UNIVERSIDAD ADOLFO IBAÑEZ

TESIS DE MAGÍSTER

APRENDIZAJE AUTOMÁTICO PARA DETECTAR FACTIBILIDAD EN EL RUTEO DE VEHÍCULOS

Autor:
Rodrigo Esteban Morán Benavente

Profesores guía:
Luis Aburto
Felipe Lagos
Wilfredo Yushimito

Tesis realizada acorde a los requerimientos para el grado de Magíster en Ingeniería Industrial e Investigación de Operaciones y para el título de Ingeniero Civil Industrial

de la

Facultad de Ingeniería y Ciencias

3 de octubre de 2023

UAI FACULTAD DE INGENIERÍA Y CIENCIAS

UNIVERSIDAD ADOLFO IBAÑEZ

Resumen

Facultad de Ingeniería y Ciencias

Magíster en Ingeniería Industrial e Investigación de Operaciones y para el título de Ingeniero Civil Industrial

APRENDIZAJE AUTOMÁTICO PARA DETECTAR FACTIBILIDAD EN EL RUTEO DE VEHÍCULOS

por Rodrigo Esteban Morán Benavente

Una de las aplicaciones más importantes en las Cadenas de Suministro, es el diseño de red de abastecimiento, en el cual, el diseño de rutas de transporte cumple un rol importante. Conjunto con el desarrollo y uso de tecnologías de Inteligencia Artificial, ha generado una amplia oportunidad de mejoras en los desempeños, anticipándose a escenarios reduciendo los costos de transporte y logísticos. En la práctica se suele resolver el problema de ruteo de vehículos mediante heurísticas, aunque en algunos casos puede generar ruteos insatisfactorios, de los cuales no se visitan todos los puntos de una instancia sabiendo el planificador que la calidad de la solución puede ser mejor. Provocando que los planificadores encargados tengan que adaptar o modificar el problema para que las heurísticas generen las rutas adecuadas.

La factibilidad de una instancia de ruteo es un factor determinante al momento de planificar rutas en la última milla, dado que repercute directamente en el tiempo de resolución como la calidad de la solución. La región factible de una instancia de ruteo está determinada por los parámetros de la instancia, donde dependiendo del caso, influye la distribución geográfica de los puntos, la demanda solicitada, ventanas horarias por cumplir, entre otros. Esta investigación busca desarrollar una herramienta capaz de predecir la factibilidad de la instancia de ruteo para la empresa SimpliRoute, anticipándose a la resolución del problema reduciendo costos. Además, en esta investigación se estudiarán las posibles causas que influyen en una baja inserción de clientes en las rutas.

Palabras clave: Aprendizaje automático, VRP, Factibilidad.

Índice general

Capítulo 1

Introducción

La logística son todas las operaciones llevadas a cabo para que un producto llegue al consumidor desde el lugar donde se obtienen las materias primas, pasando por el lugar de su producción. Son principalmente las operaciones de transporte, almacenamiento y distribución de los productos de mercado, por ello, se considera a la logística como operaciones externas a la fabricación de un producto. El objetivo de la logística es facilitar al consumidor el producto deseado, en la cantidad y momentos precisos, en el mejor punto de venta y que todo ello se realice al menor coste posible.

El último segmento de la cadena de suministro está asociado al problema de última milla, donde hay varias opciones para entregar el producto final. La última milla, es una gestión de transporte de paquetería centrado en el último trayecto que ha de realizarse en la entrega final. La importancia de la última milla en la logística ayuda a satisfacer al cliente en la mayor brevedad de tiempo y por otro lado reducir costos asociados al personal como el gasto en transporte.

Una de las claves para optimizar la última milla es mediante la optimización de rutas de transporte, con un planificador de rutas se podrá resolver el problema de ruteo de vehículos (VRP, por sus siglas en inglés), con el fin de determinar cuáles son las rutas más eficientes en función de factores como el tráfico, atascos en tiempo real, calles cortadas, obras que puedan suponer una limitante, entre otros.

1.1. Problema de Ruteo de Vehículos

El problema de ruteo de vehículos es un problema de optimización combinatorio y de programación entera, que responde ante la determinación del conjunto óptimo de rutas que debe realizar una flota de vehículos para servir a un conjunto determinado de clientes [1]. Los métodos exactos encuentran la solución óptima, pero a un costo de recursos y cantidad de tiempo enormes para problemas de gran tamaño. El costo de encontrar una solución óptima crece exponencialmente a medida que aumenta el tamaño del problema, ante lo cual se prefiere usar métodos heurísticos que encuentran soluciones cercanas al óptimo con un costo de recursos y cantidad de tiempo razonable.

Hay una gran cantidad de variantes dentro de la familia del VRP, a diferencia del problema original, se incluyen modificaciones o restricciones adicionales que responden ante diferentes necesidades, como lo pueden ser capacidad limitada por los vehículos (CVRP, por sus siglas en inglés), ventanas de tiempo (VRPTW, por sus siglas en inglés), Recogida y Entrega (VRPPD, por sus siglas en inglés), entre otros.

Ante diferentes escenarios, puede suceder que la solución entregada por las heurísticas no sea satisfactoria, con la contribución de la Inteligencia Artificial (IA), han surgido procedimientos heurísticos exitosos extendiendo la aplicación de los resultados. Obteniendo tanto técnicas como recursos computacionales específicos, como estrategias generales para construir algoritmos, que quedan por encima de las heurísticas, y van algo más allá, denominándose metaheurísticas. Permitiendo resolver un tipo de problema computacional general, las principales metaheurísticas se refieren a métodos de relajación, procesos constructivos, búsquedas por entornos y procedimientos evolutivos.

1.2. SimpliRoute

SimpliRoute es una empresa fundada en el año 2014 dedicada a la optimización e innovación tecnológica aplicada a la logística y el desplazamiento urbano. La misión de SimpliRoute es transformar las operaciones logísticas con inteligencia e innovación, entregando soluciones accesibles que lleven a nuestros clientes al siguiente nivel. SimpliRoute hasta la fecha posee más de 900 clientes con presencia en 26 países. Ha trabajado tanto con clientes grandes como pequeños de los cuales destacan Cencosud, Falabella, Liverpool y Walmart.

SimpliRoute permite reducir hasta un 30 % los costos logísticos, un 80 % los tiempos de ruteo y un 30 % las emisiones de CO_2. A través de su modelo de negocio Software as a Service (SaaS) SimpliRoute optimiza millones de entregas anuales, siendo la plataforma número 1 de logística de última milla en Latinoamérica. Acercándose más a su visión en ser referentes de inteligencia logística e innovación, con el mejor equipo a nivel global. Otras alternativas en el mercado son Beetrack, Drivin, Onfleet, Circuit, entre otros.

SimpliRoute no solamente se encarga de resolver el VRP, además incorpora un sistema de seguimiento de la flota de vehículos y mantiene informado al cliente respecto al estado de su pedido. La principal fortaleza son sus heurísticas que combinando inteligencia artificial les permite mejorar continuamente la planificación de las rutas ante nuevas instancias. Dentro de las cuales se encuentran:

- JDH: Java Dynamic Heuristic, una heurística de enfoque de inserción y mejora, en base del aprendizaje del tráfico en las calles, se encarga de entregar las rutas óptimas.
- Simplify: Una heurística basada en clusterización, con el fin de poder planificar las rutas a través de zonas.
- Big VRP: Una heurística enfocada en resolver instancias de gran tamaño, soportando 40.000 nodos y 400 vehículos en simultáneo. Siendo de gran utilidad para empresas de grandes operaciones.

También existen otras características que influyen en la resolución del VRP, como rutas balanceadas (Equilibrar carga de trabajo de conductores) o Beautify (Disminuir intersecciones entre rutas), que influyen en la calidad de la solución con el fin de facilitar una alternativa que satisfaga las necesidades del cliente.

Capítulo 2

Descripción del Problema u Oportunidad

2.1. Descripción de la problemática

El problema por abordar en este proyecto se enfoca en analizar el Problema de Ruteo de Vehículos (VRP). Este problema se ha podido resolver a través de distintas formulaciones de programación matemática para instancias pequeñas (cantidad limitada de clientes y vehículos). Sin embargo, la resolución de problemas de tamaño más real implica costos computacionales importantes debido a la complejidad combinatorial del problema (NP-HARD) teniendo dificultad para lograr buenas soluciones [2]. Esto se acentúa más cuando al VRP se le incorporan restricciones adicionales como capacidad de los vehículos y satisfacer ventanas horarias para las atenciones a clientes. Para resolver estos problemas, se han desarrollado una serie de heurísticas y algoritmos que permiten encontrar buenas soluciones para satisfacer los ruteos a mínimo costo. Si bien estos algoritmos permiten encontrar soluciones razonables para minimizar los costos de transporte, estos tienen importantes limitaciones en la práctica y uso diario para el ruteo y asignación de vehículos a clientes.

Dentro de las limitaciones, analizar aquellos casos en los que el método de solución no entrega soluciones eficientes, genera dudas acerca de si el método de solución es efectivo o se debe a un problema de la composición de la instancia. Desde la perspectiva en que se generan dudas acerca de la efectividad del método de solución, en la realidad puede generar inconvenientes al prestar un servicio de inteligencia logística. Por otro lado, en el caso que los datos de la instancia sea el problema, no es trivial conocer los aspectos a modificar para que pueda ser posible el ruteo.

La factibilidad de un problema de optimización es el primer paso para llegar al óptimo, donde hay un interés en identificar propiedades en la estructura del problema que terminan en el desarrollo de algoritmos eficientes para su resolución [3]. Para analizar la factibilidad de un problema de optimización, será necesario estudiar la región factible que se compone por las restricciones activas y los espacios asociados a los conjuntos de variables. Encontrar una solución factible y probar optimalidad está caracterizado por diferentes de puertas traseras de diferentes tipos y tamaños [4]. Los conjuntos de puertas traseras se basan en la identificación de propiedades estructurales de problemas combinatorios. Las puertas traseras para la satisfacción de restricciones son conjuntos de variables a las que la búsqueda sistemática puede limitarse al encontrar una solución o probar infactibilidad [4].

2.2. Situación Actual de la Empresa

Uno de los algoritmos más utilizados para ruteo de vehículos es JDH. En ocasiones el algoritmo no inserta la totalidad de los clientes de la instancia, interpretando la solución final propuesta como infactible y con la problemática de no visitar ciertos clientes requeridos en la instancia. La infactibilidad de SimpliRoute tiene varios orígenes o causas.

Primero hay que considerar el error humano, donde se encuentran problemas en el procesamiento de los nodos de la instancia. El primer caso consiste en geolocalizar erróneamente uno o varios nodos/clientes de la instancia. Por ejemplo, si ruteo clientes de Santiago, y aparece un cliente en Arica por falta de precisión al definir la ubicación a visitar. Considerando los tiempos de viaje es imposible realizar una visita entre Santiago y Arica en una condición laboral de trabajo durante un día. Para este tipo de situaciones, SimpliRoute ya contempla un filtro de nodos en su herramienta, con tal de no considerar puntos atípicos en la instancia, aunque el usuario podría no entender en qué aspecto (Demanda, Ubicación, entre otros) uno o más nodos son considerados atípicos.

Otro caso de error humano consiste en un uso indebido de la plataforma. El operador define una gran cantidad de clientes en la instancia conjunto con una limitada capacidad de respuesta vehicular. Por ejemplo, supongamos que se definen 200 nodos/clientes a visitar durante un día, y se tiene una flota de 3 vehículos, cada vehículo puede visitar en promedio 20 nodos/clientes por día, en base a las condiciones de viaje, tiempo de servicio, entre otros. Esto implicaría que se podría visitar máximo 60 nodos/clientes de la instancia dejando fuera a otros 140 nodos/clientes. Ante esta situación el operador debe modificar el problema, eliminando nodos/clientes a visitar o aumentando la flota de vehículos. Esto genera una molestia al operador, reclamando que la heurística no realiza un ruteo eficiente, dejando al cliente insatisfecho. Cuando en realidad la heurística hizo lo mejor que pudo dada las condiciones establecidas ante una instancia infactible. Esto señala que el operador usuario esta sesgado acerca del servicio prestado y la composición de los VRP a resolver.

Dejando de lado el error humano, ante una instancia compleja, en la que no se visita la totalidad de los nodos. Existe la posibilidad de que la heurística haya caído en un óptimo local, o ante la topología de la instancia, la heurística presenta dificultades para insertar todos los puntos. Las razones pueden deberse a que los vehículos no tienen la suficiente capacidad de transportar todo lo requerido por los clientes, o no se logra visitar a todos dadas las ventanas horarias requeridas por cada uno de los clientes, entre otras. El gran problema es que este resultado de infactibilidad se obtiene una vez ejecutado el algoritmo, lo que para ciertas instancias de gran tamaño (sobre 500 nodos) significa tiempo para el cliente. Si se utiliza JDH, el tiempo en promedio para instancias de 200 nodos es de 13 segundos, si se trata de Simplify, el tiempo en promedio para instancias de 200 nodos es de 50 segundos y por último, si se ejecuta Big VRP, una instancia de 5000 nodos toma un tiempo de 5 minutos. Aquí ocurren dos problemas: primero la solución responde de manera tardía que no podrá insertar todos los vértices, y segundo, la calidad del ruteo es deficiente dado que deja nodos fuera de las rutas programadas.

Este problema de infactibilidad tiene una consecuencia negativa en la percepción de la calidad del servicio entregado por SimpliRoute a sus clientes. Esto dado que la solución de ruteo no visita a todos los clientes requeridos y por lo tanto el operador necesita readecuar manualmente la solución. Las alternativas que posee el usuario

consisten en eliminar nodos/clientes de la instancia o agregar una mayor cantidad de vehículos, implicando mayores costos en la operación de la última milla.

Respecto de la medición de este problema, la cantidad de instancias con soluciones infactibles dependen del tamaño de estas y por lo tanto de la complejidad de su solución. De acuerdo con datos de SimpliRoute, para instancias menores a 100 nodos o clientes a visitar, el 25 % de ellas no se insertan todos los clientes en las rutas, obteniendo soluciones infactibles. Para instancias de tamaño mayor a 100 nodos, la infactibilidad de las instancias aumenta hasta un 43 %.

La oportunidad está en aprender y detectar en función de la complejidad de la instancia a rutear, qué probabilidad tiene esta instancia de ser infactible. De la misma forma, predecir qué tipo de problema o restricción no se está satisfaciendo en la instancia, de modo de recomendar al operador, alguna alternativa para mejorar su experiencia en el ruteo con las heurísticas. En particular el proyecto se hará cargo de la detección anticipada de soluciones infactibles para el CVRPTW y qué alternativas existen ante este tipo de situaciones. El punto de inicio se concentrará en el VRP resuelto por JDH, aunque está el potencial de considerar el VRP más allá de la heurística utilizada.

Capítulo 3

Estado del Arte

El estado del arte está compuesto por dos tópicos. El primero corresponde al problema de ruteo de vehículos. Posteriormente se expone la aplicación de herramientas de aprendizaje automático.

3.1. Problema de Ruteo de Vehículos

La primera formulación matemática del VRP data de los años 50 [5], y sigue siendo un área de investigación muy activa, sobre todo por su importancia económica. Una gran parte de este esfuerzo investigativo se ha enfocado en su versión clásica, el Problema de ruteo de vehículos con capacidad (CVRP). El principal argumento para este foco es la suposición que los algoritmos desarrollados para este problema pueden ser extendidos a casos más complejos y realistas [6]. Estas consideraciones más realistas conllevan muchas veces a modelos de optimización multiobjetivo, modelos que consideran incertidumbre y una amplia variedad de restricciones de la vida real relacionadas a factores de tiempo y distancia, inventario, consideraciones energéticas y ambientales, etcétera. Cabe señalar que, por la complejidad y tamaño del problema, es muy difícil usar métodos exactos de programación matemática, logrando llegar a buenas soluciones mediante el uso de algoritmos, métodos aproximados o heurísticas. Una completa y actualizada revisión de los algoritmos de ruteo y sus variantes es presentada en [7].

El principal análisis de factibilidad de instancias de ruteo consiste en usar la región infactible para llegar a mejores soluciones como lo es en [8]. Dado un conjunto de soluciones infactibles, alcanzar el óptimo mediante la eliminación de ruido, que no cometa la infactibilidad de las anteriores soluciones. Manteniendo esta estrategia, en [9] se utiliza relajación lagrangiana para corroborar la factibilidad dado un conjunto de posibles soluciones encontradas mediante métodos de solución poblacionales para el VRP. En [10] se realiza una revisión de métodos de optimización inversa. Dado una solución, encontrar los parámetros del problema de optimización resuelto, con tal de comprobar que la solución entregada minimiza los costos del VRP asociado.

Otro acercamiento respecto a factibilidad de VRP es en [11], en donde se plantea un método exacto para probar factibilidad de instancias mediante constraint programming. La exploración de la región factible consiste en analizar las restricciones del problema, el comportamiento de las variables y el dominio de la instancia. El principal objetivo es identificar una solución factible al problema que puede servir como una solución inicial que puede que no sea eficiente. Si bien en este método se plantea una formulación diferente del VRP para constraint programming, este posee sus limitantes ya que solamente es eficiente para instancias de tamaño reducido.

Otra alternativa para probar factibilidad, utilizando como base el modelo de estimación en [12], el cual propone una función de aproximación de la distancia de un Travelling Salesman Problem para puntos aleatorios. Si la instancia cumple ciertos supuestos acerca de la distribución de los nodos, se podría estimar la duración total que podría compararse con el tiempo total para realizar las entregas para determinar si es factible o no. La limitante de este método se debe a que el modelo de estimación de distancia de [12] solamente se puede aplicar si la composición espacial de la instancia sigue una distribución determinada. Si bien, es un método sencillo, no contempla la infactibilidad por capacidad-demanda por lo que se aleja del CVRPTW.

3.2. Aprendizaje automático

El uso de herramientas de aprendizaje automático ha sido de gran importancia en el último tiempo. En libros y artículos como [13], [14], [15] y [16] se menciona diferentes métodos de aprendizaje supervisado acompañado de herramientas para la visualización de resultados. Algunas de estas técnicas son:

- Regresión Logística (Logistic Regression)

  La regresión logística es un algoritmo de análisis predictivo para problemas de clasificación, basado en el concepto de probabilidad. Los supuestos de regresión logística tienden a restringir la función de costo entre 0 y 1. Los valores de entrada (X) se combinan linealmente usando pesos o valores de coeficiente (conocidos como la letra mayúscula griega Beta (β)) para predecir un valor de salida (Y).

$$Y = \frac{1}{1 + \exp^{-f(X)}}$$

  Donde:

$$f(X) = \beta_0 + \beta_1 X_1 + .. + \beta_n X_n$$

  Y es la salida pronosticada, β_0 es el intercepto y $\beta_1, ..., \beta_n$ es el coeficiente para el valor de entrada (X). La hipótesis de regresión logística tiende a limitar la función de coste entre 0 y 1, mediante la función logística (ver figura 3.1).

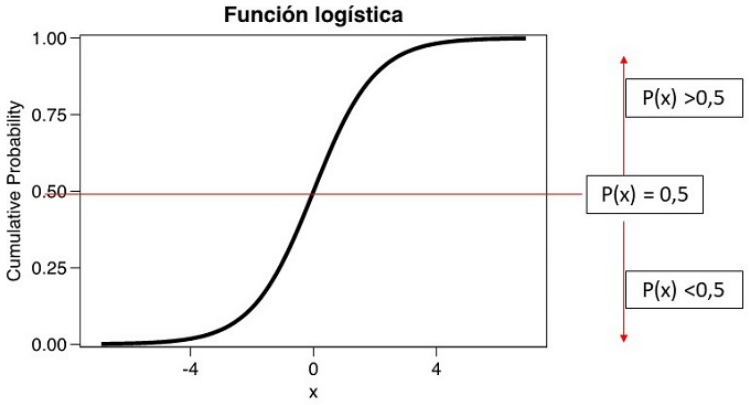

FIGURA 3.1: Función Logística

Para modelar la probabilidad de que una entrada (X) pertenezca a la clase por defecto (Y=1), podemos escribir esto formalmente como:

$$P(X) = P(Y = 1|X)$$

Hay que considerar que la predicción de probabilidad debe transformarse en valores binarios (0 o 1) para poder hacer realmente una predicción de probabilidad. La regresión logística es un método lineal, pero las predicciones se transforman utilizando la función logística. El efecto de esto es que ya no podemos entender las predicciones como una combinación lineal de las entradas como podemos con la regresión lineal, por ejemplo, el modelo se puede establecer como:

$$P(X) = \frac{1}{1 + \exp^{-(\beta_0+\beta_1 X_1+...+\beta_n X_n)}}$$

Aplicando transformación:

$$ln(P(X)/1–P(X)) = \beta_0 + \beta_1 * X_1 + ... + \beta_n * X_n$$

Se puede ver que el cálculo de la salida de la derecha es nuevamente lineal (al igual que la regresión lineal), y la entrada de la izquierda es un logaritmo de la probabilidad de la clase predeterminada.

Los coeficientes (valores Beta β) del algoritmo de regresión logística deben estimarse a partir de sus datos de entrenamiento. Esto se hace utilizando la estimación de máxima verosimilitud. Los mejores coeficientes darían como resultado un modelo que predicaría un valor muy cercano a 1 para la clase predeterminada y un valor muy cercano a 0 para la otra clase. La intuición de máxima verosimilitud para la regresión logística es que un procedimiento que busca valores para los coeficientes (valores Beta β) que minimicen el error en las probabilidades predichas por el modelo a las de los datos.

- Árboles de Decisión (Decision Tree)

  Los árboles de decisión son técnicas de machine learning que permiten la construcción de modelos predictivos basados en su clasificación según ciertas características o propiedades, o en la regresión mediante la relación entre distintas variables para predecir el valor de otra. En el análisis de decisiones, se puede usar un árbol de decisiones para representar visual y explícitamente las decisiones y la toma de decisiones.

  El árbol de decisión es una estructura que está formada por ramas y nodos de distintos tipos:

  - Los nodos internos representan cada una de las características o propiedades a considerar para tomar una decisión.
  - Las ramas representan la decisión en función de una determinada condición.
  - Los nodos finales representan el resultado de la decisión.

La importancia de las variables es clara y las relaciones se pueden ver explícitamente. Esta metodología se conoce más comúnmente como árbol de decisiones de aprendizaje a partir de datos. En general, los algoritmos del árbol de decisión se denominan CART o árboles de clasificación y regresión (*Classification and Regression Tree*). La creación de un modelo CART implica la selección de variables de entrada y puntos de división en esas variables hasta que se construya un árbol adecuado.

La selección de qué variable de entrada usar y la división o punto de corte específico se elige mediante un algoritmo codicioso (*greedy*) para minimizar una función de costo. La construcción del árbol termina usando un criterio de parada predefinido, como un número mínimo de instancias de entrenamiento asignadas a cada nodo hoja del árbol. El enfoque codicioso es un procedimiento numérico en el que todos los valores se alinean y se prueban y prueban diferentes puntos de división utilizando una función de costo. Para los problemas de modelado predictivo de regresión, la función de costo que se minimiza para elegir los puntos de división es la suma del error cuadrático en todas las muestras de entrenamiento que se encuentran dentro del rectángulo:

$$\Sigma(Y - Predicción)^2$$

Donde Y es el resultado de la muestra de entrenamiento y la *Predicción* es el resultado previsto para el rectángulo.

Para la clasificación, se utiliza la función de índice de Gini, que proporciona una indicación de cuán "puros"son los nodos hoja (cuán mezclados son los datos de entrenamiento asignados a cada nodo).

$$G = \Sigma(p_k * (1–p_k))$$

Donde G es el índice de Gini sobre todas las clases, p_k es la proporción de instancias de entrenamiento con clase k en el rectángulo de interés. Un nodo que tiene todas las clases del mismo tipo (pureza de clase perfecta) tendrá $G = 0$, mientras que un G que tiene una división de clases 50-50 para un problema de clasificación binaria (peor pureza) tendrá $G = 0,5$.

- Bosque aleatorio (Random Forest)

  Bosque aleatorio es un conjunto de algoritmos de árboles de decisión. Es una extensión de la agregación bootstrap (empaquetado - *bagging*) de árboles de decisión y se puede utilizar para problemas de clasificación y regresión.

  En el empaquetado, se crean una serie de árboles de decisión en los que cada árbol se crea a partir de una muestra de arranque diferente del conjunto de datos de entrenamiento. Una muestra de arranque es una muestra del conjunto de datos de entrenamiento donde una muestra puede aparecer más de una vez en la muestra, lo que se conoce como muestreo con reemplazo.

  El empaquetado es un algoritmo de conjunto efectivo, ya que cada árbol de decisión se ajusta a un conjunto de datos de entrenamiento ligeramente diferente y, a su vez, tiene un rendimiento ligeramente diferente. A diferencia de los modelos de árboles de decisión normales, como los árboles de clasificación y regresión (CART), los árboles utilizados en el conjunto no se podan, lo que

hace que se ajusten ligeramente al conjunto de datos de entrenamiento. Por lo general, la construcción de un árbol de decisión implica evaluar el valor de cada variable de entrada en los datos para seleccionar un punto de división. Al reducir las características a un subconjunto aleatorio que se puede considerar en cada punto de división, obliga a que cada árbol de decisión del conjunto sea más diferente. Esto es deseable ya que ayuda a que cada árbol tenga predicciones o errores de predicción menos correlacionados.

Una predicción sobre un problema de regresión es el promedio de la predicción entre los árboles del conjunto. Una predicción sobre un problema de clasificación es el voto mayoritario a favor de la etiqueta de clase en los árboles del conjunto. Cuando se promedian las predicciones de estos árboles menos correlacionados para hacer una predicción, a menudo se obtiene un mejor rendimiento que los árboles de decisión empaquetados.

El hiperparámetro más trabajado para ajustar el bosque aleatorio es la cantidad de características aleatorias que se deben considerar en cada punto de división. Otro hiperparámetro importante para ajustar es la profundidad de los árboles de decisión. Los árboles más profundos suelen estar más sobreajustados a los datos de entrenamiento, pero también están menos correlacionados, lo que a su vez puede mejorar el rendimiento del conjunto. Las profundidades de 1 a 10 niveles pueden ser efectivas.

- Aumento de Gradiente Extremo (XGBoost)

  El Aumento de Gradiente Extremo (Extreme Gradient Boosting o XGBoost), es un algoritmo de machine learning supervisado basado en un árbol de decisión que utiliza una estrategia de aumento de gradiente. El aumento de gradiente se refiere a una clase de algoritmos de aprendizaje automático de conjuntos que se pueden usar para problemas de modelado predictivo de clasificación o regresión.

  Los conjuntos se construyen a partir de modelos de árboles de decisión. Los árboles se agregan uno a la vez al conjunto y se ajustan para corregir los errores de predicción cometidos por modelos anteriores. Este es un tipo de modelo de aprendizaje automático conjunto denominado impulso. Los modelos se ajustan utilizando cualquier función de pérdida diferenciable arbitraria y algoritmo de optimización de descenso de gradiente. Esto le da a la técnica su nombre, .ªumento de gradiente", ya que el gradiente de pérdida se minimiza a medida que se ajusta el modelo. En resumen, combina un bosque aleatorio y un Gradient Boosting (GBM) para crear un conjunto de resultados mucho más preciso. XGBoost toma pasos más lentos, prediciendo secuencialmente en lugar de independientemente. Utiliza los patrones en los residuales, fortaleciendo el modelo.

- Máquinas de Vectores de Soporte (Support Vector Machines)

  El objetivo del algoritmo de las máquinas de vectores de soporte(SVM) es encontrar un hiperplano en un espacio N-dimensional (N: el número de características) que clasifique claramente los puntos de datos. Para separar las dos clases de puntos de datos, se pueden elegir muchos hiperplanos posibles. Nuestro objetivo es encontrar un plano que tenga el margen máximo, es decir, la distancia máxima entre puntos de datos de ambas clases (Ver figura 3.2).

Maximizar la distancia del margen proporciona cierto refuerzo para que los puntos de datos futuros se puedan clasificar con más confianza.

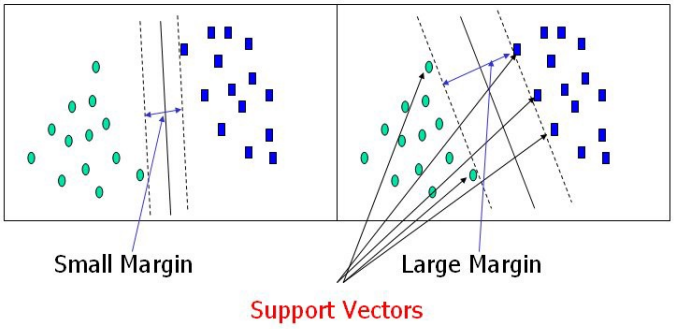

FIGURA 3.2: Representación Vectores de Apoyo

En el algoritmo SVM, buscamos maximizar el margen entre los puntos de datos y el hiperplano. La función de pérdida que ayuda a maximizar el margen es la pérdida bisagra.

$$c(x, y, f(x)) = \begin{cases} 0, & \text{Si } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{En otro caso} \end{cases}$$

El costo es 0 si el valor pronosticado y el valor real tienen el mismo signo. Si no lo son, entonces calculamos el valor de la pérdida. También añadimos un parámetro de regularización la función de coste. El objetivo del parámetro de regularización es equilibrar la maximización y la pérdida del margen. Después de agregar el parámetro de regularización, las funciones de costo se ven a continuación.

Función de pérdida para SVM:

$$min_w \, \lambda ||w||^2 + \sum_{i=1}^{n} (1 - y_i < x_i, w >)_+$$

Cuando no hay errores de clasificación, es decir, nuestro modelo predice correctamente la clase de nuestro punto de datos, solo tenemos que actualizar el gradiente desde el parámetro de regularización.

Actualización de gradiente: sin errores de clasificación

$$w = w - \alpha * (2\lambda w)$$

Cuando hay una clasificación errónea, es decir, nuestro modelo comete un error en la predicción de la clase de nuestro punto de datos, incluimos la pérdida junto con el parámetro de regularización para realizar la actualización del gradiente.

Actualización de gradiente: clasificación errónea

$$w = w + \alpha * (y_i * x_i - 2\lambda w)$$

- Redes Neuronales Perceptrón Multi-Capa (Multi-Layer Perceptron Neural Network)

  El campo de las redes neuronales artificiales a menudo se denomina simplemente redes neuronales o perceptrones de múltiples capas después de quizás el tipo de red neuronal más útil. Un perceptrón es un modelo de una sola neurona que fue un precursor de redes neuronales más grandes. Es un campo que investiga cómo se pueden usar modelos simples de cerebros biológicos para resolver tareas computacionales difíciles, como las tareas de modelado predictivo que vemos en el aprendizaje automático. El objetivo no es crear modelos realistas del cerebro, sino desarrollar algoritmos robustos y estructuras de datos que podamos usar para modelar problemas difíciles.

  Los componentes básicos de las redes neuronales son las neuronas artificiales. Estas son unidades computacionales simples que tienen señales de entrada ponderadas y producen una señal de salida usando una función de activación. Es posible que esté familiarizado con la regresión lineal, donde los pesos de las entradas son muy parecidos a los coeficientes utilizados en una ecuación de regresión. Al igual que la regresión lineal, cada neurona también tiene un sesgo que puede considerarse como una entrada que siempre tiene el valor 1,0 y también debe ponderarse. Igualmente que la regresión lineal, los pesos más grandes indican una mayor complejidad y fragilidad. Es deseable mantener los pesos en la red y se pueden utilizar técnicas de regularización.

  Las entradas ponderadas se suman y pasan a través de una función de activación, a veces llamada función de transferencia. Una función de activación es un mapeo simple de entrada ponderada sumada a la salida de la neurona. Se denomina función de activación porque gobierna el umbral en el que se activa la neurona y la fuerza de la señal de salida. Esto permite que la red combine las entradas de formas más complejas y, a su vez, proporcione una capacidad más rica en las funciones que pueden modelar. Las neuronas están dispuestas en redes de neuronas (ver figura 3.3).

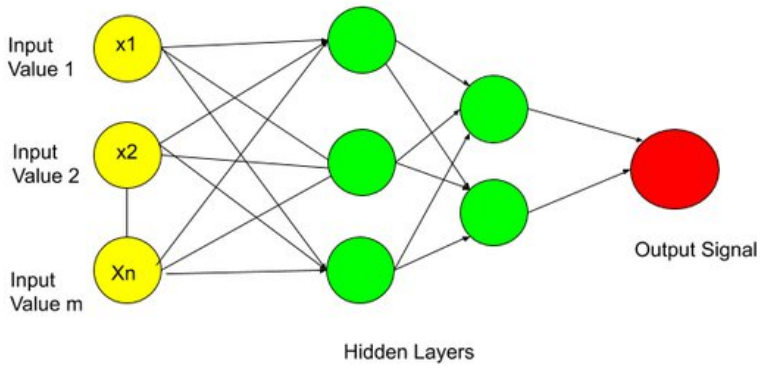

FIGURA 3.3: Representación de Redes Neuronales

Una fila de neuronas se denomina capa, y una red puede tener varias capas. La arquitectura de las neuronas en la red a menudo se denomina topología de

red. La capa inferior que recibe información de su conjunto de datos se denomina capa visible porque es la parte expuesta de la red. A menudo, una red neuronal se dibuja con una capa visible con una neurona por valor de entrada o columna en su conjunto de datos. Estas no son neuronas como las descritas anteriormente, sino que simplemente pasan el valor de entrada a la siguiente capa. Las capas posteriores a la capa de entrada se denominan capas ocultas porque no están directamente expuestas a la entrada. La estructura de red más simple es tener una sola neurona en la capa oculta que genera directamente el valor. La capa oculta final se llama capa de salida y es responsable de generar un valor o vector de valores que corresponden al formato requerido para el problema.

Una vez configurada, la red neuronal debe entrenarse en su conjunto de datos. El algoritmo de entrenamiento clásico y aún preferido para redes neuronales se llama descenso de gradiente estocástico. Aquí es donde una fila de datos se expone a la red a la vez como entrada. La red procesa la entrada hacia arriba, activando las neuronas a medida que avanza para finalmente producir un valor de salida. Esto se llama un pase hacia adelante en la red. Es el tipo de pase que también se usa después de entrenar la red para hacer predicciones sobre nuevos datos. La salida de la red se compara con la salida esperada y se calcula un error. Luego, este error se propaga a través de la red, una capa a la vez, y los pesos se actualizan de acuerdo con la cantidad que contribuyeron al error. Esta parte inteligente de las matemáticas se llama algoritmo de retro-propagación. El proceso se repite para todos los ejemplos en los datos de entrenamiento. Una ronda de actualización de la red para todo el conjunto de datos de entrenamiento se denomina época. Una red se puede entrenar para decenas, cientos o miles de épocas.

Los pesos en la red se pueden actualizar a partir de los errores calculados para cada ejemplo de entrenamiento, y esto se llama aprendizaje en línea. Puede resultar en cambios rápidos, pero también caóticos en la red. Alternativamente, los errores se pueden guardar en todos los ejemplos de entrenamiento y la red se puede actualizar al final. Esto se denomina aprendizaje por lotes y suele ser más estable. La cantidad en que se actualizan los pesos se controla mediante un parámetro de configuración denominado tasa de aprendizaje. También se denomina tamaño de paso y controla el paso o el cambio realizado en un peso de red para un error determinado.

La ecuación de actualización se puede complementar con términos de configuración adicionales que puede establecer.

- Momentum es un término que incorpora las propiedades de la actualización de peso anterior para permitir que los pesos continúen cambiando en la misma dirección incluso cuando se calcula menos error.
- La disminución de la tasa de aprendizaje se utiliza para disminuir la tasa de aprendizaje a lo largo de las épocas para permitir que la red realice grandes cambios en los pesos al principio y pequeños cambios de ajuste más adelante en el programa de entrenamiento.

- K-Medias(K-Means)

  El agrupamiento o análisis de conglomerados es un problema de aprendizaje no supervisado. A menudo se utiliza como técnica de análisis de datos para descubrir patrones interesantes en los datos, como grupos de clientes en función de su comportamiento. El algoritmo K-means identifica k número de centroides y luego asigna cada punto de datos al grupo más cercano, mientras mantiene los centroides lo más pequeños posible. Los "medios"("*means*") en K-means se refieren al promedio de los datos, es decir, encontrar el centroide.

  Lo primero es determinar el k, es decir, el número de centroides. Para determinar el k óptimo, se suele utilizar el método del codo para un rango variable de clústeres. La métrica más utilizada es el WCSS (suma de la distancia al cuadrado dentro del clúster). Una vez determinado el k, el algoritmo K-means en la minería de datos comienza con un primer grupo de k centroides seleccionados al azar, que se utilizan como puntos de inicio para cada grupo, y luego realiza cálculos iterativos (repetitivos) para optimizar las posiciones de los centroides.

- Reducción de Dimensionalidad

  Tener una gran cantidad de dimensiones en el espacio de características puede significar que el volumen de ese espacio sea muy grande y, a su vez, los puntos que tenemos en ese espacio a menudo representan una muestra pequeña y no representativa. Esto puede afectar drásticamente el rendimiento de los algoritmos de aprendizaje automático que se ajustan a los datos con muchas características de entrada, lo que generalmente se conoce como la "maldición de la dimensionalidad". Ante esto existen técnicas de reducción de dimensionalidad como:

  - Análisis de Componentes Principales (PCA)

    El análisis de componentes principales (PCA) se puede definir como la proyección ortogonal de los datos en un espacio lineal de menor dimensión, conocido como sub-espacio principal, de modo que la varianza de los datos proyectados se maximiza [14].

    PCA es un método de extracción de características que agrupa variables de una manera que crea nuevas características y permite descartar características de menor importancia. PCA es la identificación de combinaciones lineales de variables que proporcionan la máxima variabilidad dentro de un conjunto de datos. Para calcular los componentes, este método utiliza elementos del álgebra lineal (como valores y vectores propios) para determinar qué combinación daría como resultado la varianza máxima.

  - Incrustación de vecinos estocásticos con distribución-t (T-SNE)

    T-SNE es una técnica de reducción de dimensionalidad utilizada para la explotación de datos de grandes dimensiones[17]. El objetivo es determinar un espacio más de menor dimensión conservando la mínima distancia entre los grupos que más se asimilan. Una diferencia con PCA, consiste en agrupar los grupos de datos cercanos y separar los datos que son diferentes, ya que puede ocurrir que la separación de datos por PCA no siempre sea explícita.

    Se resume en 3 pasos:

1. Calcular Similitud en un espacio inicial bajo una distribución Gaussiana.

$$p_{ij} = \frac{\exp -||x_i - x_j||^2/2\sigma^2}{\sum_{k \neq l} \exp -||x_k - x_l||^2/2\sigma^2}$$

   Para cada punto x_i centramos una distribución gaussiana alrededor de este punto. Luego medimos, para cada punto x_j ($i \neq j$), la densidad bajo esta distribución gaussiana definida previamente. Finalmente, normalizamos para cada uno de los puntos. La desviación estándar está definida por la perplejidad asociada al número de vecinos alrededor de cada punto, a mayor perplejidad, mayor variación.

2. Similar a 1, Crear Espacio Dimensional Pequeño aleatorio con distribución t-Student y calcular Similitud.

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}$$

3. Para la representación en el espacio más pequeño, es necesario comparar similitudes con la medida de Kullback_Leibler. Al minimizarlo mediante la gradiente descendente se obtiene el mejor y_i posible en un espacio de dimensiones pequeñas. Esto equivale a minimizar la diferencia entre las distribuciones de probabilidad entre el espacio original y el espacio de menor dimensión.

- Interpretabilidad

  Dentro de la interpretabilidad de los modelos de aprendizaje supervisado, existen los modelos agnósticos que ayudan a comprender lo que ocurre en el proceso de la predicción de cada modelo. Algunos de ellos son:

  - LIME Local interpretable model-agnostic explanations (LIME) es un método que interpreta las predicciones individuales de modelos basándose en la aproximación local del modelo en torno a una predicción dada [18]. La técnica intenta comprender el modelo perturbando la entrada de muestras de datos y comprendiendo cómo cambian las predicciones.

    LIME modifica una sola muestra de datos ajustando los valores de las características y observa el impacto resultante en la salida. A menudo, esto también está relacionado con lo que les interesa a los humanos cuando observan el resultado de un modelo. La pregunta más común es probablemente: ¿por qué se hizo esta predicción o qué variables causaron la predicción?

    El resultado de LIME es una lista de explicaciones que refleja la contribución de cada característica a la predicción de una muestra de datos. Esto proporciona interpretabilidad local y también permite determinar qué cambios de características tendrán el mayor impacto en la predicción.

  - Shapley Value

    Shapley Value o valores de Shapley, un método de la teoría de juegos de coalición, nos dicen cómo distribuir equitativamente el "pago.entre las características. El valor de Shapley para cada variable (pago) básicamente está tratando de encontrar el peso correcto de modo que la suma de todos

los valores de Shapley sea la diferencia entre las predicciones y el valor promedio del modelo.

Los valores de Shapley son características importantes para los modelos lineales en presencia de multicolinealidad[18]. El valor de Shapley permite explicaciones contrastivas. En lugar de comparar una predicción con la predicción promedio de todo el conjunto de datos, puede compararla con un subconjunto o incluso con un solo punto de datos. Esta contrastividad también es algo que los modelos locales como LIME no tienen. Sin embargo, el valor de Shapley devuelve un valor simple por característica, pero no un modelo de predicción como LIME. Esto significa que no se puede usar para hacer declaraciones sobre cambios en la predicción de cambios en la entrada.

Dentro de las aplicaciones de machine learning en problemas de optimización tenemos el estado del arte presentado por [19]. Este se enfoca en la importancia de diferentes técnicas de aprendizaje automático para la resolución de problemas de optimización combinatoriales mediante metaheurísticas. [20] usan machine learning para aprender de las relaciones entre características cruciales de un TSP y el rendimiento del método de solución. Esto permitió categorizar los métodos de resolución seleccionando el mejor algoritmo según las componentes del TSP a resolver, también se hace mención a que se puede proyectar para el VRP.

Para poder anticiparse, es necesario incluir métodos de aprendizaje automático, en [21], el uso de árboles de decisión ayudaron a reducir el tiempo de búsqueda de soluciones de VRP y variantes en una metaheurística. A partir de las soluciones ruteo, utilizaron métricas propuestas en [22] que se dividen en características de la composición de las rutas y características de la instancia. Concepto que también en [23] utilizan para realizar segmentación para configuración automática de algoritmos utilizando características asociadas a la instancia.

Capítulo 4

Objetivo General y Objetivos Específicos

4.1. Objetivo General

Diseñar una herramienta capaz de evaluar y explicar factibilidad de instancias para el problema de ruteo de vehículos usando métodos de analítica prescriptiva y optimización entera.

4.2. Objetivos Específicos

- Desarrollar un modelo de optimización capaz de evaluar la factibilidad de instancias de ruteo.
- Realizar una comparativa de diferentes modelos de aprendizaje automático capaces de predecir factibilidad de instancias de ruteo en un tiempo menor a la resolución del VRP.
- Analizar el rendimiento de la heurística en base a la composición del VRP.
- Entregar una explicación asociada a la infactibilidad de una instancia.

Capítulo 5

Metodología

Para cumplir los objetivos planteados, la metodología a utilizar estará basada en Cross Industry Standard Process for Data Mining (CRISP-DM). El motivo detrás de esta metodología por otras está fundamentada en [24], demostrando ser más completo por su flexibilidad y adaptabilidad al negocio para proyectos de Ciencia de Datos.

5.1. Entendimiento del Negocio

SimpliRoute necesita diseñar una herramienta capaz de evaluar y explicar factibilidad de instancias para el problema de ruteo de vehículos. Mediante modelos de aprendizaje automático, se busca mejorar la experiencia de ruteo para los usuarios de SimpliRoute. Ante instancias de las que no se pueden insertar todos los nodos, es necesario corroborar la totalidad de la inserción de los nodos. Por lo que nos interesa, analizar la factibilidad de la instancia, verificar que no haya problema con la heurística, y si no se pueden visitar todos los puntos, qué alternativas tiene el usuario para abordar el problema de ruteo de vehículos.

Dada los parámetros de la instancia, responder ante la duda de si es factible o no. Con el fin de ahorrar tiempos de cálculo en instancias que no se podrán rutear todos los nodos. Como se mencionó anteriormente, se utiliza una heurística para el ruteo, y existe la posibilidad de que no inserte todo cuando en términos de factibilidad si es posible incluir todo, este concepto se asociará como infactible para SimpliRoute. En otras palabras, infactible de SimpliRoute es un infactible local, mientras que la infactibildad de la instancia es infactible global. Ante las instancias infactibles, se necesitan alternativas y/o recomendaciones dado que no se podrá insertar todos los puntos en las rutas. Por lo tanto se necesita una serie de categorías en las que se puedan clasificar las instancias infactibles, con el motivo asociado a las características de la instancia. Profundizando aún más que en el hecho de quitar nodos o agregar más vehículos.

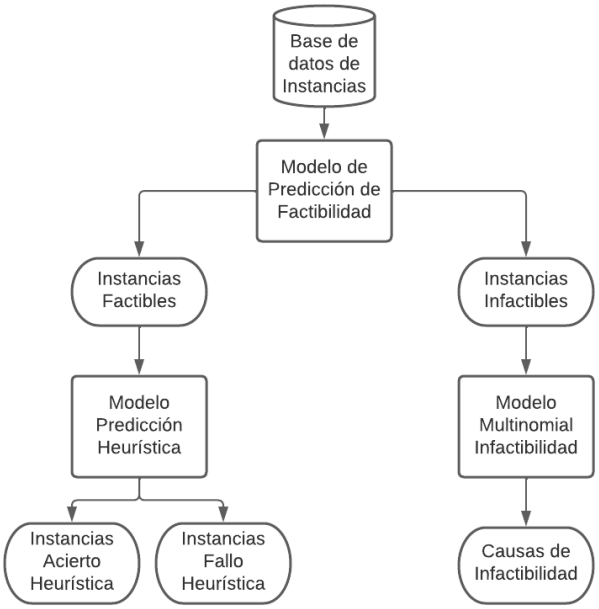

FIGURA 5.1: Entendimiento del Negocio

En la literatura y en la práctica, el enfoque está en encontrar el óptimo más que en probar factibilidad. Ya que, si no se inserta todo, el planificador debe tomar la decisión de elegir que nodos dejar fuera o aumentar la cantidad de vehículos. Pero ante un servicio de logística, una mayor profundización de la infactibilidad permitirá justificar que la dificultad se encuentra en la instancia, no en la heurística.

En la figura 5.1 aparece cada etapa para cumplir los objetivos planteados. La utilidad de estos modelos se concentra en simplificar la experiencia al usuario. Para factibilidad hay dos opciones, utilizar las instancias que he enfrentado SimpliRoute en un período de tiempo, en el que habrá que corroborar su factibilidad. Ya con el conjunto de datos de instancias factibles, podremos identificar el mejor modelo predictivo para detectar factibilidad, y ante instancias infactibles, definir el mejor modelo que pueda explicar los diferentes motivos por la que no se pueda rutear en su totalidad.

5.2. Comprensión de los Datos

Se utilizaron dos conjuntos de datos, el primero considera una muestra de las instancias resueltas por SimpliRoute en cierto período de tiempo. En su totalidad son 665 instancias, las cuales están condicionadas por sus clientes, en la que la localización de la mayoría se concentra en Chile y México. Poseen un mínimo de 2 vehículos (porque si no sería un Travelling Salesman Problem), hasta un máximo de 100, y respecto a cantidad de nodos, el mínimo es de 60, hasta un máximo de 200 nodos. Hay 237 instancias con inserción total, mientras que en 428 instancias hubo un ruteo incompleto por parte de la heurística.

La información que se maneja de las instancias es definida por el propio usuario, en lo que se encuentra el par de coordenadas para geolocalizar cada nodo perteneciente a la instancia, incluyendo el depósito, la ventana horaria establecida por cada visita, la demanda asociada que esta contempla hasta 3 tipos de productos, el tiempo de servicio estimado establecido por el propio usuario. La información principal de los vehículos consiste en el inicio y término de la jornada laboral, y la capacidad

asociada para los 3 tipos de productos. Adicionalmente, SimpliRoute contempla su propio estimador de los tiempos de viaje entre nodos, por lo que, por medio del código de la respuesta, se puede extraer la matriz asociada a los tiempos de viaje.

La segunda muestra contempla la biblioteca de instancias de ruteo de Gehring & Homberger (G&H) usadas en [25], en el que ya se conoce su factibilidad. El tamaño de esta muestra es de 300 instancias, de las cuales hay 150 instancias factibles y las otras 150 son infactibles. El tamaño de la instancia va desde 100 a 1000 nodos, y de 3 a 100 vehículos. Una característica que determina este conjunto de instancias es la distribución de los nodos, de las cuales hay 6 categorías. Tanto la primera como la segunda categoría, es una clase de problema, en el que los nodos están clusterizados por diferentes métodos. La tercera y cuarta categoría, los nodos están distribuidos de forma aleatoria por diferentes semillas iniciales. Por último, la quinta y sexta categoría es una mezcla entre aleatoriedad y clusterización de los nodos.

La información de las instancias de G&H es similar a las instancias de SimpliRoute. Una de las diferencias consiste en que en G&H sólo se contempla un tipo de producto a diferencia de SimpliRoute que puede ser multi-producto. La otra diferencia, consiste en la jornada laboral de los vehículos, que es de carácter homogéneo, definido por el propio centro de abastecimiento, mientras que en SimpliRoute puede ser heterogéneo. En este caso, hay que calcular los tiempos de viaje, que para efectos de esta investigación será equivalente a la distancia euclidiana entre ambos puntos. Para efectos de la investigación, se le dará mayor prioridad al conjunto de instancias de SimpliRoute, el conjunto de G&H cumplirá la función de validar el modelo de detección de factibilidad para instancias de mayor tamaño.

5.3. Preparación de los Datos

Dado el entendimiento del negocio, es necesario construir más de una base de datos. Primero hay que elaborar el conjunto de datos asociado a predecir factibilidad, del cual será necesario utilizar el modelo de programación de restricciones de [11]. Con este modelo se espera obtener la variable dependiente asociada a la factibilidad de la instancia. Dado la comprensión de los datos se aplicará constraint programming a un *Capacitated Vehicle Routing Problem with Time Windows*(CVRPTW) extendido a un *Vehicle Routing Problem with Multiple Products, Heterogeneous Fleet and Time Windows*(VRPMPHETW) con los siguientes conjuntos y parámetros:

- Conjuntos:

  $C = \{1, ..., N\}$ Conjunto de clientes

  $V = \{0, 1, ..., N\}$ Conjunto de vértices

  $A = \{(i, j) : 0 \leq i, j \leq N, i \neq j\}$ Conjunto de Arcos

  $K = \{1, ..., M\}$ Conjunto de vehículos

  $P = \{1, ..., p\}$ Conjunto de productos

- Parámetros:

  q_{ip}: Demanda del cliente i del producto p

  Q_{kp}: Capacidad del vehículo k para el producto p

  s_i: Tiempo de servicio del cliente i

t_{ij}: Tiempo de viaje entre el nodo i y el nodo j

O_k: Jornada de inicio del Vehículo k

R_k: Jornada término del vehículo k

e_i: Hora más temprana para servir al cliente i

l_i: Hora más tardía para servir al cliente i

Una vez determinada la variable dependiente de factibilidad, es necesario construir el conjunto de variables predictivas que están relacionadas con la factibilidad. Utilizando como base los artículos de [22] y [23] se proponen las siguientes variables:

- N: Número de clientes.
- M: Número de vehículos.
- Distprom_cent: Distancia promedio al centroide de la instancia.

$$Distprom\_cent = \frac{\sum_{i \in C} Distancia\{i, Centroide\}}{N}$$

- DistProm_depots: Distancia promedio al almacén de la instancia.

$$Distprom\_depots = \frac{\sum_{i \in C} Distancia\{i, 0\}}{N}$$

- CV_DisNodos: Coeficiente de Variación de la distancia entre los nodos.

$$CV\_DisNodos = \frac{\sigma_{Distancia}}{\mu_{Distancia}}$$

- DeltaV_Prom: Ventana horaria promedio de conducción de vehículos.

$$DeltaV\_Prom = \frac{\sum_{k \in K} (|R_k - O_k|)}{M}$$

- DeltaV_Dev: Desviación Estándar de los tiempos de conducción de vehículos.

$$DeltaV\_Dev = \sqrt{\frac{1}{M} \sum_{k \in K} (|R_k - O_k| - \mu_{jornada})^2}$$

- TsProm: Tiempo de Servicio promedio.

$$TsProm = \frac{\sum_{i \in C} s_i}{N}$$

- TvProm: Tiempo de viaje estimado promedio.

$$TvProm = \frac{\sum_{i \in V} \sum_{j \in \triangle^+ i} t_{ij}}{N^2}$$

- CxV: Ratio cantidad de nodos-cantidad de vehículos.

$$CxV = \frac{N}{M}$$

- R_Prod0: Ratio Demanda-Capacidad promedio producto Primario.

$$R\_Prod0 = \frac{\frac{\sum_{i \in C} q_{i1}}{N}}{\frac{\sum_{k \in M} Q_{k1}}{M}}$$

- R_Prod1: Ratio Demanda-Capacidad promedio producto Secundario.

$$R\_Prod1 = \frac{\frac{\sum_{i \in C} q_{i2}}{N}}{\frac{\sum_{k \in K} Q_{k2}}{M}}$$

- R_Prod2: Ratio Demanda-Capacidad promedio producto Terciario.

$$R\_Prod2 = \frac{\frac{\sum_{i \in C} q_{i3}}{N}}{\frac{\sum_{k \in K} Q_{k3}}{M}}$$

- R0_Total: Ratio Demanda-Capacidad total producto Primario.

$$R\_Prod0 = \frac{\sum_{i \in C} q_{i1}}{\sum_{k \in K} Q_{k1}}$$

- R1_Total: Ratio Demanda-Capacidad total producto Secundario.

$$R\_Prod1 = \frac{\sum_{i \in C} q_{i2}}{\sum_{k \in K} Q_{k2}}$$

- R2_Total: Ratio Demanda-Capacidad total producto Terciario.

$$R\_Prod2 = \frac{\sum_{i \in C} q_{i3}}{\sum_{k \in K} Q_{k3}}$$

- Delta_prom: Diferencia promedio de las Ventanas Horarias (Tiempo final - Tiempo de Inicio).

$$Delta\_Prom = \frac{\sum_{i \in C} |l_i - e_i|}{N}$$

- RTW_Prom: Ratio de Ventana Horaria promedio de los vehículos dividido entre la suma de los tiempos de viaje promedio con el tiempo de servicio promedio.

$$RTW\_Prom = \frac{\frac{\sum_{k \in K} |R_k - O_k|}{M}}{(TsProm + TvProm)}$$

- RTW_Total: Ratio de Ventana Horaria total de los vehículos dividido entre la suma de los tiempos de viaje promedio por nodo con el tiempo de servicio del

nodo.

$$RTW\_Total = \frac{\sum_{k \in K} |R_k - O_k|}{(\sum_{i \in C} s_i) + (\frac{\sum_{i \in V} \sum_{j \in \triangle^+ i} t_{ij}}{N})}$$

- W_Servicio: Peso dedicado al tiempo de servicio total sobre la totalidad de la instancia.

$$W\_Servicio = \frac{\sum_{i \in C} s_i}{\sum_{k \in K} |R_k - O_k|}$$

- W_Viaje: Peso dedicado al tiempo de viaje sobre la totalidad de la instancia.

$$W\_Viaje = \frac{\frac{\sum_{i \in V} \sum_{j \in \triangle^+ i} t_{ij}}{N}}{\sum_{k \in K} |R_k - O_k|}$$

- R_Tvmax: Ratio del tiempo de viaje estimado entre el depósito con el nodo más lejano y el tiempo de conducción promedio de los vehículos.

$$R\_Tvmax = \frac{Max_{j \in C}\{t_{0j}\}}{\frac{\sum_{k \in K} |R_k - O_k|}{M}}$$

Una vez definido el conjunto de variables, hay que destacar que las variables de *RTW_Prom* y *RTW_Total* es una analogía del modelo de estimación de tour de [12]. A las variables predictivas se les aplicará transformación, específicamente la estandarización de los datos para que la diferencia entre cada individuo sea representativa. Una vez transformados los datos, se aplicará una partición aleatoria de los datos en conjuntos de entrenamiento y prueba. Consecutivamente habrá una variable dependiente con el término de Heurística, la cual consiste en si la heurística fue capaz de insertar o no todos los nodos ante instancias factibles.

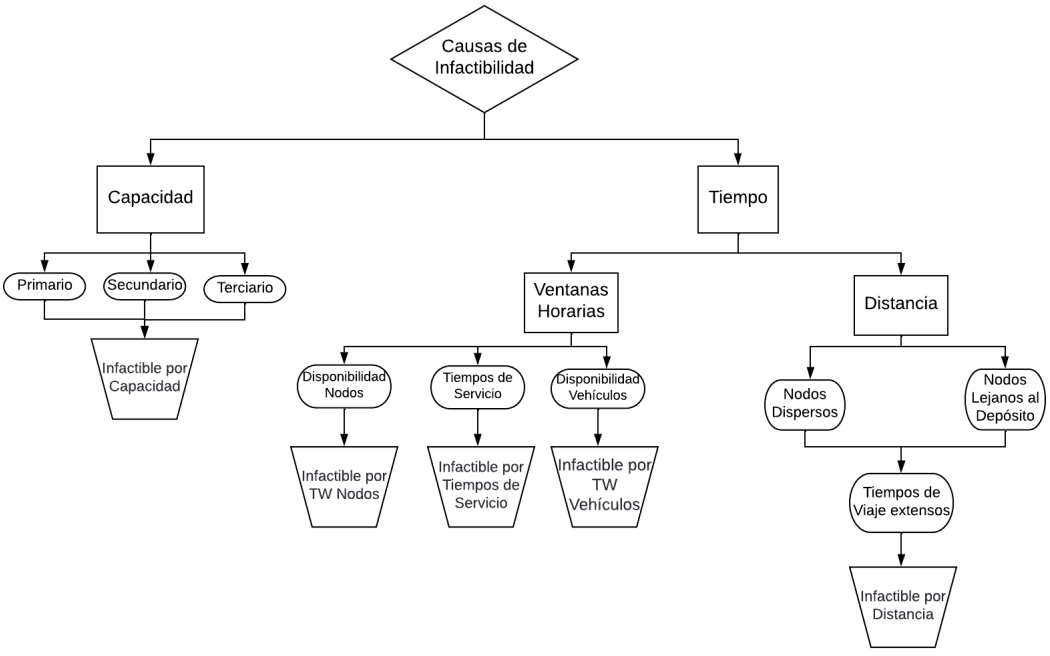

FIGURA 5.2: Causas de Infactibilidad

Para la construcción del dataset de infactibilidad, es necesario filtrar el dataset de factibilidad, enfocándose en las instancias infactibles. En base a la formulación del VRP, se estableció las categorías en el esquema de la figura 5.2. Además de las categorías propuestas en la figura, también habrá una categoría con la etiqueta Indefinido, en la cual incluirá las instancias de las cuales no pertenezcan a ningún grupo, y haya que hacer una revisión particular a las características de la instancia. Por lo tanto, las clases de las variables dependientes a trabajar son las siguientes:

- : Factibilidad: "Factible" e "Infactible".
- : Heurística: "Acierto" y "Fallo".
- : Infactibilidad: "Infactible por Capacidad", "Infactible por Ventana Horaria Nodos", "Infactible Tiempos de Servicio", "Infactible Ventana Horaria Vehículos", "Infactible Distancia" e "Infactible Indefinido".

5.4. Modelado

Para las instancias de SimpliRoute, en las que es necesario determinar la factibilidad de la instancia, se utilizará un modelo de constraint programming en base de [26] [27] con la siguiente formulación:

- Variables:

$$X_{ijk} = \begin{cases} 0, & \text{Si el arco } (i,j) \text{ es visitado por el vehículo } k \\ 1, & \text{En otro caso} \end{cases}$$

a_{ik}: Tiempo de llegada del vehículo k al cliente i

w_{ik}: Tiempo de espera del vehículo k al cliente i

- Restricciones:

$$\sum_{k \in K} \sum_{j \in V} X_{ijk} \leq 1 \qquad \forall\, i \in C \tag{5.1}$$

$$\sum_{j \in C} X_{0jk} \leq 1 \qquad \forall\, k \in K \tag{5.2}$$

$$\sum_{i \in C} X_{i0k} \leq 1 \qquad \forall\, k \in K \tag{5.3}$$

$$\sum_{i \in V} X_{ijk} = \sum_{i \in V} X_{jik} \qquad \forall\, j \in C \tag{5.4}$$

$$\sum_{i \in C} q_{ip} \sum_{j \in \triangle^+ i} X_{ijk} \leq Q_{kp} \qquad \forall\, k \in K, \forall\, p \in P \tag{5.5}$$

$$X_{ijk}(a_{ik} + w_{ik} + s_i + t_{ij} - a_{jk}) = 0 \qquad \forall\, k \in K, \forall\, i,j \in A \tag{5.6}$$

$$a_{ik} \leq l_i \sum_{j \in \triangle^+ i} X_{ijk} \qquad \forall k \in K, \forall\, i \in C \tag{5.7}$$

$$e_i \sum_{j \in \triangle^+ i} X_{ijk} \leq a_{ik} + w_{ik} \leq l_i \sum_{j \in \triangle^+ i} X_{ijk} \qquad \forall\, k \in K, \forall\, i \in C \tag{5.8}$$

$$O_k \leq a_{0k} \leq R_k \qquad \forall\, k \in K \tag{5.9}$$

$$w_{ik} \geq 0 \qquad \forall\, k \in K, \forall\, i \in C \tag{5.10}$$

$$a_{ik} \geq 0 \qquad \forall\, k \in K, \forall\, i \in C \tag{5.11}$$

$$X_{ijk} \in \{0, 1\} \qquad \forall\, k \in K, \forall\, i, j \in A \tag{5.12}$$

X_{ijk} Corresponde a la variable de decisión, mientras que a_{ik} y w_{ik} actúan como variables acumulables. En (5.1), (5.2), (5.3), (5.4) Habla de consistencia respecto a ingreso y salida de cada nodo dependiendo del arco activ. (5.5) es la restricción de abastecimiento de demanda y capacidad según vehículo y tipo de producto (En el caso que solamente haya un tipo de producto, la demanda con $p \neq 1$ equivale a 0 y la capacidad de cada vehículo con $p \neq 1$ equivale a 1). (5.6) es una restricción que actualiza el valor de los tiempos de salida y llegada a cada nodo acorde a los tiempos de viaje, servicio y espera. (5.7) y (5.8) indican las ventanas horarias de cada nodo incluyendo los tiempos de espera. En (5.9) se menciona las ventanas horarias de los vehículos. Por último, en (5.10), (5.11) y (5.12) son la naturaleza de las variables. El software de constraint programming a utilizar corresponde a Gurobi.

Una vez aplicado el modelo de constraint programming, se establecen los siguientes modelos de aprendizaje automático [15] [28] enfocado en predecir factibilidad:

- Regresión Logística [LR]
- Árbol de Decisión [DT]
- Bosque Aleatorio [RF]
- Aumento de Gradiente Extremo [XGB]
- Máquinas de Vectores de Soporte [SVM]
- Red Neuronal Perceptrón Multicapa [MLP]

A esta comparativa de modelos se hará de dos maneras: con y sin eliminación de variables (eliminación de variables representada por [Modelo_S]). Ante una eliminación generalizada del ruido, analizar como maneja cada modelo el ruido al momento de predecir factibilidad. Esta misma comparativa se aplicará para analizar la precisión de la heurística, cambiando la variable factibilidad por la variable heurística.

Para el caso de causas de infactibilidad, será necesario determinar la etiqueta establecida en preparación de los datos. Por lo que a partir de modelos de agnósticos (interpretabilidad), clusterización y reducción de dimensionalidad, se determinará el tipo de infactibilidad para cada instancia. Los modelos por utilizar consisten:

- K-Means y Análisis de Componentes Principales
- T-SNE
- LIME
- Shapley Value

La aplicación de una técnica de clusterización como K-Means, servirá para identificar tipos de instancia. Los resultados serán visualizados a través del análisis de componentes principales que serán comparados con las categorías de infactibilidad propuestas en la figura 5.2. T-SNE cumplirá la función de encontrar grupos de infactibilidad similares. A través de la similitud visualizadas en dimensiones reducidas complementará lo realizado en el modelo anterior. El uso de modelos agnósticos como LIME y Shapley Value cumplirán el rol de clasificar la instancia infactible según las variables que más influyen al momento de predecir factibilidad (en el cuadro 5.1 aparece la relación de categoría y variables). Al extraer las variables más importantes de los modelos agnósticos se clasificará dependiendo de la categoría que más representa la instancia infactible, ya que dependiendo de la complejidad de la instancia podría pertenecer a más de una categoría. En caso de que las categorías no coincidan, se analizará la distribución de los datos por medio del formato condicional para elegir la categoría más representativa.

Causa (Etiqueta)	Sigla	Variables
Infactible por Capacidad	I_Cap	R_Prod0 R_Prod1 R_Prod2 R0_Total R1_Total R2_Total
Infactible por Distancia	I_Dis	Distprom_cent DistProm_depots CV_DisNodos TvProm RTW_Prom RTW_Total W_Viaje R_Tvmax
Infactible por Tiempos de Servicio	I_Serv	TsProm RTW_Prom RTW_Total W_Servicio
Infactible por Ventana Horaria Nodos	I_TW_nodos	Delta_Prom
Infactible por Ventana Horaria Vehículos	I_TW_Veh	DeltaV_Prom DeltaV_Dev RTW_Prom RTW_Total

CUADRO 5.1: Relación variables y categoría infactibilidad

Una vez categorizadas las instancias infactibles, también habrá una eliminación de variables (formato [Modelo_S]) que, para efectos de la comparativa, al tratar con una variable dependiente multiclase, competirán solamente:

- Regresión Logística Multinomial [LR]
- Bosque Aleatorio Multinomial [RF]
- Aumento de Gradiente Extremo Multinomial [XGB]
- Red Neuronal Perceptrón Multicapa Multinomial [MLP]

5.5. Evaluación

En las comparativas de factibilidad, Heurística y causas de infactibilidad las métricas más relevantes consisten en el accuracy y tiempo de ejecución. Estableciendo como separador el conjunto de datos utilizados (luego de los resultados de constraint programming) al momento de clasificar, como los datos propios de la muestra (in-sample) correspondientes al conjunto de entrenamiento y los datos fuera de la muestra (out-sample) que equivale al conjunto de prueba. En caso de empate en términos de accuracy, se diferenciará el ganador por los tiempos que toma en predecir un conjunto de datos fuera de la muestra, ya que influye directamente en el despliegue. Por último, el tiempo de entrenamiento del modelo sería el último atributo diferenciador para definir el mejor modelo dentro de la comparativa. Adicionalmente, será necesario analizar la importancia de las variables de los mejores modelos, comparar en que se diferencian los modelos ganadores entre ellos mismos.

5.6. Despliegue

Para el despliegue de los modelos utilizados, será necesario entender como incluirlos dentro de la experiencia del cliente de SimpliRoute. En lo cual, el modelo de Predicción de Factibilidad cumplirá el rol de realizar aprendizaje semi-supervisado y ante el caso de que la instancia sea infactible, se esperará a que termine de ejecutar la solicitud de ruteo para ofrecerle al cliente la opción de analizar la infactibilidad de la instancia, proceso que incluye el modelo de causas de infactibilidad realizando aprendizaje semi-supervisado al igual que en factibilidad. Luego para post-análisis, se podrían considerar el modelo de constraint programming, que quedaría fuera de la experiencia del cliente, para no aumentar la cantidad de recursos, al igual que el modelo de heurística que podría ser considerado para estudio de instancias complejas.

Capítulo 6

Desarrollo Metodológico

6.1. Probar Factibilidad

A partir del cuadro 6.1 las diferencias encontradas por el solver de Gurobi y la heurística de SimpliRoute genera 3 conjuntos de datos. El primer conjunto respecto a detectar factibilidad corresponde a los resultados obtenidos por el solver de Gurobi, es decir, 317 instancias Factibles y 348 instancias Infactibles. El segundo conjunto corresponde a analizar la heurística, y entender las condiciones factibles de las instancias que SimpliRoute no fue capaz de insertar en su totalidad los nodos. Por lo tanto, habrá 237 instancias en las que acertó factibilidad la heurística y 80 instancias en la que falló la factibilidad de la heurística. Por último, el tercer conjunto corresponde a las 348 instancias infactibles entregados por Gurobi.

		SimpliRoute		
		Factible	Infactible	Total
Gurobi	Factible	237	80	317
	Infactible	0	348	348
	Total	237	428	665

CUADRO 6.1: Resultados Constraint Programming

Los Conjuntos de datos para las siguientes secciones son las siguientes:

1. Factibilidad SimpliRoute: 317 Factibles y 348 Infactibles.
2. Factibilidad G&H: 150 Factibles y 150 Infactibles.
3. Heurística: 237 Aciertos y 80 Fallos.
4. Infactibilidad: 348 Infactibles de Categoría Indefinida.

6.2. Predecir Factibilidad

En el cuadro 6.2 se encuentran los resultados de cada modelo utilizado, en la sección 8 aparece el proceso de eliminación de variables (cuadro 8.1). El mejor modelo para predecir Factibilidad corresponde al Aumento de Gradiente Extremo (XGBoost) el cual a pesar del costo en tiempo para entrenar, el modelo posee mejor Score para instancias fuera de la muestra en un tiempo menor que el resto de los modelos e incluso la ejecución SimpliRoute.

Modelo	Variables	Tiempo(E) [seg.]	Tiempo(P) [seg.]	Score (E) [%]	Score (P) [%]
LR	22	0,9266	0,0773	88,9	93,9
RF	22	12,393	0,027	99,2	98,1
DT	22	0,652	0,01901.	91,3	90,9
XGB	22	47,317	0,0142	99,2	98,8
XGB_S	8	46,71	0,0141	99,2	96,39
SVM	22	0,651	0,0261	88,7	93,9
MLP	22	15,481	0,0169	98,2	95,7

CUADRO 6.2: Comparativa Factibilidad

		Predicción	
		Factible	Infactible
Etiquetado	Factible	77	4
	Infactible	2	83

CUADRO 6.3: Matriz de Confusión XGBoost Conjunto Prueba SimpliRoute

Métricas:

- Accuracy = % 96.39
- Precision = % 97.6
- Recall = % 95.4
- F1 = % 96.5
- Kappa = %92.8

Revisando los valores del cuadro 6.3 y las métricas de la evaluación en el conjunto de prueba, se puede observar un gran rendimiento por parte del modelo con unas predicciones balanceadas. Sin embargo, al querer probar el XGBoost ganador con instancias de mayor tamaño como las de G&H, hay una baja de rendimiento generando un error de tipo I. Por lo tanto, será necesario reentrenar el modelo combinando ambos conjuntos, una vez mezclados, se vuelve a aplicar la división en conjuntos de entrenamiento y prueba.

		Predicción	
		Factible	Infactible
Etiquetado	Factible	23	4
	Infactible	127	146

CUADRO 6.4: Matriz de Confusión XGBoost Conjunto G&H

Métricas:

- Accuracy = % 56.3
- Precision = % 53.5
- Recall = % 97.3
- F1 = % 69
- Kappa = %12.7

		Predicción	
		Factible	Infactible
Etiquetado	Factible	314	3
	Infactible	7	341

CUADRO 6.5: Matriz de Confusión XGBoost Reentrenado SimpliRoute

		Predicción	
		Factible	Infactible
Etiquetado	Factible	149	1
	Infactible	3	147

CUADRO 6.6: Matriz de Confusión XGBoost Reentrenado G&H

- Tiempo de Entrenamiento = 60.42 segundos.
- Accuracy Conjunto Entrenamiento = %99.6
- Tiempo de Predicción conjunto de Prueba = 0.022 segundos.
- Accuracy Conjunto Prueba = %95.4

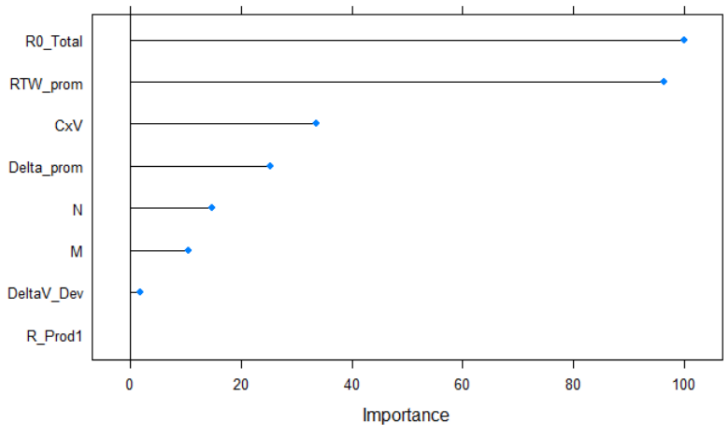

FIGURA 6.1: Importancia de las variables XGBoost Reentrenado

Con el modelo reentrenado el accuracy del conjunto de entrenamiento y de prueba es de %99.6 y %95.4 respectivamente. Adicionalmente, a partir del tiempo que toma la predicción, se puede detectar factibilidad en un menor tiempo que las heurísticas de JDH, Simplify y Big-VRP, lo que hace que el modelo se adapte a instancias de mayor tamaño. Es importante destacar que los cuadros 6.5 y 6.6 están separados por el origen de los datos. Lo cual aporta en si el modelo discrimina las instancias por el origen de los datos, pero las predicciones se ajustan para ambos conjuntos. Por otro lado, en la figura 6.1 aparece la relevancia de las variables del XGBoost Reentrenado, *R0_Total* y *RTW_Prom* predominan por el resto de las variables seleccionadas. Esto indicaría que la capacidad-demanda de los productos y la estimación de clientes a visitar poseen una mayor influencia para predecir la factibilidad de un VRP.

6.3. Factibilidad v/s Heurística

En el cuadro 6.7 se encuentran los resultados de cada modelo utilizado, en la sección 8 está disponible la selección de variables (cuadro 8.2). El mejor modelo para predecir el comportamiento de la heurística corresponde a una regresión logística, siendo el mejor en términos de accuracy. Las otras métricas también indican otras observaciones, como el *precision* en un %100 indicando solamente la presencia del error de tipo I. Adicionalmente el *Kappa* al ser sobre el %80 hay acuerdo perfecto y las clases desbalanceadas no representan ningún obstáculo.

Modelo	Variables	Tiempo(E) [seg.]	Tiempo(P) [seg.]	Score (E) [%]	Score (P) [%]
LR	22	0,741	0,148	97,5	96,7
LR_S	11	0,1521	0,0799	96,3	95,08
RF	22	5,673	0,145	98,7	91,8
DT	22	0,781	0,0234	95,08	95,08
XGB	22	31,791	0,0668	98,7	95,08
SVM	22	1,112	0,077	93,4	95,08
MLP	22	10,791	0,142	97,9	85,54

CUADRO 6.7: Resultados Comparativa Heurística

		Predicción	
		Acierto	Fallo
Etiquetado	Acierto	45	0
	Fallo	3	13

CUADRO 6.8: Matriz de Confusión Regresión Logística Conjunto Prueba SimpliRoute

Métricas:

- Accuracy = %95.08
- Precision = %100
- Recall = %93.75
- F1 = %96.77
- Kappa = %86.5

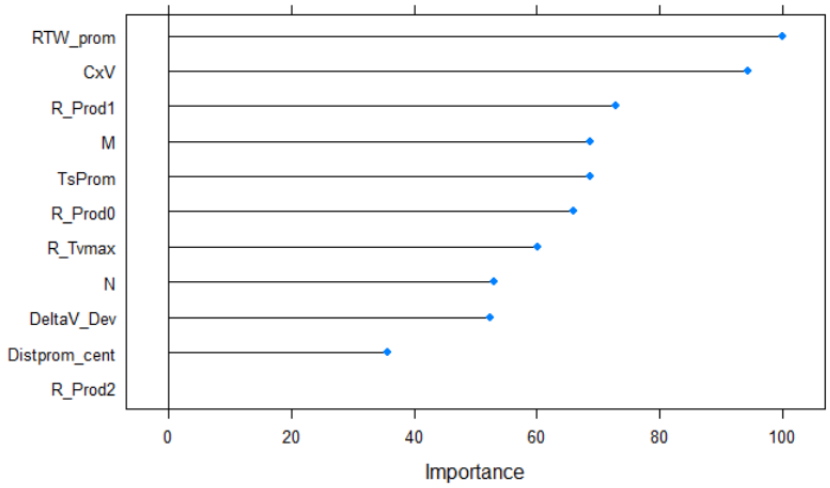

FIGURA 6.2: Importancia de las variables Regresión Logística de Heurística

En la figura 6.2 la importancia de las variables esta diversificada en comparación con las variables de predecir factibilidad (imagen 6.1). Se puede apreciar que RTW_Prom y CxV influyen en un grado ligeramente superior, lo cual podría significar que la heurística se ve afectada ante los nodos a ser visitados con los que idealmente debería visitar cada vehículo. Adicionalmente factores como demanda (R_Prod0, R_Prod1 y R_Prod2), Tamaño y Tipo de flota (M, $DeltaV\_Dev$), Tamaño y Distribución de nodos (N, R_Tvmax y $Distprom\_cent$) y por último, Tiempos de servicio ($TsProm$) tienen poder de decisión para clasificar el rendimiento de la heurística. En base a lo anterior, el rendimiento de la heurística para insertar nodos está influida por las condiciones factibles del VRP, en lo que resultaría más complejo insertar todos los nodos si se estima una baja cantidad de visitas, consecutivamente al tener una mayor cantidad de restricciones como el tipo de producto o ventanas horarias, que reducen la región factible dificultan el trabajo de la heurística.

6.4. Explicar Infactibilidad

Para poder explicar infactibilidad será necesario determinar la categoría de infactibilidad, mediante la interpretabilidad del XGBoost, el uso de modelos agnósticos como LIME y Shapley Value ayudarían a identificar la causa de infactibilidad para cada instancia. Para el caso de LIME, tomando como ejemplo la figura 6.3, es la interpretación de la predicción de la instancia de id 24, la cual es infactible. Según LIME, el XGBoost de Factibilidad también coincide en que es una instancia infactible lo cual está apoyado (support) por $R0_Total$ y CxV en mayor medida, aunque se contradice (contradicts) en cierto modo por RTW_Prom, entre otros. Luego según Shapley Value para el mismo ejemplo en la figura 6.4, todo valor negativo aporta a la infactibilidad y todo valor positivo aporta a la factibilidad. Como en ambos modelos $R0_Total$ es la variable de mayor peso en la interpretación, la instancia 24 sería categorizada como infactible por Capacidad.

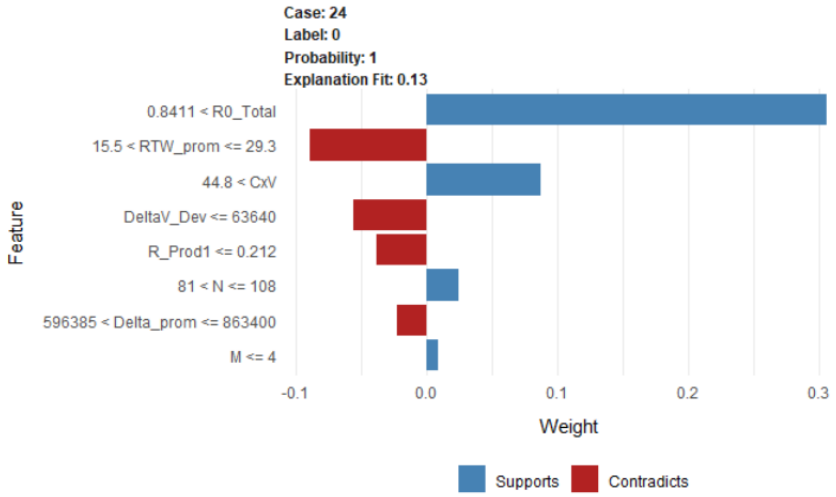

FIGURA 6.3: Lime Ejemplo

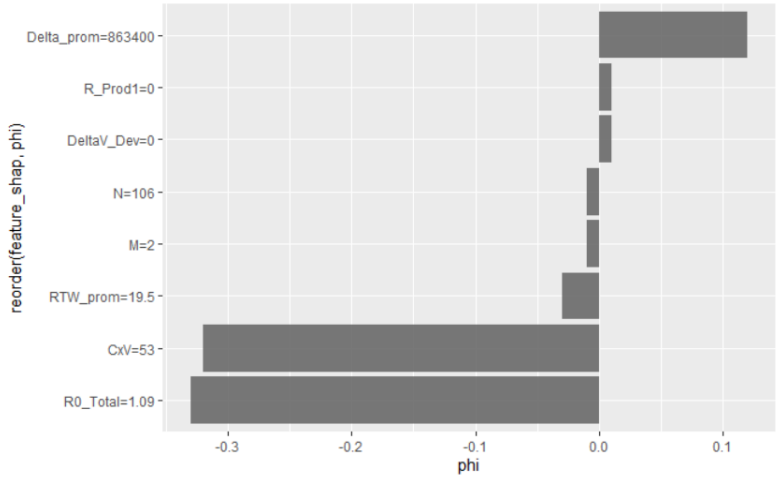

FIGURA 6.4: Shapley Value Ejemplo

Etiqueta	Conteo Previo
I_Cap	71
I_TW_nodos	25
I_TW_veh	59
I_Dis	32
I_Serv	13

CUADRO 6.9: Etiquetas Infactibilidad Inicial

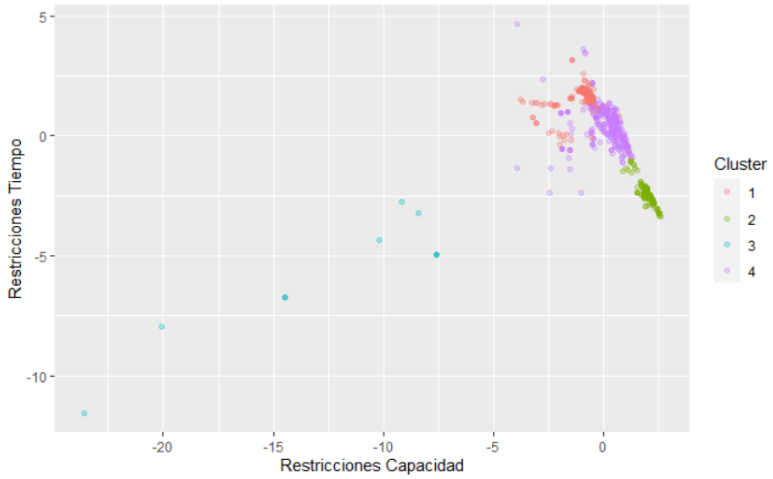

FIGURA 6.5: K-Means instancias SimpliRoute

Sin embargo, el proceso de etiquetado al ser manual podría no ser eficiente ante un elevado volumen de datos. Por lo que la búsqueda de similitudes entre VRPs mediante clusterización y técnicas de reducción de dimensionalidad servirá para agilizar el proceso. En la tabla 6.9 se indica la composición de la muestra inicial obtenida con la interpretabilidad. Luego de aplicar K-means a las instancias de SimpliRoute, se calculó el mejor $K = 4$ mediante el método Silhouette con los siguientes clústeres resultantes:

1. RTW_Prom Bajo: El primer clúster corresponde a instancias tanto factibles como infactibles siendo la última en mayor proporción, se caracteriza por tener la menor estimación de nodos a visitar por vehículo (*RTW_Prom*), Compleja Demanda-Capacidad y una flota heterogénea con la mayor dispersión en disponibilidad horaria.

2. Nodos Compactos: El segundo clúster incluye sólo instancias factibles, de las cuales se componen por tener la menor distancia hacia el centroide y almacén, adicionalmente poseen la mayor estimación de nodos a visitar por vehículo. Por lo que estaría asociado a algún cliente de SimpliRoute que realiza un servicio de mantención, donde una gran capacidad de ruteo permite visitar nodos muy cercanos entre sí. Por la visualización mediante componentes principales, son instancias que poseen los menores problemas de capacidad-demanda.

3. Outliers: El tercer clúster incluye sólo un par de instancias infactibles que corresponden a outliers con tiempos de servicios altos, los menores margenes de ventanas horarias de nodos y alta dispersión de los nodos. Por la visualización de PCA, son las instancias más alejadas del conjunto en general.

4. Variedad: El último clúster corresponde a instancias tanto factibles como infactibles en proporción balanceada, se caracteriza por ser el grupo con los tiempos de viaje más altos y nodos distantes pero no necesariamente dispersos. Al tener una proporción de factibilidad balanceada, estarían instancias incluidas de las cuales no están representadas en los otros clústeres, teniendo el resto de las características balanceadas.

En base a los resultados de K-means, se encontraron nuevas relaciones de los VRP tanto factible como infactible, pero no es suficiente para categorizar las instancias

infactibles. Al aplicar T-SNE con una perplejidad de 30 se pueden encontrar una mayor cantidad de grupos infactibles (ver figura 6.6), por lo que será necesario revisar la presencia de las etiquetas infactibles en cada grupo. Adicionalmente, T-SNE también fue capaz de detectar el clúster de nodos compactos encontrado en K-means (En la parte inferior de la figura 6.6).

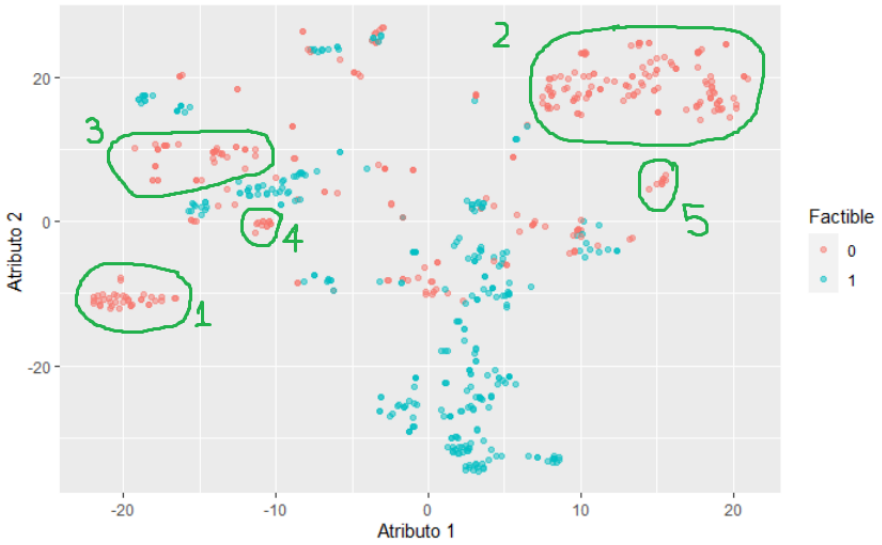

FIGURA 6.6: T-SNE SimpliRoute

Grupo	Total	Etiquetado	Falta	I_Cap	I_TW_nodos	I_TW_Veh	I_Dis	I_Serv
1	38	22	16	0	0	0	100 %	0
2	132	71	61	78.87 %	1.41 %	7.04 %	0	12.68 %
3	42	20	22	0	75 %	5 %	20 %	0
4	8	5	3	0	0	100 %	0	0
5	10	7	3	85.71 %	0	14.29 %	0	0

CUADRO 6.10: Distribución Porcentual Etiquetado Infactibilidad T-SNE

Etiqueta	Conteo Previo	Conteo Actualizado
I_Cap	71	135
I_TW_nodos	25	47
I_TW_veh	59	62
I_Dis	32	47
I_Serv	13	13

CUADRO 6.11: Etiquetas Infactibilidad Final

En la tabla 6.10 aparece la distribución porcentual de las instancias etiquetadas, si bien algunos grupos muestran más de una categoría presente, se utilizó la categoría con mayor probabilidad para asignar la etiqueta de infactibilidad. La presencia de más de una categoría puede ocurrir, pero para efectos de la investigación se concentraría en la etiqueta más representativa. Luego en 6.11 indicaría la actualización de la muestra de infactibilidad, que permitiría dar inicio a la comparativa de modelos predictivos. Las instancias infactibles que no pertenecen a alguna etiqueta serán postergadas y se mantendrán bajo la etiqueta Indefinido.

Modelo	Variables	Tiempo(E) [seg.]	Tiempo(P) [seg.]	Score (E) [%]	Score (P) [%]
LR	22	0,132	0,066	96,28	64,04
LR_S	16	0,12	0,035	96,28	58,43
RF	22	10,231	0,032	98,6	91,01
RF_S	16	8,026	0,031	98,6	89,89
XGB	22	135,63	0,077	98,6	89,89
XGB_S	16	122,61	0,034	98,6	91,01
MLP	22	13,124	0,082	93,49	84,27
MLP_S	16	11,736	0,079	93,02	88,76

CUADRO 6.12: Resultados Comparativa Infactibilidad

		Predicción				
		I_Cap	I_Dis	I_Serv	I_TW_nodos	I_TW_Veh
Etiquetado	Capacidad	39	0	0	0	1
	I_Dis	0	12	1	0	1
	I_Serv	1	0	2	0	0
	I_TW_nodos	1	0	0	13	0
	I_TW_Veh	1	1	0	1	15
	Balanced Accuracy	0.95	0.94	0.82	0.95	0.92

CUADRO 6.13: Matriz de Confusión XGBoost Conjunto de Prueba Infactibilidad

- Accuracy = %91.01
- Kappa = %87.17

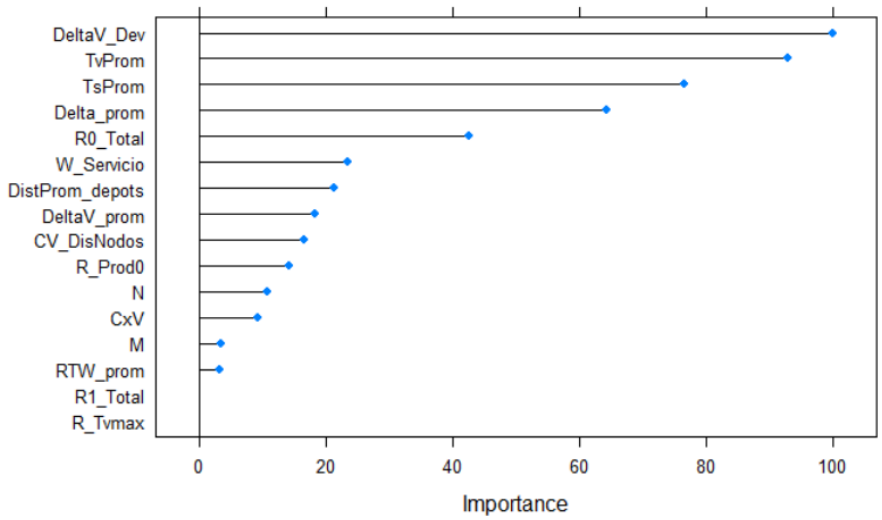

FIGURA 6.7: Importancia de las Variables XGBoost Infactibilidad

La gráfica con la eliminación de variables se encuentra en 8.1, el mejor modelo para categorizar infactibilidad corresponde a un XGBoost, ya que la evaluación del conjunto de prueba con la eliminación de variables fue la más acertada sobre el resto, llegando a rendir mejor que el modelo original con la totalidad de las variables. La matriz de confusión de 6.13 indica adicionalmente el rendimiento por clase en la última fila, siendo las mejores categorías la infactibilidad por capacidad y la infactibilidad por ventana horaria de los nodos, la peor categoría sería por tiempos

de servicio, debido a una menor presencia de esta etiqueta en comparación con las demás categorías. Sin embargo, el Kappa nos indica que la clasificación multiclase está bien representada. En la figura 6.7 aparece la importancia de las variables, de las cuales las 5 primeras representan a cada categoría.

6.5. Discusiones

En el proceso de optimización, es importante mencionar que la implementación de un método exacto como Constraint Programming no es eficiente en términos de tiempo. Por lo que no puede ser considerado dentro de la experiencia del cliente de SimpliRoute. Constraint Programming cumplió la función de analizar la factibilidad de un conjunto de instancias de VRP que ya fueron abordados. La única manera de considerar este método sería para un análisis posterior al ruteo por parte del equipo interno de SimpliRoute.

En los procesos de aprendizaje automático, para cada objetivo la composición de las variables finales fue variando, en la tabla 6.14 aparece la participación de cada variable en las diferentes etapas. De las cuales sólo 5 de ellas han participado en cada proceso, siendo la cantidad de clientes(N), el número de vehículos(M), la desviación estándar de la disponibilidad horaria vehicular ($DeltaV\_Dev$), la ratio de clientes-vehículo (CxV) y la estimación de clientes a visitar por vehículo (RTW_Prom). Variables relacionadas con la capacidad-demanda fueron variando al momento de seleccionar, ya que para una etapa podría ser mejor considerar el promedio de la capacidad-demanda, mientras que para otra es mejor considerar la totalidad de la capacidad-demanda. Las ventanas horarias son importantes para detectar factibilidad o clasificar infactibilidad, pero no posee gran influencia para analizar la efectividad de la heurística.

Variable	Factibilidad	Heurística	Infactibilidad	Total
N	1	1	1	3
M	1	1	1	3
Distprom_cent	0	1	0	1
DistProm_Depots	0	0	1	1
CV_DisNodos	0	0	1	1
DeltaV_Prom	0	0	1	1
DeltaV_Dev	1	1	1	3
TsProm	0	1	1	2
TvProm	0	0	1	1
CxV	1	1	1	3
R_Prod0	0	1	1	2
R_Prod1	1	1	0	2
R_Prod2	0	1	0	1
R0_Total	1	0	1	2
R1_Total	0	0	1	1
Delta_Prom	1	0	1	2
RTW_prom	1	1	1	3
W_Servicio	0	0	1	1
R_Tvmax	0	1	1	2
Total	8	11	16	-

CUADRO 6.14: Variables Finales utilizadas

Ante los tipos de infactibilidad, existen los siguientes puntos de mejora considerando la experiencia del ruteo, con el fin de prevenir disconformidades por parte del cliente.

- Capacidad: Analizar la opción de aumento en la capacidad vehicular. Ante un excedente de tiempo, considerar si es viable entregar los pedidos en viajes adicionales. Una posibilidad de realizar la entrega para los nodos de mayor demanda entre más de un vehículo. Por último, aumentar la flota de vehículos.
- Ventanas Horarias Nodos: El punto de inicio consiste en identificar los nodos con ventana horaria más estrecha e identificar nodos que comparten horario. Una vez reconocidos, priorizar la visita de esos nodos complejos o postergar esos nodos en una nueva instancia.
- Ventanas Horarias Vehículos: Respecto a la disponibilidad horaria de la flota ¿Se puede aumentar la jornada vehicular para uno o más vehículos? En caso de flota heterogénea ¿Puedo aumentar la jornada para los que conducen menos? Si el aumento de la jornada vehicular no es suficiente, será necesario considerar un aumento de la flota vehicular.
- Distancia: En caso de tener una alta dispersión de nodos, Segmentar el ruteo por zonas. Por otro lado, si los nodos están lejanos al almacén, identificar aquellos nodos distantes, del cual dependiendo de la necesidad del cliente, priorizar o postergar nodos distantes.
- Tiempos de Servicio: Lo primero será verificar que no haya una sobreestimación de los tiempos de servicio. De no ser el caso, identificar qué nodos pueden causar cuello-botella en el ruteo, con el fin de priorizar o postergar los nodos con tiempo de servicio alto.

Para las instancias infactibles de las cuales no pertenecen a alguna categoría (indefinido), será necesario utilizar el etiquetado manual. Considerando las alternativas para cada etiqueta, hay dos perspectivas, la primera consiste en un potencial de mejora de la heurística para poder incrementar la inserción de nodos en el ruteo por parte de SimpliRoute. La segunda perspectiva sería la del planificador que utiliza el software de ruteo, con la que tendría un punto de inicio para mejorar su experiencia en la planificación de rutas.

Capítulo 7

Conclusiones y Trabajos Futuros

7.1. Conclusiones

En el inicio de este trabajo se planteó como meta de investigación determinar la factibilidad de un VRP, a través de características que describen la composición de la instancia, y en el caso de que fueran infactibles, clasificarlos en diversas categorías. Desde el punto de vista de optimización, se utilizó un método exacto para probar la factibilidad desde un CVRPTW hasta un VRPMPHETW, por medio de un conjunto de instancias pertenecientes a SimpliRoute, del cual determino en una serie de conjuntos de datos para cumplir los diferentes objetivos de investigación.

A partir de los datos de SimpliRoute, se consiguió predecir factibilidad a través de modelos de aprendizaje supervisado. Por medio de un XGBoost, es posible determinar la factibilidad de un VRP en un menor tiempo que la ejecución de las heurísticas, con gran precisión. Sin embargo, al evaluar factibilidad en un conjunto teórico de mayor tamaño como el de Gehring & Homberger, hubo un sesgo ante las instancias de mayor tamaño, por lo que fue necesario reentrenar el modelo utilizando ambos conjuntos de datos para mantener un buen rendimiento. Luego al analizar la heurística mediante analítica prescriptiva, una regresión logística fue la mejor en predecir el comportamiento de la heurística ante instancias factibles. La capacidad de inserción por parte de la heurística de SimpliRoute está correlacionada con la región factible, ya que mientras más complejo el VRP, la heurística tendría mayor dificultad para insertar la totalidad de los nodos.

Para explicar infactibilidad, hubo que realizar una clasificación de instancias a partir de una mezcla de modelos agnósticos, clusterización y técnicas de reducción de dimensionalidad. En donde había presencia de tipos de instancias que pueden estar relacionados con los clientes de SimpliRoute, específicamente en T-SNE se pudo encontrar una mayor cantidad de grupos con características similares para agilizar la categorización de infactibilidad. Una vez categorizadas las instancias infactibles, a través de métodos de aprendizaje supervisado, se puede predecir la causa de infactibilidad asociada al VRP. Por medio de otro XGBoost, en el que las variables más importantes están diferenciadas por las categorías establecidas. Una vez clasificadas las instancias infactibles, hay que entender que la investigación se centró en la composición del VRP, más que en buscar nodos específicos que puedan alterar la factibilidad, por lo que puede haber diferentes puntos de mejora en la experiencia del ruteo.

Esta investigación entrega resultados asociados a la factibilidad de un VRP, lo cual permitiría anticiparse a la heurística y ahorrar tiempo de ejecución para instancias que resultan infactibles. Dado una instancia infactible, esta podrá pertenecer a una

categoría en la que, por medio de indicadores de la composición de la instancia, explicarle al usuario el motivo detrás de un ruteo incompleto.

7.2. Trabajos Futuros

Como Trabajos Futuros y Proyecciones, estaría en primera instancia la mezcla entre factibilidad y etiquetado de infactibilidad en un único modelo. Otra alternativa sería la automatización del etiquetado de infactibilidad para agilizar el análisis de VRP, evitando el etiquetado manual que podría ralentizar el proceso. También es importante considerar que este modelo abarca una serie de VRP's, por lo que está presente la escalabilidad y adaptación a nuevas variantes del VRP para aumentar el alcance de las predicciones. Una vez clasificadas las instancias infactibles, definir una estrategia para abordar infactibilidad con el fin de aumentar el porcentaje de inserción de instancias infactibles. Como se analizó la composición del VRP, la identificación o búsqueda de nodos complejos, podría ser un potencial que acercaría al usuario a una mejor experiencia en la planificación de rutas.

Capítulo 8

Anexos

CUADRO 8.1: Eliminación de Variables Predecir Factibilidad

	Dependent variable:		
	Factible		
	(1)	(2)	(3)
N	-0.505^{**}	-0.435^{**}	-0.336^{*}
	(0.226)	(0.206)	(0.183)
M	1.840^{**}	1.263^{**}	0.324
	(0.832)	(0.545)	(0.562)
DistProm_depots			
Distprom_cent	0.448		
	(0.479)		
CV_DisNodos	0.320	0.316	
	(0.219)	(0.221)	
R_Prod0	0.747^{**}	0.747^{**}	
	(0.313)	(0.312)	
R_Prod1	-9.022	-9.702^{***}	-12.012^{**}
	(6.174)	(3.487)	(4.711)
R_Prod2	-4.791		
	(254.416)		
R0_Total	-1.330^{***}	-1.337^{***}	-0.634^{**}
	(0.469)	(0.463)	(0.307)
R1_Total	-0.017		
	(1.572)		
R2_Total	6.619		
	(349.403)		

DeltaV_Prom			
DeltaV_Dev	-0.691^{**}	-0.786^{***}	-0.975^{***}
	(0.270)	(0.252)	(0.234)
CxV	-3.364^{***}	-3.308^{***}	-3.010^{***}
	(0.693)	(0.628)	(0.572)
Delta_prom	1.608^{***}	1.511^{***}	1.191^{***}
	(0.276)	(0.249)	(0.211)
TsProm	0.417	0.397	
	(0.363)	(0.346)	
TvProm	-0.477		
	(3.542)		
RTW_Total	-6.139^{***}	-5.004^{***}	
	(1.738)	(1.528)	
RTW_prom	12.208^{***}	10.891^{***}	8.044^{***}
	(2.119)	(1.521)	(1.017)
W_Servicio			
W_Viaje			
R_Tvmax	0.735^{**}	0.556^{**}	
	(0.286)	(0.227)	
Constant	1.605	1.073^{*}	0.411
	(25.476)	(0.636)	(0.611)
Observations	499	499	499
Log Likelihood	-109.975	-110.640	-121.330
Akaike Inf. Crit.	257.950	249.280	260.660
Note:			*p<0.1; **p<0.05; ***p<0.01

CUADRO 8.2: Eliminación de Variables Heurística

	Dependent variable:		
	Heurística		
	(1)	(2)	(3)
N	-2.623^{*}	-2.623^{*}	-1.411^{**}
	(1.347)	(1.347)	(0.708)

M	89.207*	89.207*	40.287**
	(45.522)	(45.522)	(15.995)
DistProm_depots			
Distprom_cent	−9.532**	−9.532**	−0.811
	(4.725)	(4.725)	(0.571)
CV_DisNodos	0.395	0.395	
	(0.436)	(0.436)	
R_Prod0	−19.969	−19.969	−2.878**
	(12.393)	(12.393)	(1.186)
R_Prod1	−193.044	−1,995.675	−23.334***
	(44,278.300)	(40,327.860)	(8.788)
R_Prod2	−656.851		−140.978
	(17,114.830)		(587.377)
R0_Total	2.428	2.428	
	(1.789)	(1.789)	
R1_Total	35.123	614.781	
	(13,299.300)	(124,959.800)	
R2_Total	716.768	1,088.985	
	(11,758.580)	(130,316.800)	
DeltaV_Prom			
DeltaV_Dev	−0.837	−0.837	−0.634**
	(0.690)	(0.690)	(0.321)
CxV	−6.553	−6.553	−9.218***
	(4.643)	(4.643)	(2.740)
Delta_prom	−1.085	−1.085	
	(1.159)	(1.159)	
TsProm	−15.326**	−15.326**	−7.019**
	(7.454)	(7.454)	(2.792)
TvProm	8.676**	8.676**	
	(3.928)	(3.928)	
RTW_Total	129.305*	129.305*	
	(77.497)	(77.497)	

RTW_prom	11.101*	11.101*	14.590***
	(6.039)	(6.039)	(4.106)
W_Servicio			
W_Viaje			
R_Tvmax	−3.387**	−3.387**	−2.034**
	(1.490)	(1.490)	(0.913)
Constant	36.150	17.413	−4.979
	(556.969)	(5,133.384)	(77.310)
Observations	244	244	244
Log Likelihood	−18.764	−18.764	−30.321
Akaike Inf. Crit.	75.528	73.528	84.642
Note:			*p<0.1; **p<0.05; ***p<0.01

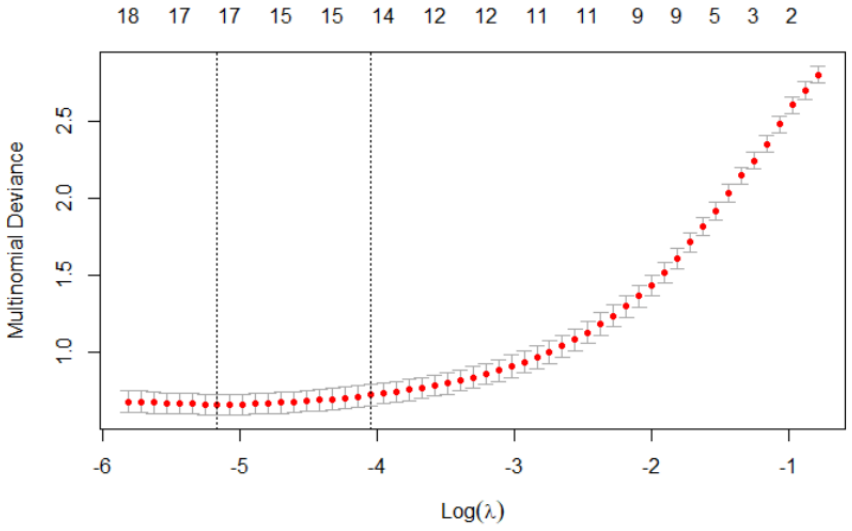

FIGURA 8.1: Resultados Lasso Multinomial

Bibliografía

[1] Paolo Toth, Daniele Vigo, *Vehicle Routing*. 2014.

[2] O. Díaz and A. CRUZ, "El problema del transporte," *Centro de Investigación en Ingeniería y Ciencias Aplicadas. Pág*, vol. 3, 2006.

[3] I. P. Gent and T. Walsh, "Easy problems are sometimes hard," *Artificial Intelligence*, vol. 70, no. 1-2, pp. 335–345, 1994.

[4] B. Dilkina, C. P. Gomes, Y. Malitsky, A. Sabharwal, and M. Sellmann, "Backdoors to combinatorial optimization: Feasibility and optimality," in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems* (W.-J. van Hoeve and J. N. Hooker, eds.), (Berlin, Heidelberg), pp. 56–70, Springer Berlin Heidelberg, 2009.

[5] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80–91, 1959.

[6] G. Laporte, "What you should know about the vehicle routing problem," *Naval Research Logistics (NRL)*, vol. 54, no. 8, pp. 811–819, 2007.

[7] A. Mor and M. Speranza, "Vehicle routing problems over time: a survey," *4OR*, vol. 18, 06 2020.

[8] Juan P. Castro, Dario Landa-Silva, José A. Moreno Pérez, "Exploring feasible and infeasible regions in the vehicle routing problem with time windows," *Nature inspired cooperative strategies for optimization (NICSO 2008)*, pp. 103–114, 2008.

[9] Daniel Guimarans, Rosa Herrero, Daniel Riera, Juan Angel A., Jose Juan Ramos, "Combining probabilistic algorithms, Constraint Programming and Lagrangian Relaxation to solve the Vehicle Routing Problem," *ANNALS OF MATHEMATICS AND ARTIFICIAL INTELLIGENCE*, vol. 62, no. 3-4, pp. 299–315, 2011.

[10] C. Heuberger, "Inverse combinatorial optimization: A survey on problems, methods, and results," *J. Comb. Optim.*, vol. 8, pp. 329–361, 09 2004.

[11] Francesca Rossi, Peter van Beek, Toby Walsh, *Handbook of Constraint Programming*, ch. 23. Elsevier, 2006.

[12] B. Çavdar and J. Sokol, "A distribution-free tsp tour length estimation model for random graphs," *European Journal of Operational Research*, vol. 243, no. 2, pp. 588–598, 2015.

[13] J. Gareth, W. Daniela, H. Trevor, and T. Robert, *An introduction to statistical learning: with applications in R*. Spinger, 2013.

[14] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[15] Trevor Hastie, Robert Tibshirani, Jerome Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2 ed., 2016.

[16] D. Janzing, L. Minorics, and P. Blöbaum, "Feature relevance quantification in explainable ai: A causal problem," 2019.

[17] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[18] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *NIPS*, 2017.

[19] Maryam Karimi-Mamaghan, Mehrdad Mohammadi, Patrick Meyer, Amir Mohammad Karimi-Mamaghan, El-Ghazali Talbi, "Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art," *European Journal of Operational Research*, p. 30, 2021.

[20] Kate Smith-Miles, Leo Lopes, "Measuring instance difficulty for combinatorial optimization problems," *Computers Operations Research*, vol. 39, no. 5, pp. 875–889, 2012.

[21] Flavien Lucas, Romain Billot, Marc Sevaux, Kenneth Sörensen, "Reducing Space Search in Combinatorial Optimization Using Machine Learning Tools," *Learning and Intelligent Optimization*, vol. 12096, pp. 143–150, 2020.

[22] F. Arnold and K. Sörensen, "What makes a vrp solution good? the generation of problem-specific knowledge for heuristics," *Computers & Operations Research*, vol. 106, pp. 280–288, 2019.

[23] J. Rasku, T. Kärkkäinen, and N. Musliu, "Feature Extractors for Describing Vehicle Routing Problem Instances," in *5th Student Conference on Operational Research (SCOR 2016)* (B. Hardy, A. Qazi, and S. Ravizza, eds.), vol. 50 of *OpenAccess Series in Informatics (OASIcs)*, (Dagstuhl, Germany), pp. 7:1–7:13, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

[24] A. Azevedo and M. F. Santos, "Kdd, semma and crisp-dm: a parallel overview," *IADS-DM*, 2008.

[25] J. Homberger and H. Gehring, "A two-phase hybrid metaheuristic for the vehicle routing problem with time windows," *European journal of operational research*, vol. 162, no. 1, pp. 220–238, 2005.

[26] J. Jiang, K. M. Ng, K. L. Poh, and K. M. Teo, "Vehicle routing problem with a heterogeneous fleet and time windows," *Expert Systems with Applications*, vol. 41, no. 8, pp. 3748–3760, 2014.

[27] P. Kabcome and T. Mouktonglang, "Vehicle routing problem for multiple product types, compartments, and trips with soft time windows," *International Journal of Mathematics and Mathematical Sciences*, vol. 2015, 2015.

[28] Faraway, J. J, *Extending the linear model with R: generalized linear, mixed effects and nonparametric regression models*. 1 ed., 2005.