidad del árbol, número de iteraciones y learning rate

- Se observa de forma clara cómo para un *learning rate* más bajo (0,001), el algoritmo necesitaría más iteraciones hasta alcanzar valores cercanos a los alcanzados con el más alto (0,01). Con este último *learning rate* el algoritmo aprende mucho más rápido.
- Ambos tienen el mismo comportamiento con respecto al tamaño máximo permitido en los árboles. A medida que aumenta este valor, el algoritmo aprende más rápido y alcanza mejores métricas. El problema que podría surgir con estos valores es que el modelo terminara sobre ajustándose a la muestra surgiendo problemas con la varianza del modelo.
- Para futuros trabajos estaría bien entrenar el primer modelo (*learning rate* de 0,001) para un número mayor de árboles. Alcanzaría valores semejantes de RMSE a los del otro modelo, y además sería más complicado que el modelo se encontrara sobre ajustado.

Teniendo en cuenta que el modelo que mejor resultados de *resampling* presenta es el que tiene los siguientes hiperparámetros:

- El número de iteraciones o árboles poco ramificados igual a 3000
- *Learning rate* igual a 0,01
- Tamaño máximo permitido de los árboles que forman el modelo igual 9

Se tomará como el óptimo (posteriormente se validará si existe sobreajuste con el subconjunto de datos de prueba), obteniendo las siguientes métricas

Tabla 5.6: Métricas de Validación cruzada en entrenamiento para Gradient Boosting Optimizado

Métrica	*gbmFit_optimo*
Rsquared	0.9183930
RMSE	0.005003811
MAE	0.003543136

5.2.4 PERCEPTRÓN MULTICAPA (MLP)

5.2.4.1 MLP Modelo 1

Al igual que en los modelos anteriores, se recurrirá a la librería *Caret* que proporciona el modelo *nnet* para el entrenamiento de redes neuronales.

En cuanto al método de validación, de nuevo se utilizará *Cross-Validation* con $k = 10$.

El primer MLP creado, sin modificar los hiperparámetros, presentaría la siguiente forma:

```
set.seed(150) #For replication
mlp.fit1 <- train(form = `(l/km)` ~ ., data = fTR,
                  method = "nnet", linout = TRUE,
                  preProcess = c("center","scale"),
                  trControl = ctrl_tune,
                  metric = "RMSE")
```

Se deja que sea la propia librería de *Caret* la que busque la combinación óptima de hiperparámetros.

Además, es importante realizar un centrado y escalado de las variables antes de que entren en la red neuronal. De esta forma, nos aseguramos de que la magnitud de los inputs es semejante. De no ser así, los pesos conectados a ciertos inputs se actualizarán mucho más rápidos que otros, penalizando sobre el proceso de aprendizaje.

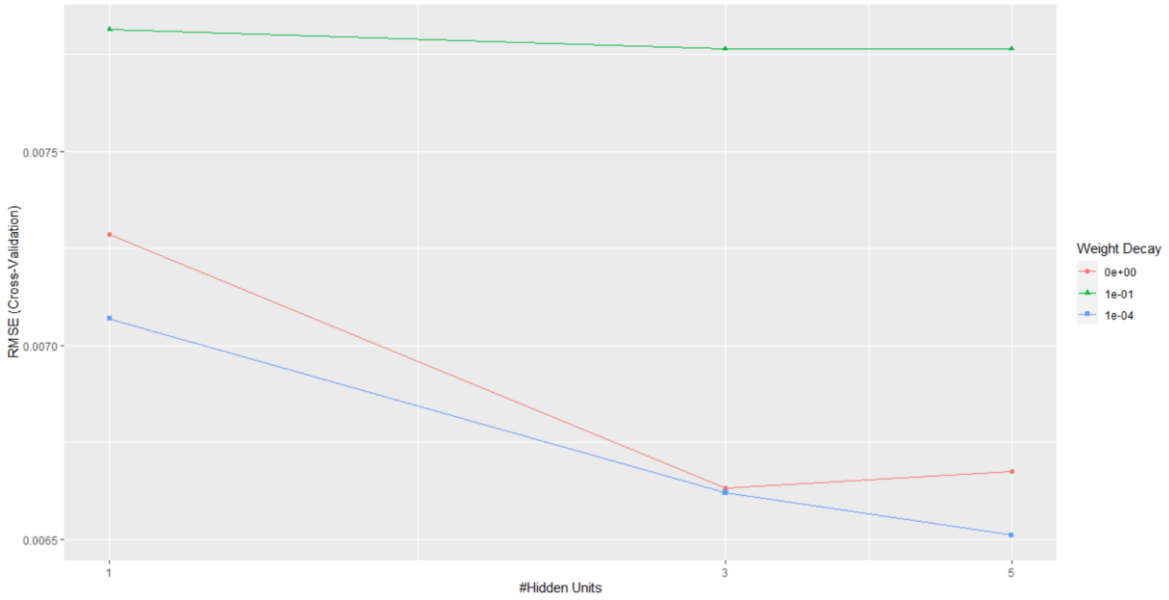

Ilustración 5.10: Variación del RMSE por Weight Decay y capas interiores ocultas

Como se puede observar en la Figura 5.10 será la siguiente pareja de hiperparámetros la que alcance mejores resultados

- Tamaño de las capas igual a 5
- *Weight Decay* igual a 1e-04

Tras analizar la variación del *RMSE* para el modelo entrenado, en la Figura 5.11 se puede observar la representación gráfica que tendría la red neuronal utilizada:

- Las capas intermedias ocultas se ven representadas por H
- Las capas representadas por B son las que representan el sesgo, aplican valores constantes a los nodos.
- Las líneas que unen las capas representan los pesos. Las negras son pesos positivos, mientras que las grises negativos. El ancho de la línea es directamente proporcional a la magnitud relativa del peso.

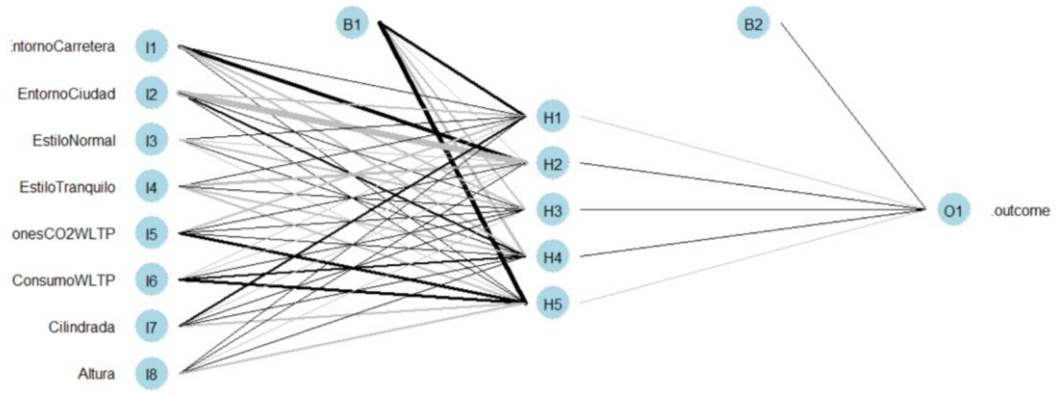

Ilustración 5.11: Representación gráfica de la Red Neuronal 1

También es interesante conocer la importancia de los predictores a la hora de modelizar la variable de salida. Para ello se ha realizado un análisis de la sensibilidad, mostrado en la Figura 5.12. En él, se pueden observar los siguientes puntos:

- La variable que tiene más peso dentro del modelo es *Entorno Carretera*, seguida de *Entorno Tranquilo*.
- *Entorno Ciudad* presenta una relación no lineal con la variable output, puesto que tiene una desviación estándar muy alta. Es por ello, por lo que se encuentra clasificada como la tercera variable más importante.
- En el tercer gráfico se puede observar que la distribución asociada a *Estilo Normal* es la más estrecha de todas, por lo que existirá una relación lineal entre esta variable

y el output. De hecho, esto mismo también se puede observar en el primer gráfico, cuyo valor de varianza se sitúa como el más bajo

- El resto de las variables parecen presentar relaciones no lineales con la de salida

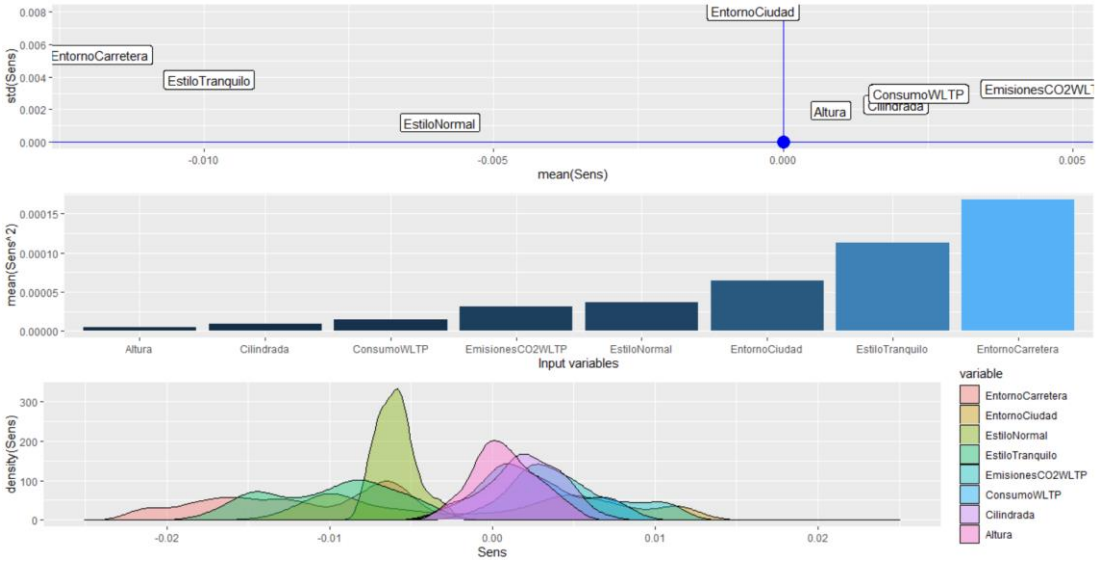

Ilustración 5.12: Gráficos representativos de Sensibilidad del Modelo Perceptrón Multicapa 1

Este modelo obtiene los siguientes resultados de validación

Tabla 5.7: Métricas de Validación cruzada en entrenamiento para Perceptrón Multicapa 1

Métrica	*Mlp.fit_1*
Rsquared	0.8578602
RMSE	0.006127507
MAE	0.005001915

5.2.4.2 MLP, optimización del modelo

Al igual que en los modelos anteriores, se puede generar un rango de valores para estos hiperparámetros y que sea el propio modelo iterando el que consiga obtener la pareja óptima de ellos. El algoritmo entonces tendría la siguiente forma

```
set.seed(150)
mlp.fit_optimo = train(form = `(l/km)` ~ ., data = fTR,
                       method = "nnet", linout = TRUE,
                       maxit = 500,
                       tuneGrid = expand.grid(size = seq(5,25,length.out = 10),
                                              decay=c(10^(-9),0.0001,0.001,0.01,0.1,1)),
                       preProcess = c("center","scale"),
                       trControl = ctrl_tune,
                       metric = "RMSE")
```

Generando los siguientes gráficos:

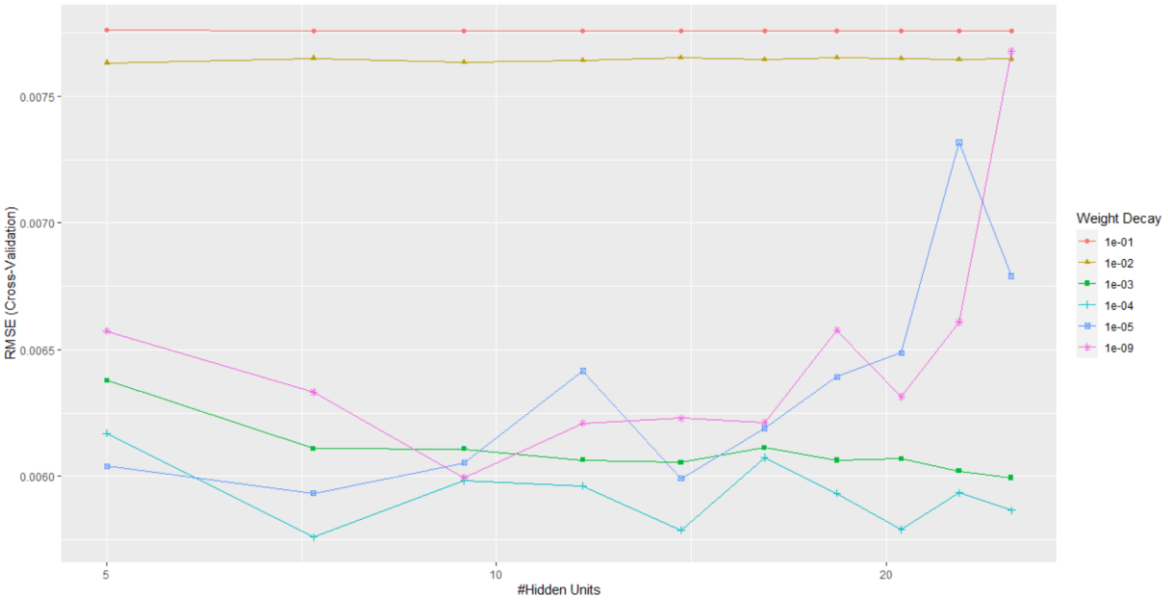

Ilustración 5.13: Variación del RMSE por Weight Decay y número de capas interiores ocultas

De la Figutra 5.13 se puede extraer las siguientes conclusiones:

- El *Weight Decay* que minimiza el RMSE en *Cross-Validation* es 1e-04 al igual que en el primer modelo
- El número de capas del modelo óptimo ocultas se encuentra en torno a 7

- Para valores de *Weight Decay* muy altos la red neuronal apenas entrena, no modificando apenas el *RMSE* para ningún valor de las capas intermedias ocultas.
- Se puede observar de forma clara que la red neuronal con *Weight Decay* igual a 1e-09, a partir de un número de capas intermedias cercano a 20 empeora notablemente su comportamiento. Muy probablemente estemos ante un caso de sobreajuste del modelo.

La forma que presenta la red neuronal se encuentra en la Figura 5.14

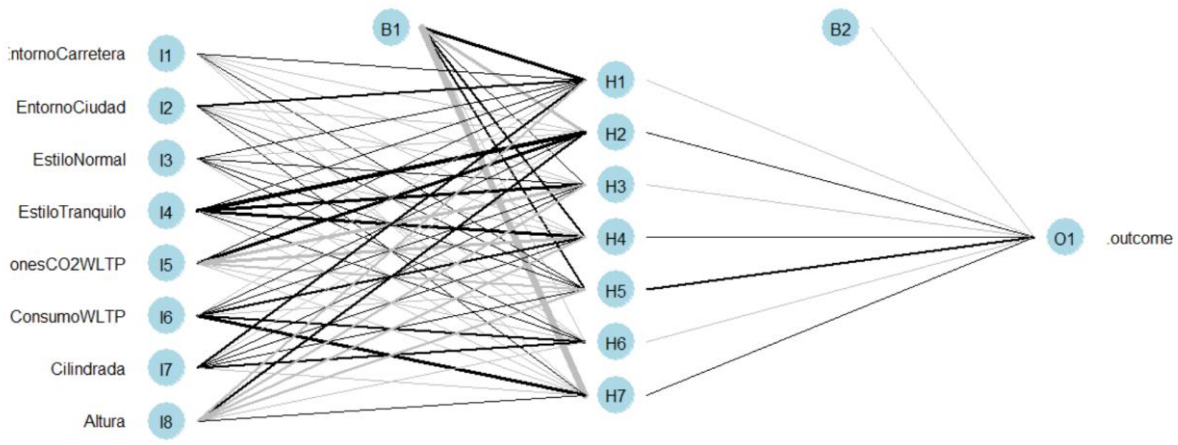

Ilustración 5.14: Representación gráfica del Perceptrón Multicapa Optimizado

De nuevo, el análisis de sensibilidad puede arrojar información importante acerca de las relaciones entre los predictores y la variable de salida:

- *Entorno Carretera* seguida de *Estilo Tranquilo* siguen siendo las variables que más pesan dentro del modelo. Para este modelo *Estilo Carretera* aumenta su importancia, destacando aún más sobre el resto.
- El resto de las variables tienen un peso semejante entre ellas

- *Estilo Normal* presenta una curva de densidad aún más estrecha y alta. Por lo que muy probablemente siga manteniendo la relación lineal con la variable de salida
- Las demás variables presentan curvas de densidad mucho más anchas, debido a que se posicionan en puntos más altos en el eje y (desviación estándar) del primer gráfico. Esto es indicativo de relaciones no lineales entre estos predictores y la variable que tratan de predecir.

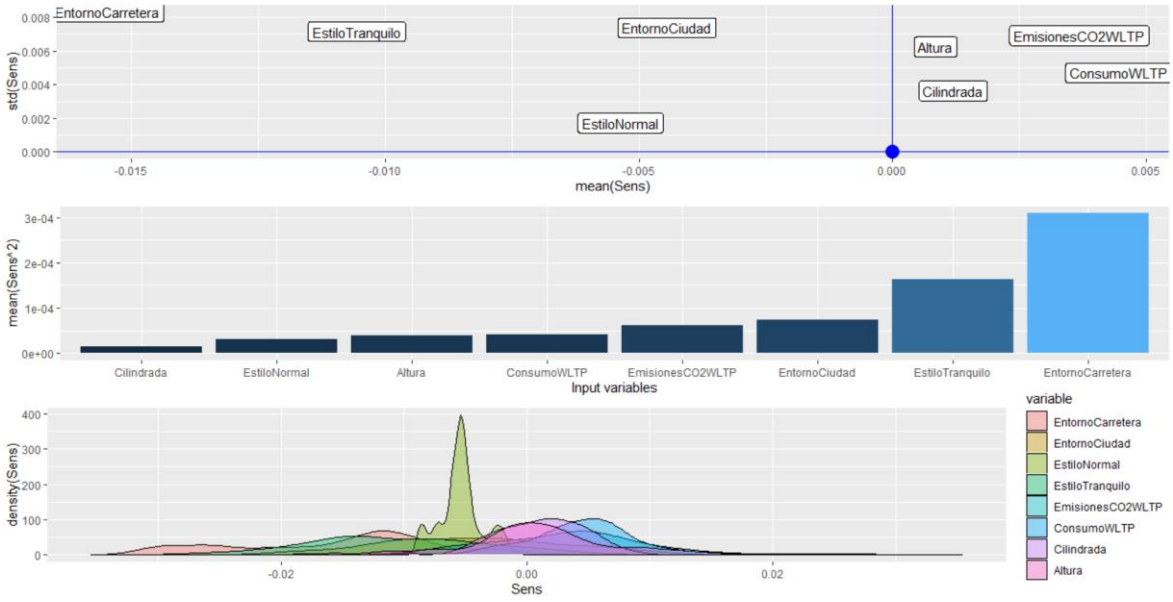

Ilustración 5.15: Gráficos representativos de Sensibilidad del Modelo Perceptrón Multicapa Optimizado

Por último, en la Tabla 5.15 se muestran las métricas obtenidas durante la validación cruzada.

Tabla 5.8: Métricas de Validación cruzada en entrenamiento para Perceptrón Multicapa Optimizado

Métrica	*Mlp.fit_optimo*
Rsquared	0.8899398
RMSE	0.005759780
MAE	0.004560204

5.3 COMPARATIVA DE MODELOS EN PREDICCIÓN

Las dos medidas más comunes a la hora de evaluar la precisión de modelos de regresión son el Error Medio Absoluto (*MAE*) y la Raíz Cuadrada del Error Cuadrático Medio (*RMSE*).

Antes de presentar las métricas, es necesario hacer una breve introducción teórica de ambas, y elegir la encargada de evaluar qué modelo genera mejores predicciones.

El *MAE* se calcula como un promedio de diferencias absolutas entre los valores reales y las predicciones (error de predicción). El *MAE* es una puntuación lineal, lo que significa que todas las diferencias individuales se ponderan por igual en el promedio. Esto quiere decir que, un error de 10 sobre 0 implica que es exactamente el doble que uno de 5 sobre 0.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \tilde{y}_i|$$

El *RMSE* también se encarga de medir la media de la magnitud del error, siendo la raíz cuadrada de la media de las diferencias entre los valores reales y las predicciones (error de predicción) al cuadrado.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{N}(y_j - \tilde{y}_j)^2}$$

Como semejanzas entre ambas, se encuentra que las dos expresan la media del error de predicción del modelo, y tienen un rango de valores de 0 a ∞. Para ambas se cumple que menores valores implican un mejor rendimiento del modelo.

La principal diferencia entre ambas reside en la raíz cuadrada del *RMSE*. Tomar una raíz de la media de los errores al cuadrado implica que dará un peso muy alto a errores que sean grandes. Por lo tanto, esta medida puede ser interesante para aquellos casos de uso en los que se traten de minimizar los errores muy elevados. Por ejemplo, si nos vamos al ejemplo explicado previamente, un error de 10 sobre 0 será más del doble que uno de 5 sobre 0.

Otra de las diferencias entre ambos reside en que el *RMSE* no toma valores absolutos, pudiendo aportar valores no deseados en ciertos cálculos matemáticos.

Dado que el presente caso de uso se destinará a el potencial cálculo de consumos de combustible para flotas de decenas o centenas de coches, será recomendable utilizar una métrica que de alguna manera penalice las desviaciones de consumos muy elevadas. Dado que ello dará lugar a una potencial desviación de costes (tanto positiva como negativa) muy diferente a la que las predicciones han mostrado.

Es por esto, por lo que la medida utilizada para comparar los modelos en test será el RMSE.

A continuación, se pueden observar las métricas para los diferentes modelos creados.

Tabla 5.9: Comparativa métricas de predicción de los modelos creados

Modelo	*RMSE Entrenamiento*	*RMSE Test*	*Delta RMSE (%)*
Árbol de Regresión	0,00794871	0,00929914	17%
Random Forest	0,00330055	0,00568928	72%
Gradient Boosting	0,00360029	0,00402752	12%
Perceptrón Muticapa (*MLP*)	0,00408905	0,00499566	22%

En el gráfico mostrado en la Figura 5.16 se encuentran representados los valores recogidos en la Tabla 5.9. En ella, se puede ver de forma clara cómo el modelo de *Gradient Boosting* además de ser el que mejores predicciones ofrece, es el que generaliza mejor en dichas predicciones.

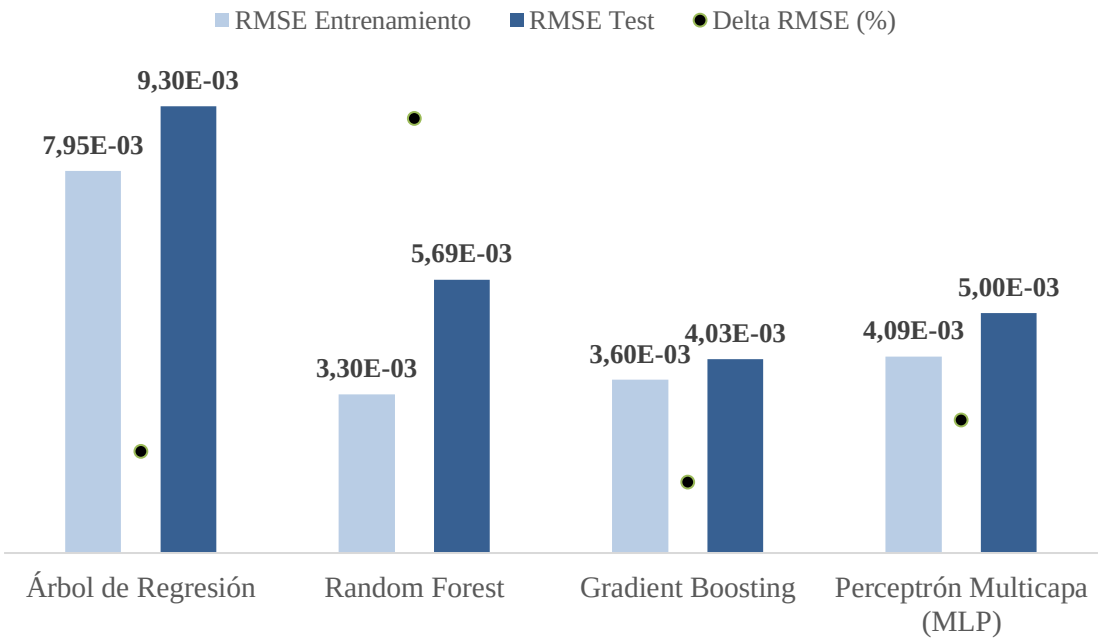

Ilustración 5.16: Representación gráfica de las métricas de los modelos de predicción creados

Por lo tanto, será el modelo de *Gradient Boosting* el seleccionado para predecir los consumos de los vehículos cuyo Tipo de Combustible sea Gasolina.

6. CÁLCULO DE COSTES DE CONSUMO

Con el modelo ya creado, se pueden comenzar a generar predicciones de consumo para los vehículos actuales que presenta la empresa en su flota, y los llamados "futuribles".

Estos últimos pueden o venir dados por el propio cliente (debido a tenga ofertas comerciales de varios vehículos y quiera conocer en detalle cuáles serían los más adecuados), o estar abiertos a todo el mercado español. De ambas formas, los potenciales vehículos se extraerían de la base de datos en la que se encuentran las características.

Se predicen los consumos para las 9 combinaciones de Estilo y Entorno. De tal forma que para cada vehículo habrá 9 filas iguales.

A través de la información extraída del Formulario de Conductores, se conoce el porcentaje aproximado de los kilómetros, que los propios conductores estiman conducir en un Estilo o Entorno. Con estos porcentajes se estimaría el *mix* de consumos.

Conociendo el consumo por conductor y vehículo, habrá que hacer una unión con las tablas de Factores de Emisión y Precios de Combustible/Electricidad. Con la información de los costes unitarios de la energía y los factores de emisión, la operación restante será multiplicar estas columnas por el *Mix* de Consumo para el conductor y vehículo que corresponda.

Por ejemplo, el *Mix* de Consumo el caso de un conductor que estima hacer el 40% de su kilometraje diario por Carretera, el 20% por Ciudad y el 40% restante por Autovía, el 90% del tiempo conduciendo de forma Tranquila y el 10% Deportivo se calcularía tal y como se encuentra representado en el esquema de la Figura 6.1

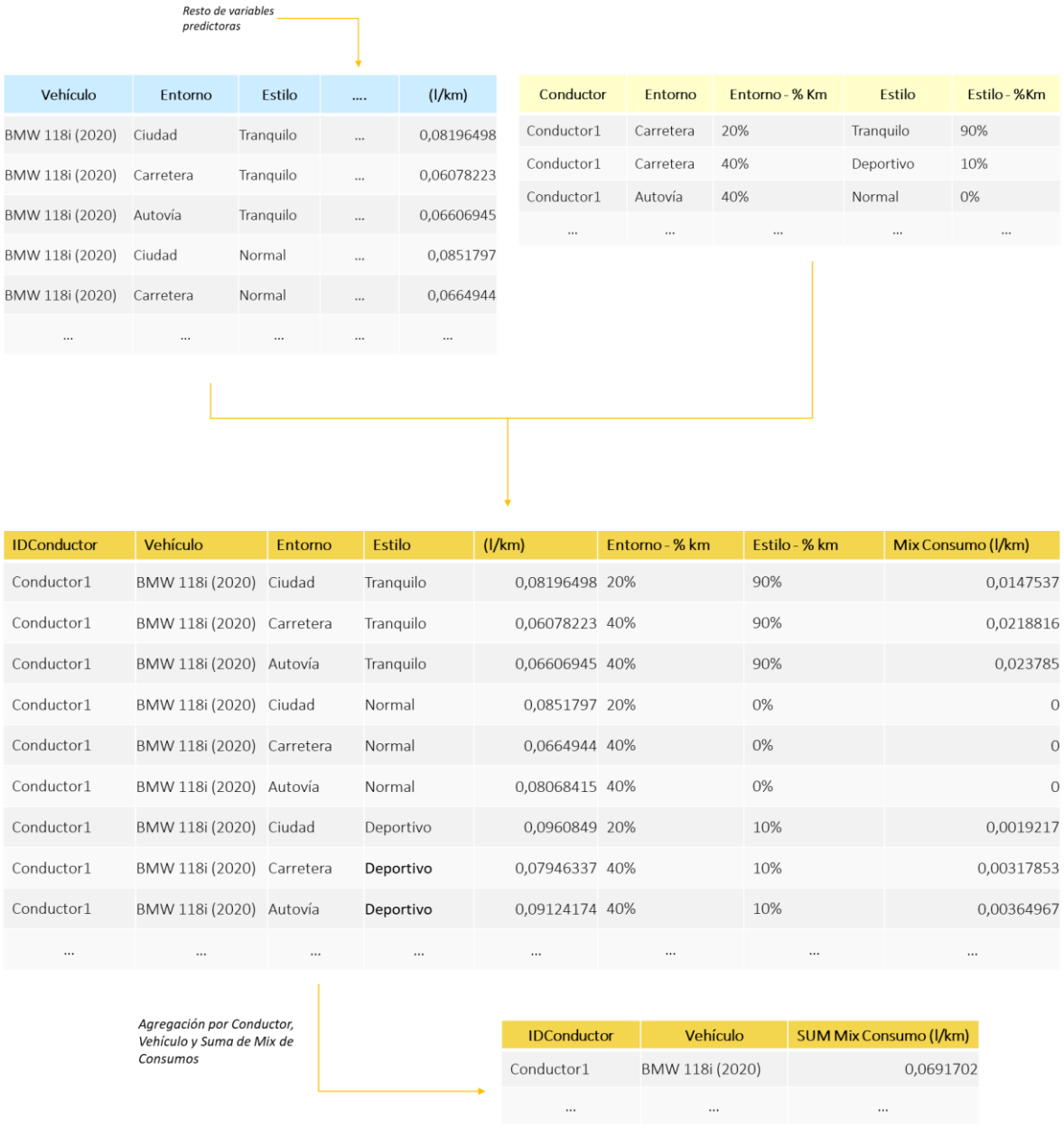

Ilustración 6.1: Representación esquemática del proceso de cálculo de costes

7. VISUALIZACIÓN

A lo largo del presente Capítulo, se pretende explicar de forma detallada la visualización diseñada para crear los informes para clientes que soliciten la Transformación Energética de Flota a través de *SwitchFleet*.

Este informe está desarrollado mediante la herramienta de Power Bi. Y se envía a través de un enlace creado tras la publicación del mismo en la web. Consta de 4 pestañas que muestran información diferenciada:

- **Pestaña 1**: Resumen Ejecutivo de las Flotas propuestas
- **Pestaña 2**: Impacto en la desviación de costes y emisiones de cada flota
- **Pestaña 3**: Impacto Medioambiental vs Económico de la Flota Actual y Propuesta
- **Pestaña 4**: Resumen de la oferta de Renting correspondiente y enlace al archivo PDF con todos los detalles

A continuación, se muestran las imágenes de cada Pestaña con los principales elementos que hacen de este Informe una herramienta totalmente interactiva.

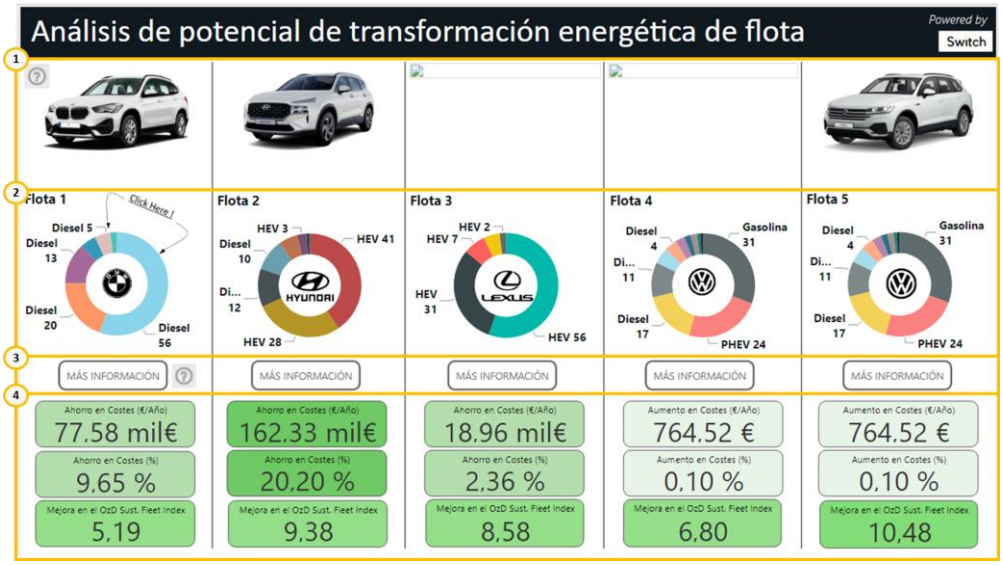

Ilustración 7.1: Pestaña 1 del Informe Power Bi

La Figura 7.1 se corresponde con la Pestaña 1, en la que se muestra el Resumen Ejecutivo del informe completo. El usuario, de un vistazo, puede conocer la Marca de la flota propuesta, de qué tipo de vehículos se compone y los ahorros o incrementos en costes anuales.

- 1. Imágenes de los vehículos que forman la flota. En caso de que la flota esté compuesta de varios, se mostrará sólo la primera de ellas. Esta imagen se encuentra conectada con el gráfico circular que se muestra en el punto 2.
- 2. Gráficos circulares e imagen de logos de la marca que compone la flota. En los gráficos circulares, cada logo indica el recuento de modelos por tipo de combustible (cada segmento del círculo se corresponde con un modelo)
- 3. Botones para acceder a los detalles de cada flota
- 4. Tarjetas con resumen de desviaciones en costes y OzD Index (índice calculado en base a las emisiones anuales de CO2 por kilómetro). El nombre y color de las tarjetas varía en función del valor que se muestra en ellas para resaltar las flotas más interesantes en cuanto a ahorro de costes y mejora de las emisiones.

Ilustración 7.2: Pestaña 2 del Informe Power Bi

La imagen de la Figura 7.2 es la que se mostraría al hacer *click* en el botón "MÁS INFORMACIÓN" en la Parte 3, Pestaña 1, para la Flota 1. En ella se encuentran los siguientes elementos:

- 1. Filtros para seleccionar las marcas (si hay más de una) y modelos que forman la Flota que se está mostrando. Este filtro se encuentra conectado con toda la información mostrada en la pestaña
- 2. Gráficos de cascada con desviación de costes y emisiones. En el correspondiente a costes, se encuentran separados aquellos pertenecientes al ahorro o incremento que supone el alquiler de la nueva Flota, y el generado por el consumo de combustible.
- 3. Mapa que representa el número de vehículos por zona territorial de la empresa cliente
- 4. Selector de Flota. Si se seleccionara cualquiera de los botones, el informe iría a la misma página correspondiente a la Flota seleccionada.
- 5. Imagen del vehículo y logo de la marca filtrados
- 6. Botón para avanzar a la siguiente pestaña
- 7. Recuento de vehículos del modelo seleccionado
- 8. Flota de la que se está mostrando la información
- 9. Botón para volver a la Pestaña 1
- 10. Botón para volver a la anterior pestaña (en este caso Pestaña 1)
- 11. Botón creado a través de un Marcador, cuyo objetivo es limpiar los filtros seleccionados previamente

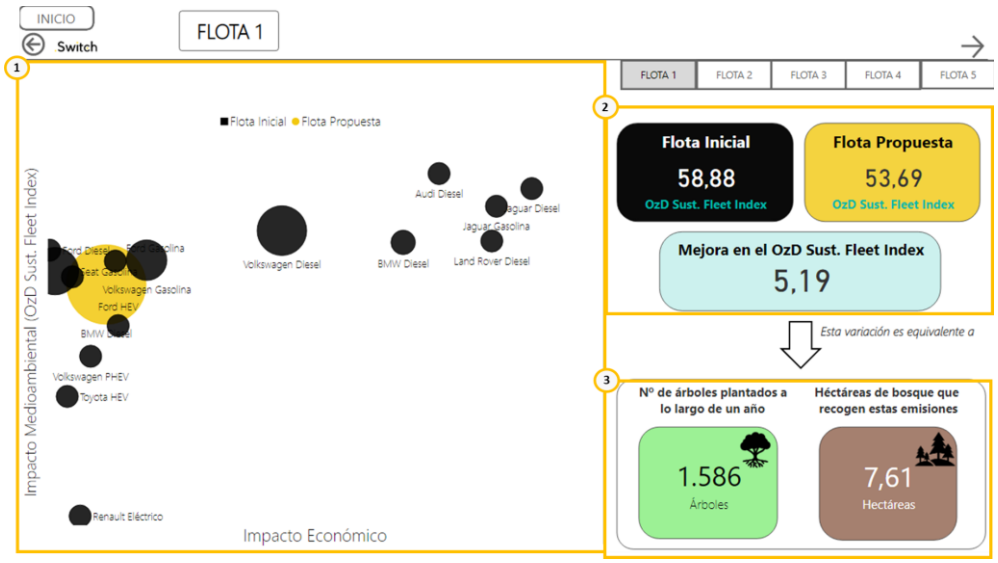

Ilustración 7.3: Pestaña 3 del Informe Power Bi

En la Figura 7.3 se encuentran representados los 3 siguientes elementos:

- 1. Gráfico en el que se encuentran enfrentados el Impacto generado en los costes por marca y tipo de combustible. Lo ideal sería que la flota propuesta se encontrara más a la izquierda y debajo de la flota actual del cliente. El radio de las circunferencias marca el recuento de vehículos propuestos de dicha marca y tipo de combustible.
- 2. Tarjetas con el valor de la media del *OzD Sustainability Fleet Index* para la flota actual y propuesta, y diferencia entre ambas. De ser positiva, el título de esta sería "Mejora en el OzD Sust. Fleet Index", mientras que si fuera negativa sería "Deterioro en el OzD Sust. Fleet Index".
- 3. Tarjetas con el valor equivalente en árboles y hectáreas de bosque plantados o talados anualmente en función del valor de ahorro o incremento en emisiones de CO_2. A través de estas tarjetas se busca crear conciencia al cliente de lo que realmente supone para el mundo, el cambio a la nueva flota en términos de emisiones.

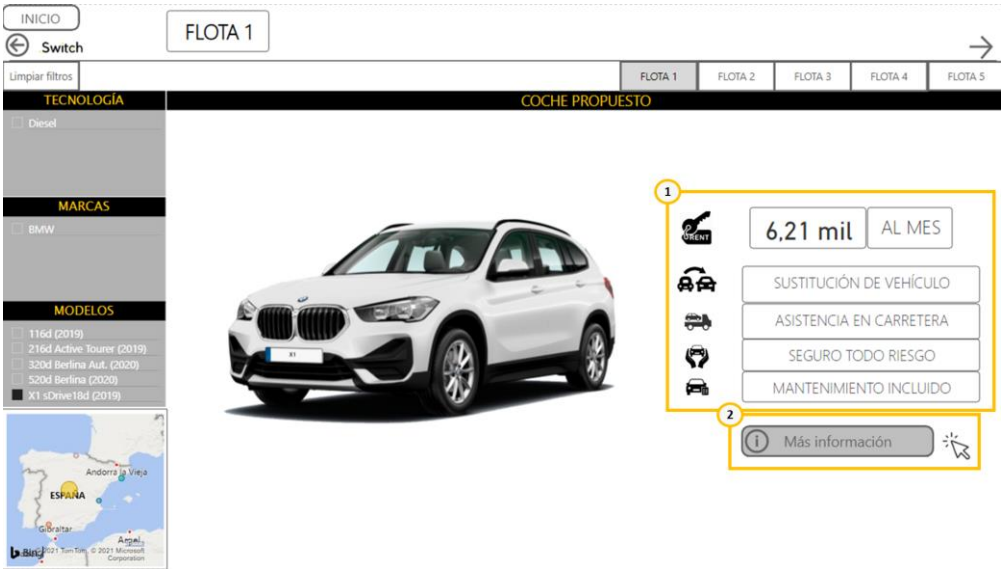

Ilustración 7.4: Pestaña 4 del Informe Power Bi

Por último, en la Figura 7.4 se muestran principalmente los siguientes elementos:

- 1. Resumen de la oferta de alquiler del vehículo mostrado con los principales ítems a tener en cuenta dentro de la misma (de momento el proceso de extracción es manual, pero en un futuro sería automático)
- 2. Botón que dirige al usuario a la *url* donde se encuentra almacenada la oferta correspondiente.

8. CONCLUSIONES Y TRABAJOS FUTUROS

Dado que el alcance del presente proyecto es muy amplio, buscando desarrollar una aplicación que sea capaz de tratar toda la cadena de valor del dato (desde la extracción hasta la visualización, pasando por el modelado), es lógico pensar que alguno de los módulos que lo conforman no se encuentra todo lo optimizado que podría estar.

Desde el primer momento se ha buscado desarrollar un mínimo producto viable que funcionara, aunque fuera muy básico, y de ahí ir introduciendo mejoras paulatinas en su operatividad.

Si se analizan en detalle cada uno de los segmentos de la cadena que forma el proceso se tendrían los siguientes pasos a desarrollar:

1. **Arquitectura y *pipeline***. Es imprescindible contar con un orquestador que automatice extracciones, transformaciones y procesos de almacenamiento. Este orquestados además debería de ejecutar cada proceso asociado a un ETL en un tiempo marcado por los requisitos de la aplicación. Por otro lado, sería conveniente migrar toda la estructura a servidores en la nube. De esta forma, se conseguiría generar una aplicación con procesos más escalables y flexibilidad en el almacenamiento disponible.
2. **Fuentes de datos**. Se ha destinado un esfuerzo considerable en hacer que la base de datos de las Características de Vehículos adoptara una estructura definida (eliminando columnas con un porcentaje elevado de valores nulos). Sería de gran utilidad para este caso, el estudio de la conveniencia de bases de datos no estructuradas. Pese a suponer un salto tecnológico representativo en el almacenamiento de los datos, supondría una solución mucho más flexible en la que se perdería mucha menos información.

   Otra de las grandes mejoras esperables en términos de fuentes de datos vendrá a partir de la instalación de los dispositivos de Geotab, una vez los clientes los adquieran.

A través del incremento en la utilización de esta plataforma, se espera poder eliminar el formulario de conductores, y así evitar la segmentación en el estilo y entorno de conducción. Cada conductor tendrá unos consumos asociados por sus trayectos diarios que se almacenan a través de los dispositivos conectados al vehículo.

3. **Análisis Exploratorio**. En el modelo de imputación de huecos en blanco, estaría bien generar algo semejante a un modelo de *ensemble* entre el modelo de *K-NN imputer* y el modelo *MICE* (*Multiple imputation by chained equations*). Este último tiene un comportamiento más complejo que el utilizado, y es por eso por lo que no se utilizó en el proceso. Generando un modelo de *ensemble* se podría llegar a valores mucho más certeros en aquellos que son cualitativos, y más exactos en aquellos cuantitativos.
4. **Modelos de Predicción**. En este apartado las grandes mejoras pueden llegar a través de la utilización de nuevos modelos o el *fine-tuning* de los utilizados. El modelo *XG Boost* hubiera sido el candidato perfecto para probar si es posible generar nuevos modelos con potencial para mejorar las métricas obtenidas por los antiguos.

   También hay que tener en cuenta el equilibrio entre la utilización de modelos demasiado complejos que presenten excelentes métricas, pero que funcionen como una caja negra (comportamiento opaco, de difícil acceso en caso de fallo), o utilizar modelos con métricas algo más bajas.
5. **Cálculo de costes.** Automatización de la obtención de ofertas de proveedores de servicios de *renting* de vehículos
6. **Visualización.** Siguiendo con el punto anterior, una de las grandes mejoras a desarrollar sería la creación de un modelo, que mediante técnicas de NLP pudiera extraer información de las ofertas comerciales, y la almacenara en una base de datos asociada a la propia oferta.

En cuanto a las funcionalidades de la aplicación *SwitchFleet* sería interesante desarrollar mediante técnicas de NLP, un *rating* de cada vehículo. Esta variable podría entrar a formar parte del proceso de filtrado (ahora mismo sólo cuentan costes y emisiones). Los datos a través de los cuales se pueden generar los algoritmos de NLP podrían ser obtenidos de redes sociales o páginas web actualizadas.

De esta forma, no sólo serán los datos cuantitativos los que fijen qué flota es la más adecuada para un cliente, sino que entrarán en juego datos difícilmente medibles que no aparecen en ninguna base de datos (sensaciones de conducción, espacio interior, valoración de la marca…)

9. BIBLIOGRAFÍA

[1] Asociación Empresarial para el Desarrollo e Impulso del Vehículo Eléctrico (AEDIVE) (septiembre 3, 2020). *Ozone Drive, startup tecnológica especializada en servicios avanzados de movilidad eléctrica.* https://aedive.es/ozone-drive-startup-tecnologica-novilidad -electrica/

[2] ElReferente (mayo 2, 2016). *Ozone Drive busca financiación para crear más vehículos eléctricos.* https://elreferente.es/tecnologicos/ozone-drive-busca-financiacion-para-crear-mas-coches-electricos/

[3] Indiegogo (mayo 21, 2015). *Ozone Drive, the Silent Revolution of Electric Cars.* *https://www.indiegogo.com/projects/ozone-drive-connecting-islands-to-electric-cars#/*

[4] Bulut, O. (enero 11, 2021). *Effective Feature Selection: Recursive Feature Elimination Using R.* https://towardsdatascience.com/effective-feature-selection-recursive-feature-elimination-using-r-148ff998e4f7

[5] Brownlee, J. (agosto 28, 2020). *Recursive Feature Elimination (RFE) for Feature Selection in Python*. Machine Learning Mastery. https://machinelearningmastery.com/rfe-feature-selection-in-python/

[6] Badr, W. (enero 5, 2019*). 6 Different Ways to Compensate for Missing Values In a Dataset.* https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779

[7] Gelman, A. y Hill, J. (septiembre 2012). "Missing data imputation". *Data Analysis Using Regression and Multilevel/Hierarchical* Models. New York, NY, USA pp 529-544

[8] Amat, J. (mayo, 2020). *Detección de anomalías: Isolation Forest. c*ienciadedatos.net. https://www.cienciadedatos.net/documentos/66_deteccion_anomalias_isolationforest.html#Introducci%C3%B3n

[9] Lewinson, E. (julio 2, 2018). *Outlier Detection with Isolation Forest.* https://towardsdatascience.com/outlier-detection-with-isolation-forest-3d190448d45e

[10] Lewinson, E. (marzo 10, 2019). *Outlier Detection with Extended Isolation Forest.* https://towardsdatascience.com/outlier-detection-with-extended-isolation-forest-1e248a3fe97b

[11] Tony, F., Ming, K. y Zhi-Hua, Z. *Isolation Forest*

[12] Amat, J. (abril, 2020). *Machine Learning con R y mlr3*. cienciadedatos.net. https://www.cienciadedatos.net/documentos/60_machine_learning_con_r_y_mlr3#An%C3%A1lisis_exploratorio

[13] Muñoz, A., Sánchez E. y Portela J. (octubre, 2020). *Chapter 2 – Classification.* Machine Learning I. Universidad Pontificia de Comillas

[14] Amat, J. (abril, 2020). *Machine Learning con R y Caret.* cienciadedatos.net. https://www.cienciadedatos.net/documentos/41_machine_learning_con_r_y_caret#Conclusi%C3%B3n_an%C3%A1lisis_exploratorio

[15] Muñoz, A., Sánchez E. y Portela J. (octubre, 2020). *Chapter 3 – Regression.* Machine Learning I. Universidad Pontificia de Comillas

[16] Sánchez, E. (enero 2021). *Ensemble Learning.* Machine Learning II. Universidad Pontificia de Comillas

[17] Amat, J. (febrero, 2017). *Árboles de decisión, random forest, gradient boosting y C5.0.* cienciadedatos.net. https://www.cienciadedatos.net/documentos/33_arboles_decision_random_forest_gradient_boosting_c50#%C3%81rboles_de_regresi%C3%B3n

[18] Brownlee, J. (abril 20, 2020). *How to develop a Random Forest Ensemble in Python.* Machine Learning Mastery. https://machinelearningmastery.com/random-forest-ensemble-in-python/

[19] Kuhn, M. (marzo 27, 2019). *5 – Model Training and tuning.* The Caret Package. https://topepo.github.io/caret/model-training-and-tuning.html

[20] Dive Into Deep Learning. *4.1 Multilayer Perceptron. http://d2l.ai/chapter_multilayer-perceptrons/mlp.html*

[21] Baumbah, L. (julio 2020). *Neural Network – Activation Function.* https://morioh.com/p/21b55ba475f9

[22] Stöttner, T. (mayo 16, 2019). *Why Data should be Normalized before Training a Neural Network.* https://towardsdatascience.com/why-data-should-be-normalized-before-training-a-neural-network-c626b7f66c7d

[23] Heim, R. (diciembre 26, 2019). *How to Train a Multilayer Perceptron Neural Network.* All About Circuits. https://www.allaboutcircuits.com/technical-articles/how-to-train-a-multilayer-perceptron-neural-network/

[24] Pizarro, J., Portela, J. y Muñoz, A. (febrero 2021). *NeuralSens: Sensitivity Analysys of Neural Networks.*

[25] JJ (marzo 23, 2016). *MAE and RMSE — Which Metric is Better?*. https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d

[26] Sab (junio 2, 2019). *MAE vs MSE vs RMSE*. http://zerospectrum.com/2019/06/02/mae-vs-mse-vs-rmse/

ANEXO A. DIAGRAMA GANTT DESARROLLO