Trabajo de Fin de Grado

Grado en Ingeniería Informática

Minería de datos aplicada a la predicción de flujos de tráfico

Data mining applied to traffic flow prediction

Javier Ramos Fernández

La Laguna, 31 de mayo de 2018

D. **Jesús Manuel Jorge Santiso**, con N.I.F. 42.097.398-S profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

C E R T I F I C A

Que la presente memoria titulada:

"Minería de datos aplicada a la predicción de flujos de tráfico"

ha sido realizada bajo su dirección por D. **Javier Ramos Fernández**,

con N.I.F. 45.865.421-V.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 31 de mayo de 2018

Agradecimientos

Gracias a mis padres Jose Luís y Conchi por apoyarme día a día, por el cariño que me han dado, proporcionarme el sustento necesario, ser una fuente de inspiración y soportar mis frustraciones durante el desarrollo de este trabajo.

A mi hermano mayor Eduardo por ser siempre un ejemplo a seguir y por sus consejos de incalculable valor.

A mi hermano mellizo por ser un compañero leal, honesto, trabajador y hacer que el recorrido de mi vida sea un camino lleno de alegrías.

Al resto de mi familia por hacerme pasar unos veranos inolvidables y por darme su apoyo incondicional.

A mis compañeros de clase por proporcionarme ayuda académica constantemente siempre que la he necesitado.

A mi tutor Jesús por haber confiado en mí desde el principio y por la comprensión que ha tenido conmigo en todo momento para llevar a cabo este proyecto.

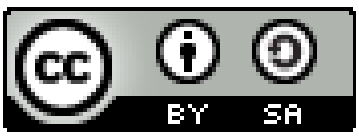
A mis amigos de toda la vida por ayudarme a evadirme de mis obligaciones y hacerme pasar muy buenos ratos.

A la gente que ha dedicado su tiempo a escuchar mis problemas y me ha animado a seguir adelante.

A todos gracias de corazón. Con paciencia y dedicación se puede conseguir todo lo que te propongas.

Javier

Licencia



Resumen

El objetivo de este trabajo ha sido realizar predicciones del tiempo promedio de viaje y el volumen de tráfico en la red de carreteras propuesta por la competición KDDCup 2017 (concretamente en ***Hangzhou****, provincia de* ***Zhejiang****, en China). Para llevar a cabo esta tarea, nos hemos basado en la utilización de técnicas de aprendizaje automático para construir modelos de predicción que estimen de la forma más precisa posible los futuros valores reales de tiempo promedio de viaje y volumen de tráfico. De esta forma, se pretende contribuir a la ejecución de medidas preventivas por parte de las autoridades de gestión del tráfico a través de la realización de predicciones fiables para el futuro flujo del tráfico.*

Las herramientas principales utilizadas para desarrollar este proyecto son el sistema de gestión de bases de datos relacional denominado ***PostgreSQL*** *(para guardar los datos suministrados) y el lenguaje de programación* ***Python*** *(para hacer uso de las librerías que contienen las técnicas de aprendizaje automático, así como otras bibliotecas de utilidad). Mediante la integración de estas dos herramientas se ha establecido un canal de comunicación entre las diferentes combinaciones de datos creadas y los algoritmos de aprendizaje automático elegidos para llevar a cabo las estimaciones oportunas y realizar evaluaciones de las mismas.*

Palabras clave: KDDCup 2017, Aprendizaje Automático, Tráfico carreteras, Tiempo Promedio de Viaje, Volumen de Tráfico, PostgreSQL, Python

Abstract

The objective of this work has been to make predictions of the average travel time and traffic volume on the road network proposed by the KDDCup 2017 competition (specifically in ***Hangzhou****,* ***Zhejiang*** *Province, China). To carry out this task, we have relied on the use of automatic learning techniques to construct predictive models that estimate as accurately as possible the real future values of average travel time and traffic volume. In this way, it is intended to contribute to the implementation of preventive measures by traffic management authorities by making reliable predictions for the future flow of traffic.*

The main tools used to develop this project are the relational database management system called ***PostgreSQL*** *(to save the supplied data) and the* ***Python*** *programming language (to make use of the libraries containing the machine learning techniques, as well as other useful libraries). Through the integration of these two tools a communication channel has been established between the different data combinations created and the chosen machine learning algorithms to carry out the appropriate estimations and evaluations of them.*

Keywords: KDDCup 2017, Machine Learning, Road Traffic, Average Travel Time, Traffic Volume, PostgreSQL, Python

Índice general

Capítulo 1 Introducción.....1
1.1 Antecedentes y estado del arte.....2
1.2 Objetivos.....4
1.3 Organización de la memoria.....4
Capítulo 2 Tecnologías.....6
2.1 Almacenamiento de datos.....6
2.2 Análisis de datos.....7
2.3 Gestión del proyecto.....8
Capítulo 3 La minería de datos.....10
3.1 Introducción.....10
3.2 Técnicas de minería de datos.....11
3.2.1 Técnicas basadas en vecindad.....12
3.2.1.1 El algoritmo K-NN básico.....12
3.2.1.2 Método de regresión basado en KNN.....14
3.2.2 Técnicas basadas en redes neuronales.....14
3.2.2.1 Entrenamiento de una red neuronal.....16
3.2.3 Técnicas basadas en árboles de decisión.....17
3.2.4 Técnicas estadísticas.....20
3.2.4.1 Componentes de las series temporales.....22
3.2.4.2 Clasificación descriptiva de las series temporales.....24

3.2.4.3 Funciones de autocorrelación y autocorrelación parcial.....26
3.2.4.4 Estimación de las componentes de una serie temporal.....26
3.2.4.5 Modelo ARIMA.....29
3.2.4.6 Fases del modelo ARIMA.....31
3.2.4.7 Regresión Lineal.....32
3.2.5 Técnicas basadas en optimización.....33
Capítulo 4 Proyecto de Minería de Datos.....35
4.1 Problema planteado por la competición KDDCup 2017.....36
4.1.1 Contexto.....36
4.1.2 Tareas.....37
4.1.3 Etapas y reglas de la competición.....37
4.1.4 Métricas de evaluación.....38
4.2 Creación de bases de datos para almacenar los datos de la competición y modificaciones.....39
4.2.1 Estructura de bases de datos propuesta.....39
4.2.2 Base de datos con los datos originales.....40
4.2.2.1 Tabla vehicle_routes.....40
4.2.2.2 Tabla road_links.....40
4.2.2.3 Tabla vehicle_trajectories_training.....41
4.2.2.4 Tabla traffic_volume_tollgates_training.....41
4.2.2.5 Tabla weather_data.....41
4.2.2.6 Tabla travel_time_intersection_to_tollgate.....42
4.2.2.7 Tabla traffic_volume_tollgates.....42
4.2.3 Base de datos con los tipos de datos modificados.....42
4.2.3.1 Tabla road_links_modified.....42
4.2.3.2 Tabla vehicle_routes_modified.....43
4.2.3.3 Tabla vehicle_trajectories_training_modified.....43
4.2.3.4 Tabla traffic_volume_tollgates_training_modified.....43
4.2.3.5 Tabla weather_data_modified.....43

4.2.3.6 Tabla travel_time_intersection_to_tollgate_modified.....44
4.2.3.7 Tabla traffic_volume_tollgates_modified.....44
4.2.4 Base de datos con los datos relacionados con la fase de pruebas....44
4.2.4.1 Tabla travel_time_intersection_to_tollgate_test.....44
4.2.4.2 Tabla traffic_volume_tollgates_test.....44
4.2.4.3 Tabla tabla_resultado_average_travel_time.....45
4.2.4.4 Tabla tabla_resultado_traffic_volume.....45
4.2.5 Base de datos con los datos reales de los valores a predecirse.....45
4.2.5.1 Tabla travel_time_intersection_to_tollgate_real.....46
4.2.5.2 Tabla traffic_volume_tollgates_real.....46
4.3 Creación de gráficas para visualizar los datos almacenados.....46
4.3.1 Gráficas del tiempo promedio de viaje de todos los días por rutas. 46
4.3.2 Gráficas del tiempo promedio de viaje de algunos días en todas las horas en cada una de las rutas.....48
4.3.3 Gráficas del volumen de tráfico de todos los días en todas las horas en cada una de los pares barrera de peaje-dirección.....49
4.4 Predicciones del tiempo promedio de viaje.....50
4.4.1 Primera aproximación: Algoritmos diversos de Machine Learning..51
4.4.1.1 Creación de las vistas minables.....52
4.4.1.2 Realización de predicciones a partir de las vistas.....56
4.4.2 Segunda aproximación: ARIMA.....60
4.4.2.1 Preparación de los datos.....60
4.4.2.2 Realización de estimaciones.....61
4.4.3 Tercera aproximación: Ventana deslizante.....63
4.4.3.1 Preparación de los datos.....63
4.5 Predicciones del volumen de tráfico.....65
4.5.1 Primera aproximación: Algoritmos diversos de Machine Learning..66
4.5.2 Segunda aproximación: ARIMA.....67
4.5.3 Tercera aproximación.....70

Capítulo 5 Conclusiones y líneas futuras.....71
5.1.1 Conclusiones.....71
5.1.2 Líneas futuras.....72
Capítulo 6 Summary and Conclusions.....73
6.1.1 Conclusions.....73
6.1.2 Future lines.....74
Capítulo 7 Presupuesto.....75

Índice de figuras

Figura 3.1: El proceso de descubrimiento de conocimiento en bases de datos (KDD) [7].....10
Figura 3.2: Algoritmo básico KNN [8].....13
Figura 3.3: Ejemplo de aplicación del algoritmo K-NN básico [8].....14
Figura 3.4: Estructura de una neurona artificial.....15
Figura 3.5: Estructura de una red neuronal.....16
Figura 3.6: Ejemplo de árbol de regresión.....18
Figura 3.7: Crecimiento level-wise del árbol en XGBoost.....19
Figura 3.8: Crecimiento leaf-wise del árbol en LightGBM.....20
Figura 3.9: Ejemplo de serie temporal.....20
Figura 3.10: Ejemplo de una serie temporal con tendencia.....21
Figura 3.11: Ejemplo de una serie temporal con ciclos estacionales [16].....21
Figura 3.12: Serie temporal con pulsos.....22
Figura 3.13: Serie temporal con tendencia agregada a la estacionalidad.....23
Figura 3.14: Serie temporal con tendencia multiplicada por la estacionalidad.....24
Figura 3.15: Comparación de una serie estacionaria con una no estacionaria con respecto a la media.....25
Figura 3.16: Comparación de una serie estacionaria con una no estacionaria con respecto a la varianza.....25
Figura 3.17: Comparación de una serie estacionaria con una no estacionaria

con respecto a la autocovarianza.....25
Figura 3.18: Serie temporal y su tendencia.....27
Figura 3.19: Componente irregular de la serie temporal tras eliminar su tendencia.....27
Figura 3.20: Fases del modelo ARIMA.....32
Figura 3.21: Regresión Lineal.....33
Figura 3.22: SVM [22].....34
Figura 3.23: SVR.....34
Figura 4.1: Topología de la red de carreteras de la zona objetivo [24].....37
Figura 4.2: Ventanas de tiempo para la predicción del tráfico.....38
Figura 4.3: Cálculo del error de predicción del tiempo promedio de viaje. .38
Figura 4.4: Cálculo del error de predicción del volumen de tráfico.....39
Figura 4.5: Rutas de la competición KDDCup2017 con los enlaces que las forman.....40
Figura 4.6: Tiempo promedio de viaje medio en cada uno de los días de entrenamiento de los meses de julio y agosto de 2016.....47
Figura 4.7: Tiempo promedio de viaje medio en cada uno de los días de entrenamiento de los meses de septiembre y octubre de 2016.....47
Figura 4.8: Tiempo promedio de viaje por horas en algunos días en la ruta A-2.....48
Figura 4.9: Tiempo promedio de viaje por horas en algunos días en la ruta B-3.....48
Figura 4.10: Barrera de peaje 1 en la dirección de entrada.....49
Figura 4.11: Barrera de peaje 3 en la dirección de entrada.....50
Figura 4.12: Bloque principal de código que crea la evolución de las 2 horas previas a la ventana de tiempo correspondiente en los datos de entrenamiento.....52
Figura 4.13: Obtención de las rutas de tráfico junto con las ventanas de tiempo comprendidas en los intervalos de tiempo a predecir.....52
Figura 4.14: Estructura iterativa para crear las columnas relacionadas con

la evolución del tráfico en las dos horas previas a la ventana de tiempo considerada....53
Figura 4.15: Consulta SQL que rellena la columna del tiempo promedio de viaje 20 minutos antes de la ventana de tiempo considerada....54
Figura 4.16: Consulta SQL que establece los valores de las columnas de la evolución del tiempo promedio de viaje de las 2 horas previas a la ventana de tiempo en consideración utilizando los valores de las columnas del anterior intervalo de tiempo....55
Figura 4.17: Combinación de la tabla con los datos de entrenamiento del tiempo promedio de viaje junto con los datos meteorológicos....55
Figura 4.18: Código SQL que crea una vista para cada ruta e intervalo a partir de la vista con los datos combinados....56
Figura 4.19: Obtención de los datos de entrenamiento para la ruta e intervalo en consideración....57
Figura 4.20: Obtención de los datos de prueba para la ruta e intervalo en consideración....57
Figura 4.21: Entrenamiento y predicción del tiempo promedio de viaje en los distintos días de predicción con el modelo XGBoost....58
Figura 4.22: Cálculo del error medio de los días de predicción para una ruta e intervalo determinado....58
Figura 4.23: Acumulación de los errores correspondiente a los días a predecir de cada uno de los intervalos a estimar....58
Figura 4.24: Cálculo del segundo sumatorio de la fórmula del error MAPE para un algoritmo....58
Figura 4.25: Acumulación de los errores del segundo sumatorio de la fórmula del error MAPE....58
Figura 4.26: Cálculo del primer sumatorio de la fórmula del error MAPE para un algoritmo....59
Figura 4.27: Obtención de los valores reales del tiempo promedio de viaje para las rutas e intervalos de tiempo a predecir....60

Figura 4.28: Cálculo del valor del tiempo promedio de viaje de una parte de aquellas rutas e intervalos de las que no disponemos datos.....61
Figura 4.29: Concatenación de los datos de entrenamiento de una ruta determinada con los datos reales de las dos horas previas a un intervalo a predecir en un día y ruta determinada.....61
Figura 4.30: Cálculo del mejor modelo ARIMA para una ruta, día y ventana de tiempo a estimar.....62
Figura 4.31: Concatenación de las dos horas previas a los intervalos de tiempo a predecir un día determinado con la serie temporal de una ruta...63
Figura 4.32: Serie temporal a problema de aprendizaje supervisado.....64
Figura 4.33: Estructura generada con el método de la ventana deslizante..64
Figura 4.34: Ejemplo de gráfica del volumen de tráfico en una ruta determinada en los días de entrenamiento.....68
Figura 4.35: Eliminación del ruido de los datos proporcionados para la predicción del volumen de tráfico.....68
Figura 4.36: Gráfica resultante de la eliminación de ruido.....69

Índice de tablas

Tabla 3.1: Estructura de los datos de entrada para el algoritmo KNN [8]. . 12
Tabla 4.1: Errores MAPE de la primera aproximación de predicciones del tiempo promedio de viaje. 59
Tabla 4.2: Errores MAPE de la tercera aproximación de predicciones del tiempo promedio de viaje. 65
Tabla 4.3: Errores MAPE de la primera aproximación de predicciones del volumen de tráfico. 67
Tabla 4.4: Errores MAPE de la tercera aproximación de predicciones del volumen de tráfico. 70
Tabla 7.1: Presupuesto del material utilizado en el proyecto. 75

Capítulo 1
Introducción

Cada día generamos una gran cantidad de información, algunas veces conscientes de que lo hacemos y otras veces inconscientes de ello porque lo desconocemos. Nos damos cuenta de que generamos información cuando registramos nuestra entrada en el trabajo, cuando entramos en un servidor para ver nuestro correo, cuando pagamos con una tarjeta de crédito o cuando reservamos un billete de avión. Otras veces no nos damos cuenta de que generamos información, como cuando conducimos por una vía donde están contabilizando el número de automóviles que pasan por minuto, cuando se sigue nuestra navegación por Internet o cuando nos sacan una fotografía del rostro al haber pasado cerca de una oficina gubernamental.

En las últimas décadas se ha producido un aumento exponencial del volumen y variedad de información, generándose grandes volúmenes de conjuntos de datos. El almacenamiento masivo de datos ha generado un interés por analizar, interpretar y extraer información útil de los mismos con el objetivo de obtener conocimiento. Actualmente, los datos son la materia prima para conseguir información provechosa, que se puede utilizar para llevar a cabo una toma de decisiones y la realización de conclusiones. De esta manera, surge el concepto de **minería de datos**, que se define como el proceso de *extraer conocimiento útil y comprensible*, previamente desconocido, desde grandes cantidades de datos almacenados en distintos formatos. Es decir, la tarea fundamental de la misma es encontrar modelos inteligibles a partir de los datos que permitan encontrar aspectos previamente desconocidos de los mismos.

En muchas situaciones, el método tradicional de convertir los datos en conocimiento consiste en un análisis e interpretación realizada de forma manual. El especialista en la materia analiza los datos y elabora un informe o hipótesis que refleja las tendencias o pautas de los mismos. Esta forma de actuar es lenta, costosa y muy subjetiva. En realidad, el análisis manual es impracticable en dominios donde el volumen de los datos crece exponencialmente. Consecuentemente, muchas decisiones importantes se

realizan no sobre la base de los datos disponibles, sino siguiendo la propia intuición del usuario al no disponer de las herramientas necesarias. No obstante, actualmente existen herramientas de apoyo, como son las herramientas **OLAP**(*On-line Analytical Processing*, Procesamiento Analítico en Línea), que son técnicas de análisis descriptivo y de sumarización que permite transformar los datos en otros datos agregados o cruzados de manera sofisticada.

En este trabajo pretendemos predecir el tiempo de duración de los recorridos y el volumen de tráfico soportado por una red de carreteras. Los datos utilizados han sido proporcionados por el *KDDCup 2017* y, para estas tareas, utilizaremos diversas técnicas de minería de datos, tales como *XGBoost*, *LightGBM*, *Redes Neuronales* y *k-Vecinos Más Cercanos*, entre otros.

1.1 Antecedentes y estado del arte

El problema de predecir los flujos de tráfico en las redes de carreteras ha sido ampliamente estudiado en la literatura. Algunos ejemplos de estudio en este ámbito son los siguientes:

1. En [1] se propone utilizar series temporales multivariadas (método de predicción multivariante) para pronosticar variables de tráfico. Este método se plantea frente a las series temporales escalares que, aunque en teoría son genéricamente suficientes para reconstruir la dinámica de los sistemas subyacentes, resulta más enriquecedor utilizar todas las variables disponibles.

2. En [2] se estudia la cuestión acerca del tiempo de validez en que se mantiene eficaz una serie histórica de tiempos de flujo del tráfico para predecir el futuro. Es decir, se plantea el tiempo t que puede perdurar la eficacia de los datos de flujo de tráfico existentes en un tiempo t_0 para estimar tendencias de variación del flujo de tráfico en un tiempo t_0+t .

3. En [3] se expone un Trabajo de Fin de Grado en el que se trata el tema de la predicción del tráfico en las carreteras de la red de la Generalitat Valenciana. El autor pretendía con este trabajo analizar la posibilidad de mejorar la metodología empleada en algunas fases de la explotación de los datos de aforos a través del uso de métodos científicos, complementándola con herramientas estadísticas. Esto se hace puesto que los planes de aforo no pueden obtener el muestreo completo de toda la red durante todo el año dado el alto número de puntos de toma de datos y los recursos necesarios para abarcarlos todos de forma

permanente. Para realizar la mejora, el autor considera la Intensidad Media Diaria (IMD) de tráfico como la variable más importante a calcular en un Plan de Aforos. Para poder llevar a cabo el cálculo de dicha variable, define los tramos sobre la red de carreteras (además de realizar estudios de retramificación para valorar los cambios en la red y adaptar los tramos definidos a la realidad viaria conforme ésta va evolucionando). A continuación, realiza un diseño de muestreo (de cada uno de los tramos de aforo con el objetivo de obtener muestras lo suficientemente representativas como para caracterizar el tráfico en cada tramo, de forma que la asignación de recursos sea óptima) y, a partir de esto, se diseña el plan de distribución de muestreo de estaciones (diversos tipos de estación según la frecuencia de muestreo de los mismos), asignando cada una de ellas a una tipología. Para llevar a cabo esto último se tienen en cuenta los recursos materiales y humanos de los que se dispone para poder cumplir con el muestreo del plan anual de aforos resultante de dichas asignaciones.

4. En [4] se estudian las tendencias de movilidad urbana en Paris, Santiago de Chile, Singapur y Viena con el objetivo de analizar la demanda de las diversas formas de transporte que existen en esas ciudades y establecer políticas adecuadas. No solo se examinan estas tendencias, sino también sus causas. Para ello, primero se identifican las tendencias específicas de cada una de las ciudades principalmente a través de indicadores de transporte, como los datos de viaje con respecto a los usuarios y estructuras espaciales, además de analizar el contexto de cada ciudad (infraestructura, desarrollo económico y desarrollo social de la ciudad) y realizar un modelo para explicar el comportamiento de los usuarios frente unas tendencias u otras a partir de la diferenciación entre los motivos socio-emocionales y racionales de los mismos. A continuación, se consultan a expertos en el sistema de transporte de cada una de las ciudades para validar esas tendencias identificadas y preparar análisis cualitativos. Por último, se realizan análisis para comprender y describir las tendencias desde la perspectiva de los viajeros. El artículo examina una amplia gama de modos de transportes espacialmente y socialmente diferenciados.

5. En [5] se realizan estimaciones a corto plazo (15 minutos en el futuro) con la información histórica del tráfico de la red de autopistas del Reino Unido utilizando redes neuronales, de tal forma que esto permita reducir la congestión del transporte mediante la mejora de sistemas inteligentes de transporte utilizados para controlar el tráfico para que realicen decisiones proactivas sobre la red de carreteras (anticiparse al inicio de la congestión del tráfico). Se plantean efectuar estas decisiones

anticipadas a través de advertencias de la congestión esperada, lo que permitiría a los controladores disponer de más tiempo para evaluar las diferentes estrategias de mitigación, en lugar de una vez que se materialice la congestión. También se propone que las predicciones se hagan visibles al público, de manera que el sistema de transporte se pueda beneficiar ya que permitiría a los usuarios optimizar sus planes de viaje, ya sea redirigiendo o reprogramando el itinerario de su viaje.

1.2 Objetivos

Los objetivos contemplados a completar en el desarrollo de este TFG son los siguientes:

- *Familiarizarse con diferentes técnicas y algoritmos específicos de minería de datos.* Una de las finalidades de este TFG es aprender las nociones fundamentales de distintas técnicas de minería de datos y aplicarlas en el contexto de flujos de tráfico.
- *Aprender a utilizar una variedad de tipos de software de propósito específico para proyectos de minería de datos.* En este trabajo también se pretende cultivarse en el empleo de diferentes tipos de software relacionados con el almacenamiento, manejo y tratamiento de datos; la minería de datos y la gestión de proyectos.
- *Realización de un proyecto de KDD completo.* Otro de los objetivos de este proyecto es aplicar el proceso denominado **KDD (Knowledge Discovery in Databases)**, que es el proceso de extracción de conocimiento en bases de datos.

Los objetivos específicos de este proyecto de minería de datos son:

- *Realización de predicciones del tiempo promedio de viaje.* Dada una serie de rutas y unos intervalos de tiempo, nos planteamos realizar predicciones del tiempo promedio de viaje en tales trayectos.
- *Realización de predicciones del volumen de tráfico.* Dadas unas barreras de peaje, direcciones de conducción e intervalos de tiempo, nos planteamos realizar estimaciones del volumen de tráfico en las mismas.

1.3 Organización de la memoria

La disposición de la información que se va a seguir para abarcar todo lo relacionado con el desarrollo y ejecución del proyecto es la siguiente:

- En el capítulo 2 se introducen las tecnologías utilizadas para llevar a cabo la carga de las bases de datos que almacenan los datos del proyecto y la aplicación de diferentes modelos de predicción sobre los mismos.
- En el capítulo 3 se explican las nociones fundamentales de la minería de datos, así como la descripción de distintas técnicas de aprendizaje automático empleadas para descubrir patrones y tendencias en nuestros datos.
- En el capítulo 4 se abordan de forma detallada las distintas fases del proyecto realizado. Estas fases comprenden la creación de las bases de datos, su modificación para adecuarlas a las tareas de predicción, la realización de predicciones del *tiempo promedio de viaje* y del *volumen de tráfico* y la comparación de resultados de los distintos algoritmos de aprendizaje automático .
- En el capítulo 5 se exponen las conclusiones y las líneas futuras planteadas para seguir desarrollando el proyecto llevado a cabo.
- En el capítulo 6 se presentan las conclusiones y las líneas futuras en inglés.
- En el capítulo 7 se detalla el presupuesto utilizado para llevar a cabo el proyecto.

Capítulo 2
Tecnologías

La elección de las herramientas y tecnologías empleadas para el desarrollo del proyecto se ha realizado meticulosamente, de tal forma que nos facilitara el desarrollo de las diferentes tareas a abordar en el proyecto. A continuación, se enumeran las principales tareas del proyecto junto con el software al que se ha recurrido para completarlas.

2.1 Almacenamiento de datos

El almacenamiento de los datos proporcionados es fundamental para llevar a cabo un análisis y un tratamiento adecuado de los mismos, de tal forma que estén disponibles para las diferentes técnicas de minería de datos. Existen multitud de sistemas para almacenar y gestionar bases de datos, como *MariaDB*, *Oracle*, *SQL Server* y *PostgreSQL*.

MariaDB es un sistema de administración de bases de datos relacional. Se trata de un programa capaz de almacenar una enorme cantidad de datos de gran variedad y de distribuirlos. Este sistema incluye todos los elementos necesarios para instalar el programa, preparar diferentes niveles de acceso de usuario, administrar el sistema y proteger y hacer volcados de datos.

Oracle Database es un sistema de gestión de bases de datos de tipo objeto-relacional desarrollado por *Oracle Corporation*. Se considera como uno de los sistemas de bases de datos mas completos, destacando el *soporte de transacciones*, la *estabilidad,* la *escalabilidad* y el *soporte multiplataforma.* No obstante, la gran potencia que tiene y su elevado precio hace que solo se utilice en empresas muy grandes y multinacionales, por norma general.

Microsoft SQL Server es un sistema de gestión de base de datos relacional (RDBMS) producido por *Microsoft*. Su principal lenguaje de consulta es *Transact-SQL*. Algunas de las características principales de este sistema son el soporte de transacciones y de procedimientos almacenados, la escalabilidad, la estabilidad y la seguridad.

El sistema gestor de bases de datos utilizado para almacenar y gestionar los datos de este proyecto de minería de datos es **PostgreSQL**. *PostgreSQL* es un potente sistema de base de datos objeto-relacional de código abierto basado en el paquete POSTGRES. Es un sistema que permite la creación, gestión y administración de bases de datos, así como la elección y manejo de las estructuras necesarias para el almacenamiento y búsqueda de la información del modo más eficiente posible.

Comparado con *PostgreSQL*, *MariaDB* se ha enfocado tradicionalmente en aplicaciones web de lectura mayormente, donde la principal preocupación es la optimización de consultas sencillas. En cambio, *PostgreSQL* se ha enfocado tradicionalmente en la fiabilidad, integridad de datos y características integradas enfocadas al desarrollador. Por otra parte, la diferencia principal entre *Oracle* y *PostgreSQL* es el hecho de que el primero no es software de código abierto, mientras que el segundo si, además de que *PostgreSQL* hace más sencillo el análisis de datos y tiene una mayor seguridad. En cuanto a *Microsoft SQL* Server, una de las diferencias principales de *PostgreSQL* con respecto a este sistema gestor de bases de datos es que es **multiplataforma**; *PostgreSQL* puede ejecutarse en Linux, BSD y Windows, pero *Microsoft SQL Server* solo se puede ejecutar en Windows. Además, el segundo posee una facilidad de uso mayor que el primero, pero *PostgreSQL* es más lento que *Microsoft SQL Server*.

2.2 Análisis de datos

Existen diversos tipos de software para aplicar diferentes técnicas de minería de datos sobre la información almacenada y encontrar patrones en la misma que nos puedan aportar información valiosa sobre los futuros flujos de tráfico. Algunos de los más importantes son *RapidMiner*, el *lenguaje R*, *Weka* y el *lenguaje Python*.

RapidMiner es un programa informático para el análisis y la minería de datos. Permite el desarrollo de procesos de análisis de datos mediante el encadenamiento de operadores a través de un entorno gráfico. Algunas de las características principales de este software son que está *desarrollado en Java*, es *multiplataforma* e incluye gráficos y herramientas de *visualización de datos*.

R es un entorno y un lenguaje de programación enfocado en el **análisis estadístico**. Es un *lenguaje interpretado*, funciona mediante comandos y proporciona una *amplia gama de herramientas estadísticas* que incluyen análisis de datos y generación de gráficos de alta calidad.

Weka es un software libre y de código abierto basado en Java. Comprende una colección de *algoritmos de aprendizaje automático* para minería de datos y empaqueta *herramientas de preprocesamiento*, *clasificación*, *regresión*, *clustering*, *reglas de asociación* y *visualización de datos*. Además, tiene una interfaz gráfica fácil de usar para la visualización bidimensional de datos minados.

Para realizar el análisis de datos de este proyecto se ha procedido a emplear el lenguaje de programación denominado **Python**. *Python* es un lenguaje de programación *interpretado* y *multiparadigma*, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es ampliamente utilizado para el *aprendizaje automático* y la *ciencia de datos* debido al excelente soporte de bibliotecas y a que es un lenguaje de programación de propósito general.

Este lenguaje tiene a su disposición un ecosistema de bibliotecas en *Python* para matemáticas, ciencias e ingeniería denominado **SciPy**. Es un complemento de Python necesario para el aprendizaje automático y está compuesto por los siguientes módulos básicos relevantes: ***NumPy*** (permite trabajar eficientemente con datos en vectores y matrices), ***Matplotlib*** (permite crear gráficos en 2D y gráficos a partir de datos) y ***Pandas*** (contiene herramientas y estructuras de datos para organizar y analizar datos).

Por otra parte, la biblioteca fundamental para desarrollar y realizar aprendizaje automático en Python se denomina **scikit-learn**. El núcleo de la biblioteca son los algoritmos de aprendizaje automático para clasificación, regresión, agrupamiento y más, y también proporciona herramientas para tareas relacionadas tales como la *evaluación de modelos*, *ajuste de parámetros* y *preprocesamiento de datos*.

Python, con respecto a *R*, es un lenguaje de propósito general con una sintaxis fácil de entender y con una curva de aprendizaje muy corta. En cambio la funcionalidad de *R* se desarrolla pensando en los estadísticos, lo que le da ventajas específicas de campo tales como importantes características para la visualización de datos, pero es más difícil de aprender.

2.3 Gestión del proyecto

Aparte de llevar a cabo un almacenamiento de los datos proporcionados y su utilización por parte de diferentes técnicas de minería de datos, es

imprescindible planificar y organizar los recursos utilizados en el desarrollo del TFG. Para ello se han empleado distintos tipos de software.

Por un lado se ha utilizado **GitHub** para guardar toda la información vinculada al TFG. Es una plataforma de **desarrollo colaborativo de software** para alojar proyectos utilizando el sistema de control de versiones *Git*. GitHub, aparte de ser un servicio de alojamiento de código, ofrece varias herramientas útiles para el **trabajo en equipo**.

Por otra parte se ha procedido a usar **Gedit** para desarrollar la programación de código necesaria. Es un editor de textos de propósito general que enfatiza la simplicidad y facilidad de uso. Es el editor predeterminado de *GNOME*.

También nos hemos apoyado en el software denominado **ProjectLibre**, una aplicación de planificación de proyectos que permite introducir tareas y secuenciarlas, añadir recursos (humanos, materiales y de coste), emitir informes, imprimir gráficos y un largo etcétera.

Capítulo 3
La minería de datos

3.1 Introducción

La **minería de datos** es el proceso de descubrir automáticamente información útil en grandes repositorios de datos [6]. Las técnicas de minería de datos se utilizan para rastrear grandes bases de datos con el fin de encontrar patrones novedosos y útiles que, de otro modo, podrían seguir siendo desconocidos. También proporcionan capacidades para predecir el resultado de una observación futura, como predecir el tiempo meteorológico o, en relación a este trabajo, características de tráfico.

La minería de datos es una parte integral del *descubrimiento de conocimiento en bases de datos* (**KDD)**, cuyo proceso general se muestra en la **Figura 3.1**. Este proceso consiste en una serie de pasos, desde la selección de datos hasta el postprocesamiento de los resultados de la minería de datos.

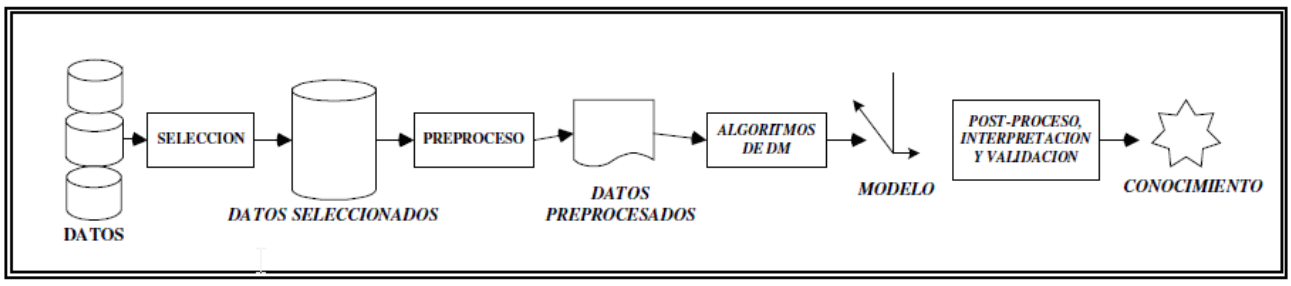


Figura 3.1: El proceso de descubrimiento de conocimiento en bases de datos (KDD) [7]

En este proceso, los **datos de entrada** seleccionados se *transforman* en un formato apropiado para el análisis posterior (**preprocesamiento**, en el que se incluyen la fusión de datos de múltiples fuentes, la limpieza de datos para eliminar el ruido, la duplicación de observaciones y la selección de registros y características que son relevantes para la tarea de minería de datos).

A continuación, se **integran los resultados de la minería de datos** en los sistemas de apoyo a la toma de decisiones; es la **fase de modelado**, donde se seleccionan y aplican los métodos de minería de datos apropiados. Por último, se lleva a cabo un **tratamiento posterior** que garantice que en el sistema de apoyo a la adopción de decisiones sólo se incorporen resultados válidos y útiles. Es decir, se **identifican los patrones obtenidos** y que son realmente interesantes, basándose en algunas medidas y se realiza una **evaluación de los resultados obtenidos**.

Las tareas de minería de datos se dividen generalmente en dos categorías principales:

- **Tareas predictivas**. El objetivo de estas tareas es predecir el valor de un atributo particular basado en los valores de otros atributos. El atributo a predecir se conoce comúnmente como la *variable objetivo* o *dependiente*, mientras que los atributos utilizados para hacer la predicción se conocen como las *variables explicativas* o *independientes*. Algunos ejemplos de tareas predictivas son la ***clasificación*** (para las variables objetivo discretas) y la ***regresión*** (para variables objetivo continuas).
- **Tareas descriptivas.** El objetivo es describir los datos que resumen las relaciones subyacentes en los datos. Algunos ejemplos de tareas descriptivas son el ***análisis de asociaciones*** (utilizado para descubrir patrones representados en forma de reglas de implicación o subcojuntos de características que describan características fuertemente asociadas en los datos), ***clustering*** (para encontrar grupos de observaciones estrechamente relacionados de tal forma que las observaciones que pertenecen al mismo clúster sean más similares entre sí que con respecto a observaciones que pertenecen a otros clústeres) y ***detección de anomalías*** (identificar observaciones cuyas características son significativamente diferentes del resto de los datos).

3.2 Técnicas de minería de datos

Para realizar las predicciones del tiempo promedio de viaje y el volumen de tráfico propuestas ha sido imprescindible la utilización de técnicas de minería de datos apropiadas para dichas tareas. En los siguientes apartados se lleva a cabo una explicación detallada de las mismas.

3.2.1 Técnicas basadas en vecindad

Este tipo de técnicas son de *clasificación supervisada* (predicen un dato (o un conjunto de ellos) desconocido a priori a partir de otros conocidos) y se basan en clasificar una nueva instancia en función de las clases, conocidas de antemano, de las instancias más próximas a ella. Así, las muestras pertenecientes a una clase se encontrarán próximas en el espacio de representación.

El algoritmo utilizado que se basa en criterios de vecindad se denomina ***k vecinos más cercanos*** *(k-NN Nearest Neighbors)*. La idea básica sobre la que se fundamenta este algoritmo es que un nuevo caso se va a clasificar en la *clase más frecuente a la que pertenezcan sus* ***k vecinos más cercanos.*** El paradigma se fundamenta, por tanto, en una idea muy simple e intuitiva, lo que unido a su fácil implementación hace que sea un algoritmo clasificatorio muy extendido.

3.2.1.1 El algoritmo K-NN básico

Para comprender el algoritmo básico de esta técnica de aprendizaje automático se expone el siguiente ejemplo:

		X_1	...	X_j	...	X_n	C
$(\mathbf{x}_1, c_1)$	1	x_{11}	...	x_{1j}	...	x_{1n}	c_1
	⋮	⋮		⋮		⋮	⋮
$(\mathbf{x}_i, c_i)$	i	x_{i1}	...	x_{ij}	...	x_{in}	c_i
	⋮	⋮		⋮		⋮	⋮
$(\mathbf{x}_N, c_N)$	N	x_{N1}	...	x_{Nj}	...	x_{Nn}	c_N
$\mathbf{x}$	$N+1$	$x_{N+1,1}$	...	$x_{N+1,j}$	...	$X_{N+1,n}$	?

Tabla 3.1: Estructura de los datos de entrada para el algoritmo KNN [8]

La notación a utilizar es la siguiente:

- La **Tabla 3.1** corresponde a un fichero ***D*** de ***N*** casos, cada uno de los cuales está caracterizado por ***n*** variables predictoras, $\boldsymbol{X_1,\ldots,X_n}$, y una variable a predecir, la clase ***C***.
- Los ***N*** casos se denotan por:

  $$(x_1, c_1), \ldots, (x_n, c_n)$$

  para todo *i=1,...,N* donde

y

$$c_i \in (c^1, \ldots, c^m)$$

para todo $i=1,\ldots,M$

$\boldsymbol{c^1,\ldots,c^m}$ denotan los $\boldsymbol{m}$ posibles valores de la variable clase $\boldsymbol{C}$.

- El nuevo caso que se pretende clasificar se denota por

$$x = (x_1, \ldots, x_n)$$

En la siguiente figura se presenta un pseudocódigo para el clasificador K-NN básico:

COMIENZO
Entrada: $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_N, c_N)\}$
$\mathbf{x} = (x_1, \ldots, x_n)$ nuevo caso a clasificar
PARA todo objeto ya clasificado (x_i, c_i)
calcular $d_i = d(\mathbf{x}_i, \mathbf{x})$
Ordenar $d_i (i = 1, \ldots, N)$ en orden ascendente
Quedarnos con los K casos $D_{\mathbf{x}}^K$ ya clasificados más cercanos a $\mathbf{x}$
Asignar a $\mathbf{x}$ la clase más frecuente en $D_{\mathbf{x}}^K$
FIN

Figura 3.2: Algoritmo básico KNN [8]

Tal y como puede observarse en el mismo, se calculan las distancias de todos los casos ya clasificados al nuevo caso, $\boldsymbol{x}$, que se pretende clasificar. Una vez seleccionados los $\boldsymbol{K}$ casos ya clasificados más cercanos al nuevo caso, $\boldsymbol{x}$, a éste se le asignará la clase (valor de la variable $\boldsymbol{C}$) más frecuente de entre los $\boldsymbol{K}$ objetos. La siguiente figura muestra de manera gráfica un ejemplo de esto:

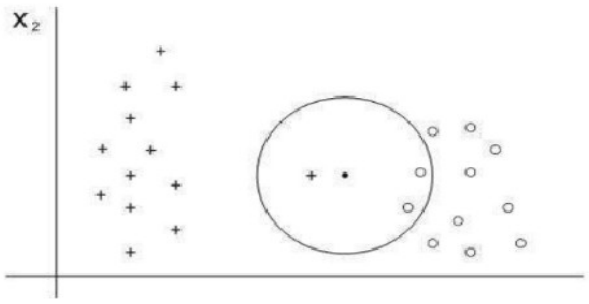


Figura 3.3: Ejemplo de aplicación del

algoritmo K-NN básico [8]

Tal y como se puede apreciar en la figura, tenemos ***24*** casos ya clasificados en dos posibles valores (***m=2***). Las variables predictoras son $\boldsymbol{X_1}$ y $\boldsymbol{X_2}$, y se ha seleccionado ***K=3***. De los 3 casos ya clasificados que se encuentran más cercanos al nuevo caso a clasificar, $\boldsymbol{x}$ (representado por $\bullet$), dos de ellos pertenecen a la clase $\circ$, por lo que el clasificador 3-NN predice la clase $\circ$ para el nuevo caso.

En caso de que se produzca un *empate entre dos o más clases*, conviene tener una *regla heurística* para su ruptura. Un ejemplo de de regla heurística para la ruptura de empates pueden ser seleccionar la clase con distancia menor, etc. Además, todos los datos deben estar normalizados para evitar que las características en el conjunto de entrada con valores más altos dominen el cálculo de la distancia.

En este proyecto de minería de datos se aborda un problema de regresión, por lo que, para llevar a cabo las tareas de predicción, se utiliza el *método de regresión basado en KNN.*

3.2.1.2 Método de regresión basado en KNN

El método de los *k-vecinos más cercanos* se adapta fácilmente a la **regresión de funciones con valores continuos**. Mediante una medida de distancia en el conjunto de datos ya clasificados se determinan los $\boldsymbol{k}$ datos más cercanos al nuevo dato $\boldsymbol{x_q}$ para aproximar una función a partir de los $\boldsymbol{k}$ valores ya seleccionados. Esta función corresponde al p*romedio de los* $\boldsymbol{k}$ *valores más cercanos*; si se considera el promedio aritmético (todos los datos dentro del grupo tienen igual relevancia), la función aproximación tiene la siguiente forma:

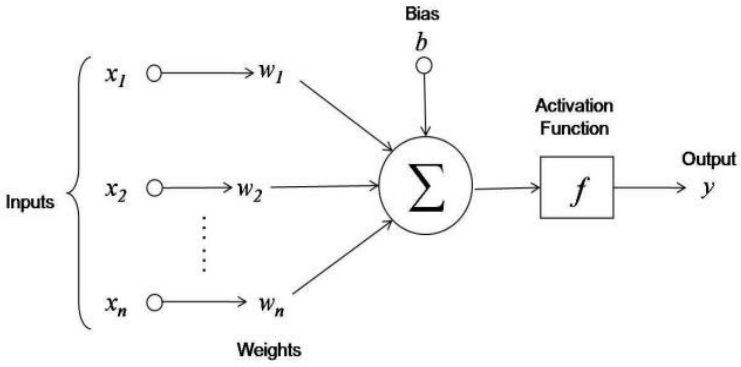
$$\hat{f}(x_q) = \frac{1}{k} \sum_{i=1}^{k} f(x_i)$$

3.2.2 Técnicas basadas en redes neuronales

Las **redes neuronales** son un sistema de interconexión de neuronas en una red que colabora para producir un estímulo de salida. Un ejemplo de red neuronal es el *perceptrón multicapa.*

El **perceptrón multicapa** es una red neuronal artificial (RNA) formada por múltiples capas que le permiten resolver problemas que no son linealmente separables. El bloque de construcción de este algoritmo son las *neuronas artificiales*, que son unidades computacionales simples que ponderan

las señales de entrada y producen una señal de salida usando una función de activación:

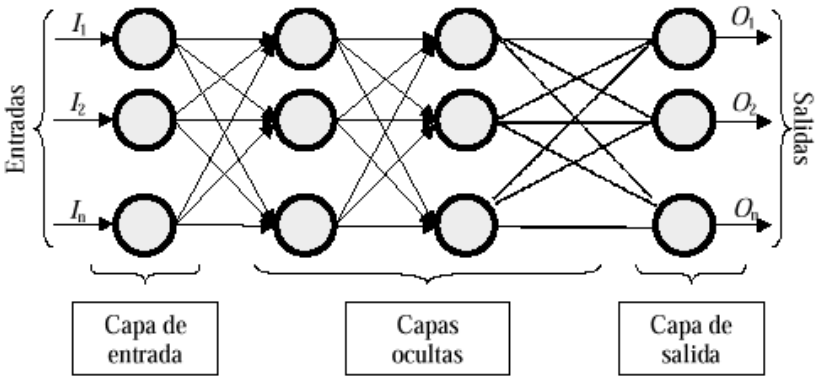


[9]

Figura 3.4: Estructura de una neurona artificial

En primer lugar en el *perceptrón multicapa* se recibe una serie de **entradas**, que pueden ser características de un conjunto de entrenamiento o salidas de otras neuronas. A continuación, se aplican unos **pesos** a las entradas, que se suelen inicializar a valores aleatorios pequeños, como valores en el rango de *0* a *0.3*. Después, las entradas ponderadas se suman junto con un *sesgo* o *bias* que tiene la neurona (se interpreta como una entrada que permite desplazar la función de activación a la izquierda o a la derecha, que siempre tiene el valor 1.0 y que también debe ser ponderada) que, a su vez, pasan a través de una **función de activación**, obteniendo así las **salidas**. Esta *función de activación* es un simple mapeo de la entrada ponderada sumada a la salida de la neurona; es decir, se utiliza para determinar la salida de la red neuronal como *si* o *no:* mapea los valores resultantes de *0* a *1* o de *-1* a *1*, etc. (dependiendo de la función). Las distintas funciones de activación se engloban en dos tipos, *funciones de activación lineales* y *funciones de activación no lineales* y existen una gran variedad de ellas: *función sigmoide*, *Tanh*, *ReLU*, etc.

La estructura de una red neuronal se compone de las siguientes partes:



[10]

Figura 3.5: Estructura de una red neuronal

- **Capa de entrada:** La capa que toma la entrada del conjunto de datos se denomina *capa de entrada* o *capa visible*, ya que es la parte expuesta de la red. Éstas no son neuronas como se describió anteriormente, sino que simplemente pasan el valor de entrada a la siguiente capa.
- **Capas ocultas:** Las capas posteriores a la capa de entrada se denominan *capas ocultas* porque no están expuestas directamente a la entrada y están formadas por aquellas neuronas cuyas entradas provienen de capas anteriores y cuyas salidas pasan a neuronas de capas posteriores.
- **Capa de salida:** La última capa oculta se denomina *capa de salida* y es responsable de emitir un valor o vector de valores que corresponden al formato requerido para el problema. La elección de la función de activación en la capa de salida está fuertemente limitada por el tipo de problema que se está modelando.

3.2.2.1 Entrenamiento de una red neuronal

El primer paso para entrenar una red neuronal es *preparar los datos*, de tal forma que éstos deben ser numéricos y tienen que estar escalados de manera consistente. Con la *normalización* (reescalar al rango entre 0 y 1) y la *estandarización* (para que la distribución de cada columna tenga la media de cero y la desviación estándar de 1, de modo que todas las entradas se expresan en rangos similares), el proceso de entrenamiento de la red neuronal se realiza con mucha mayor velocidad.

El algoritmo de entrenamiento para redes neuronales más utilizado se denomina ***descenso por gradiente estocástico***, en el que una fila de datos se expone a la red neuronal a la vez como entrada. En este algoritmo la red

procesa la entrada hacia delante activando las neuronas a medida que se va avanzando a través de las capas ocultas hasta que finalmente se obtiene un valor de salida. Esto se denomina *propagación hacia delante* en la red neuronal.

La salida del grafo se compara con la salida esperada y se calcula el *error*. Este error es entonces propagado de nuevo hacia atrás a través de la red neuronal, una capa a la vez, y los pesos son actualizados de acuerdo a su grado de contribución al error calculado (*algoritmo de retropropagación*). El proceso se repite para todos los ejemplos de los datos de entrenamiento y el proceso de actualizar la red neuronal para todo el conjunto de datos de entrenamiento se denomina *época*. Una red neuronal puede ser entrenada por decenas, cientos o muchos miles de épocas.

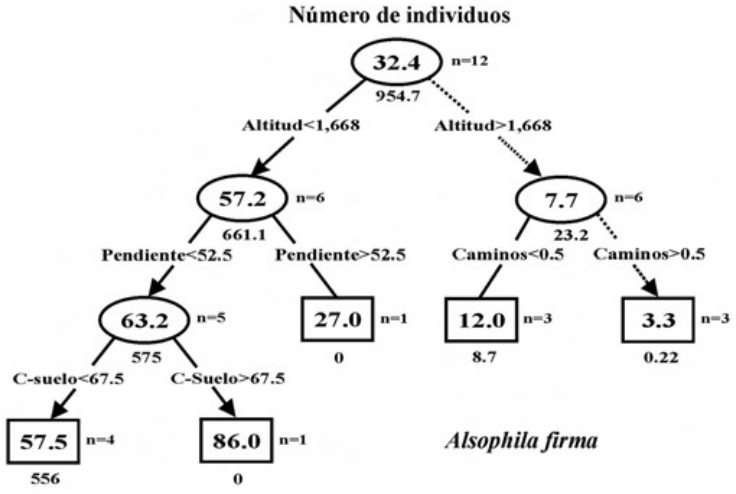
Los pesos en la red neuronal se pueden actualizar a partir de los errores calculados para cada ejemplo de entrenamiento y esto se denomina ***aprendizaje en línea***. De forma alternativa, los errores se pueden guardar en todos los ejemplos de entrenamiento y la red se puede actualizar al final. Esto se denomina ***aprendizaje por lotes***.

Por otra parte, el grado en el que se actualizan los pesos es controlado por un parámetro de configuración denominado ***velocidad de aprendizaje***. Este parámetro controla el cambio realizado en el peso de la red neuronal para un error determinado. A menudo se utilizan tamaños de peso pequeños tales como *0.1*, *0.01* o más pequeños.

Una vez que una red neuronal ha sido entrenada puede ser usada para realizar predicciones. La topología de la red neuronal y el conjunto final de pesos es todo lo que necesita para implantar el modelo. Las predicciones se realizan proporcionando la entrada a la red y ejecutando una propagación hacia delante que genera una salida que se utiliza como predicción.

3.2.3 Técnicas basadas en árboles de decisión

El **aprendizaje mediante árboles de decisión** es un método de aproximación de una función objetivo de valores discretos en el cual la función objetivo es representada mediante un *árbol de decisión.* En estas estructuras de árbol, las *hojas* representan etiquetas de clase y las *ramas* representan las conjunciones de características que conducen a esas etiquetas de clase. Los árboles de decisión donde la variable de destino puede tomar valores continuos se llaman **árboles de regresión**:



[11]

Figura 3.6: Ejemplo de árbol de regresión

Un ejemplo de algoritmo basado en árboles de decisión es el ***XGBoost*** (*eXtreme Gradient Boosting*), una implementación de este tipo de técnicas potenciada por gradientes y centrada en la velocidad de cálculo y el rendimiento del modelo.

Para entender esta técnica de minería de datos es de gran importancia comprender en qué consiste la **potenciación del gradiente** (o *gradient boosting*). Es una técnica en la que se crean nuevos modelos que predicen los residuos o errores de modelos anteriores y luego se suman para llevar a cabo la predicción final. Se denomina *potenciación del gradiente* porque utiliza un algoritmo de descenso de gradiente para minimizar la pérdida al añadir nuevos modelos. Es una técnica de aprendizaje automático utilizada para el análisis de regresión y los problemas de clasificación estadística, que produce un modelo predictivo en forma de un conjunto de modelos predictivos débiles, en este caso *árboles de decisión.*

Este algoritmo es eficiente para convertir hipótesis relativamente pobres en hipótesis muy buenas. Una *hipótesis débil* se define como aquella cuyo desempeño es al menos ligeramente mejor que el azar. La **potenciación de hipótesis** (*hypothesis boosting*) es la idea de filtrar las observaciones, dejando aquellas observaciones que el *weak learner* puede manejar y enfocándose en desarrollar nuevos aprendizajes débiles para manejar las observaciones difíciles restantes. La idea es utilizar el método de aprendizaje débil varias veces para obtener una sucesión de hipótesis, cada una reorientada hacia los ejemplos que los anteriores encontraban difíciles y mal clasificados, de manera que las salidas de los modelos subsiguientes se sumen y corrijan los residuos en las predicciones. Para llevar a cabo esto, la potenciación del gradiente requiere tres elementos: una **función de coste** a optimizar, un ***weak learner*** para hacer predicciones y un **modelo aditivo** para añadir *weak learners* para minimizar la función de error, en el que los árboles se añaden uno a la vez y

los árboles existentes en el modelo no se modifican.

Aparte del **XGBoost**, otro ejemplo es el **LightGBM** (*Light Gradient Boosting Machine*)**,** un framework de potenciación del gradiente que también utiliza un *algoritmo de aprendizaje basado en árboles*. Es un framework rápido, distribuido y de alto rendimiento, utilizado para clasificar, priorizar y muchas otras tareas de aprendizaje automático.

LightGBM es similar a *XGBoost* pero varía en algunas formas específicas, especialmente en la forma en que crea los árboles. *LightGBM* realiza la construcción de los árboles **verticalmente**, mientras que *XGBoost* lo hace **horizontalmente**, lo que significa que el primero genera los árboles *leaf-wise* (búsqueda primero el mejor) mientras que el otro los genera *level-wise* (búsqueda en anchura).

En comparación con el crecimiento *level-wise*, el algoritmo *leaf-wise* puede converger mucho más rápido. Esto se debe a que, al crecer el árbol sobre la misma hoja en *LightGBM*, el algoritmo *leaf-wise* puede reducir más errores que el algoritmo *level-wise* y, por lo tanto, da como resultado una precisión mucho mejor. Además, es sorprendentemente rápido, de ahí la palabra *Light* ('Ligero').

Sin embargo, las divisiones *leaf-wise* provocan un aumento de la complejidad y pueden dar lugar a un ajuste excesivo si no se utilizan con los parámetros adecuados. Se puede superar este inconveniente especificando otro parámetro de *profundidad máxima* que especifica la profundidad a la que se producirá la división.

A continuación se exponen una serie de figuras para explicar la diferencia de forma visual:

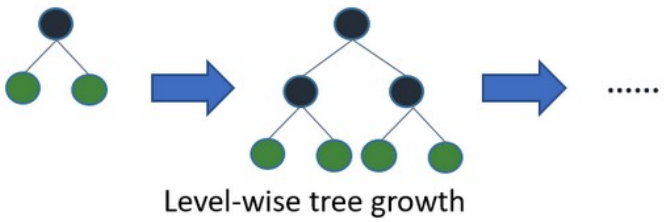
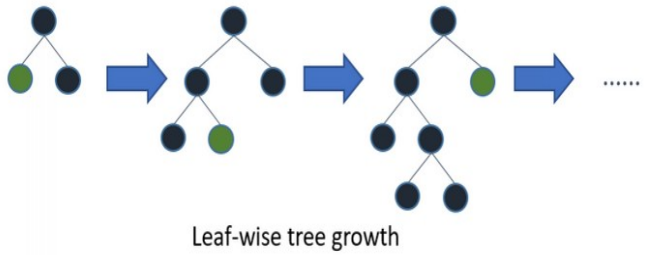


Figura 3.7: Crecimiento level-wise del árbol en XGBoost [12]



[12]

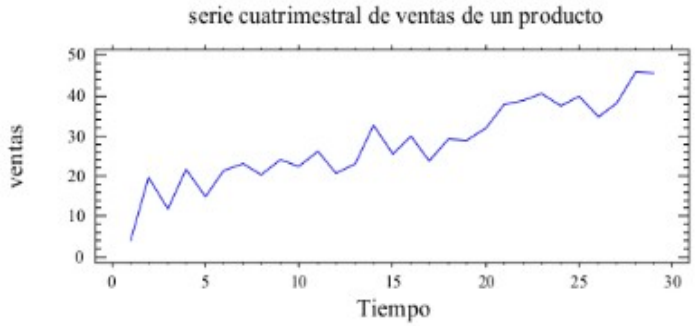
Figura 3.8: Crecimiento leaf-wise del árbol en LightGBM

Este algoritmo presenta una serie de ventajas con respecto a otros algoritmos de *boosting*: **mayor velocidad de entrenamiento**, **mayor eficiencia**, **menor uso de memoria**, **mayor precisión** y **compatibilidad con grandes conjuntos de datos**. No obstante, no es aconsejable utilizar *LightGBM* en pequeños conjuntos de datos puesto que este algoritmo es sensible al sobreajuste.

3.2.4 Técnicas estadísticas

La **estadística** es una rama de las matemáticas y una herramienta que estudia usos y análisis provenientes de una muestra representativa de datos, que busca *explicar las correlaciones y dependencias de un fenómeno físico o natural* de ocurrencia en forma aleatoria o condicional. Dos técnicas estadísticas conocidas son el ***modelo ARIMA*** y la ***Regresión Lineal***.

Para entender el *modelo ARIMA*, es imprescindible el concepto de *serie temporal*. Una **serie temporal** es una colección ordenada de mediciones tomadas en intervalos regulares [13]. Los intervalos pueden representar cualquier unidad de tiempo, pero debe utilizarse un mismo intervalo para todas las mediciones. Un ejemplo de serie temporal es el siguiente :



[14]

Figura 3.9: Ejemplo de serie temporal

Las series temporales muestran una o varias de estas características:

- **Tendencia:** Una **tendencia** es un cambio gradual ascendente o descendente *a largo plazo* en el nivel de la serie o la trayectoria que siguen los valores de la serie (cambio en la media) de aumentar o disminuir a lo largo del tiempo.

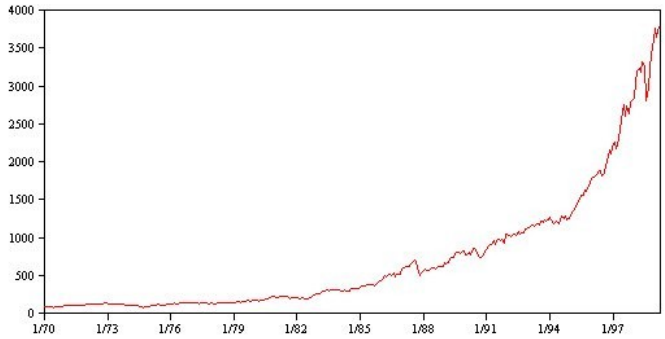


Figura 3.10: Ejemplo de una serie temporal con tendencia [15]

Las tendencias también pueden ser **lineales** o **no lineales**. Las tendencias *lineales* son incrementos aditivos positivos o negativos en el nivel de la serie y las tendencias *no lineales* suelen ser multiplicativas, con incrementos proporcionales a los valores de series anteriores.

- **Ciclos estacionales y no estacionales:** Un **ciclo estacional** es un patrón repetitivo y predecible de los valores de las series temporales. Se dice que las series con un ciclo estacional muestran **estacionalidad**; los patrones estacionales resultan útiles para obtener buenos ajustes y previsiones.

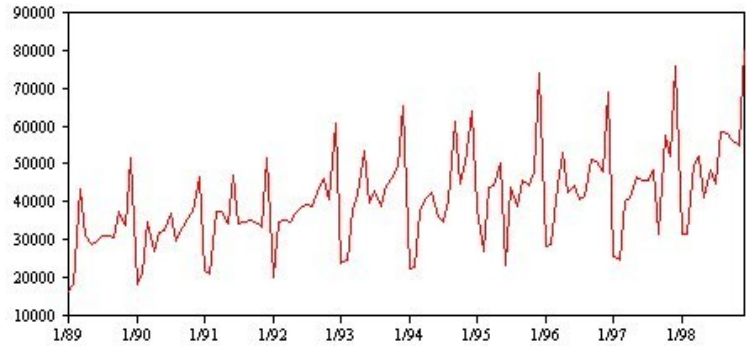
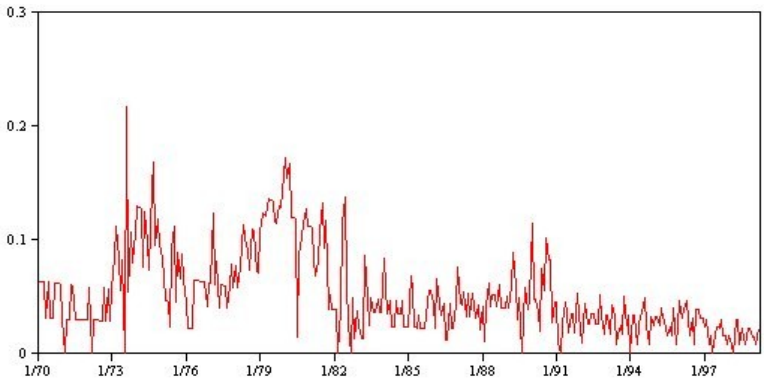


Figura 3.11: Ejemplo de una serie temporal con ciclos estacionales [16]

Por otra parte, un **ciclo no estacional** es un patrón repetitivo y

posiblemente impredecible de los valores de las series. Este tipo de patrones son difíciles de modelar y suelen aumentar la incertidumbre de las previsiones.

- **Pulsos y pasos:** Muchas series temporales experimentan cambios bruscos de nivel. Normalmente son de dos tipos: un cambio repentino y *tempora*l, o ***pulso***, en el nivel de la serie, y un cambio repentino y *permanente*, o ***paso***, en el nivel de la serie.



[17]

Figura 3.12: Serie temporal con pulsos

Cuando se observan pasos o pulsos, es importante encontrar una explicación convincente. Los modelos de series temporales están diseñados para explicar cambios graduales y no repentinos. Por tanto, suelen subestimar los pulsos y pueden quedar inutilizados por los pasos, lo que da como resultado modelos poco ajustados y previsiones imprecisas. No obstante, si se puede explicar una alteración, se puede modelar mediante una **intervención** o un **evento**.

3.2.4.1 Componentes de las series temporales

El estudio descriptivo de series temporales se basa en la idea de descomponer la variación de una serie en varias componentes básicas. Este enfoque descriptivo consiste en encontrar componentes que correspondan a una ***tendencia a largo plazo***, un ***comportamiento estacional*** y una ***parte aleatoria***. Por lo tanto, las componentes o fuentes de variación que se consideran habitualmente son las siguientes:

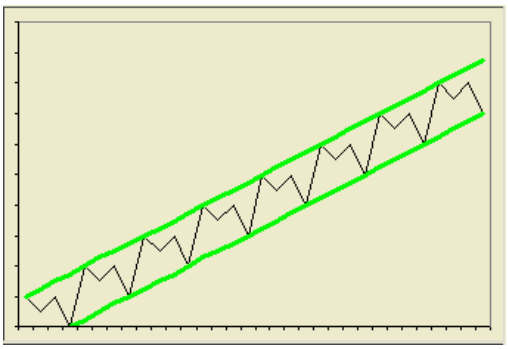
- **Tendencia secular o regular**: La denotaremos por ***t***.
- **Efecto Estacional (Variación estacional):** La denotaremos por ***e***.
- **Componente Aleatoria (Variación aleatoria, residual, irregular o accidental)**: Una vez identificados los componentes anteriores y

después de haberlos eliminado, persisten unos valores que son aleatorios. Se pretende estudiar qué tipo de comportamiento aleatorio presentan estos residuos, utilizando algún tipo de modelo probabilístico que los describa y nos permita conocer el comportamiento de la serie a largo plazo. La denotaremos por ***r***.

Las tres componentes enumeradas explican conjuntamente los valores de la variable analizada en cada instante. En relación a estos componentes, los dos esquemas generalmente más admitidos sobre la forma en que la serie temporal se descompone en sus tres componentes son el ***aditivo*** y el ***multiplicativo***.

- El **esquema aditivo s**upone que las observaciones se generan como suma de las tres componentes, es decir:

$$Y = t + e + r$$



[18]

Figura 3.13: Serie temporal con tendencia agregada a la estacionalidad

En este caso cada componente se expresa en el mismo tipo de unidad que las observaciones. La variación residual, en este modelo, es independiente de las demás componentes; es decir la magnitud de dichos residuos no depende del valor que tome cualquier otra componente de la serie (análogamente la variación estacional y la tendencia).

- El **esquema multiplicativo** supone que las observaciones se generan como producto de las tres componentes, es decir:

$$Y = t \times e \times r$$

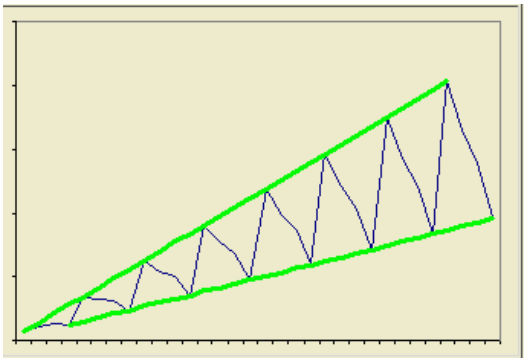


Figura 3.14: Serie temporal con tendencia multiplicada por la estacionalidad [18]

En este modelo (multiplicativo puro) la *tendencia secular* se expresa en el mismo tipo de unidad que las observaciones, y el resto de las componentes en tanto por uno. Aquí no se cumple la hipótesis de independencia del esquema aditivo.

- Otro tipo de modelo multiplicativo que si cumple la hipótesis de independencia es aquél llamado **modelo multiplicativo** mixto, que es el siguiente:

$$Y = t \times e + r$$

La distinción entre el modelo multiplicativo y el mixto tiene que ver con el comportamiento de la componente irregular.

3.2.4.2 Clasificación descriptiva de las series temporales

Las series temporales se pueden clasificar en:

- **Estacionarias**: Una serie es estacionaria cuando es estable, es decir, cuando *la media y la variabilidad son constantes a lo largo del tiempo*. Es una serie básicamente estable a lo largo del tiempo, sin que se aprecien aumentos o disminuciones sistemáticos de sus valores.

- **No Estacionarias**: Son series en las cuales l*a media y/o variabilidad cambian en el tiempo*. Los cambios en la media determinan una tendencia a crecer o decrecer a largo plazo, por lo que la serie no oscila alrededor de un valor constante.

La diferencia entre los dos tipos de series temporales se puede visualizar mejor en las siguientes imágenes:

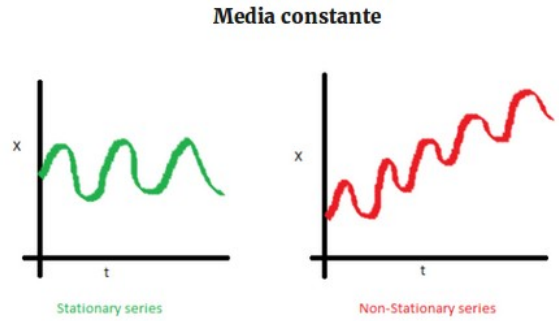


Figura 3.15: Comparación de una serie estacionaria con una no estacionaria con respecto a la media [19]

La serie de la izquierda tiene una media constante, en cambio la figura de la derecha muestra tendencia, y su media se incrementa con el paso del tiempo.

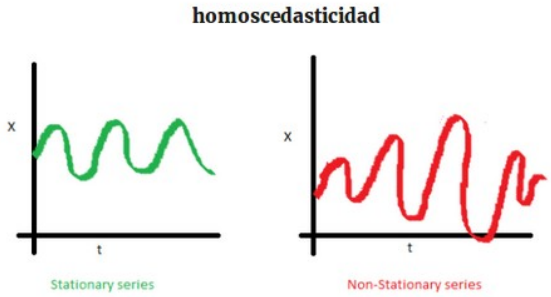
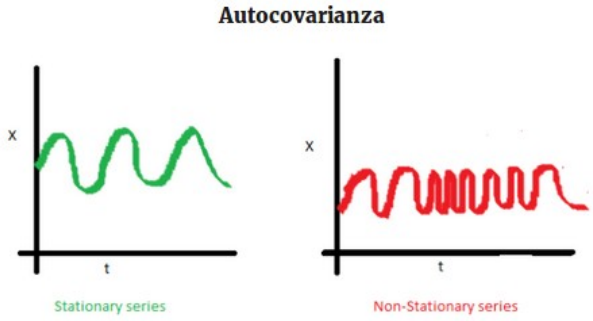


Figura 3.16: Comparación de una serie estacionaria con una no estacionaria con respecto a la varianza [19]

La serie de la derecha no es estacionaria, su varianza se incrementa.



[19]

Figura 3.17: Comparación de una serie estacionaria con una no estacionaria con respecto a la autocovarianza

En la serie de la derecha, la autocovarianza (covarianza) no es constante.

3.2.4.3 Funciones de autocorrelación y autocorrelación parcial

La ***autocorrelación*** y la ***autocorrelación parcial*** son medidas de asociación entre valores de series actuales y pasadas e indican cuáles son los valores de series pasadas más útiles para predecir valores futuros. Con estos datos se puede determinar el orden de los procesos en un modelo ARIMA.

- La **autocorrelación simple** (**FAC**) mide la relación lineal entre las observaciones de una serie de dato $\boldsymbol{Y_t}$, distanciados en un lapso de tiempo $\boldsymbol{k}$. El lapso de tiempo k se conoce como *retardo* o *retraso*. Este retardo denota el periodo de tiempo entre los valores de la serie para el cual se mide el tipo y grado de correlación de la variable considerada.

- La **autocorrelación parcial** (**FACP**), es una medida asociada a la autocorrelación simple. Es la *estimación de la autocorrelación simple*, para el mismo retardo k, con la eliminación del efecto producido por las autocorrelaciones para retardos menores a k, las cuales están presentes en la estimación de la autocorrelación simple. La autocorrelación parcial no considera las autocorrelaciones acumuladas para el retardo k para el que se estima.

Para saber los tipos de modelo **AR** y **MA** (explicados más adelante), es preciso observar las funciones de autocorrelación parcial y autocorrelación simple respectivamente.

3.2.4.4 Estimación de las componentes de una serie temporal

Con el objetivo de aislar la componente aleatoria de una serie temporal y realizar un análisis para saber qué modelo probabilístico se le adecúa más y saber el comportamiento de la serie a largo plazo, es necesario identificar las componentes de *tendencia* y *estacionalidad* de la misma.

Estimación de la tendencia:

Para estimar la variable de *tendencia* de la serie temporal, supondremos que tenemos una serie no estacionaria sin componente estacional, es decir, que la serie se puede descomponer en

$$Y_t = t + r$$

Para estimar $\boldsymbol{t}$ debemos realizar alguna hipótesis sobre su forma:

- **Tendencia determinista:** En este caso supondremos que la tendencia es una *función determinística*. La función más sencilla posible es una recta, es decir,

$$t = a + bt$$

donde a y b son dos constantes a determinar. La forma de estimar estas constantes es mediante un modelo de regresión lineal entre las variables Y_t y el tiempo $t = 1,2,3,...$ De esta forma, si estimamos los parámetros a y b, entonces la componente irregular será

$$r = Y - a - bt$$

El siguiente ejemplo presenta una tendencia que se podría expresar de forma lineal. La tendencia de la serie y la componente irregular serían, en este ejemplo,

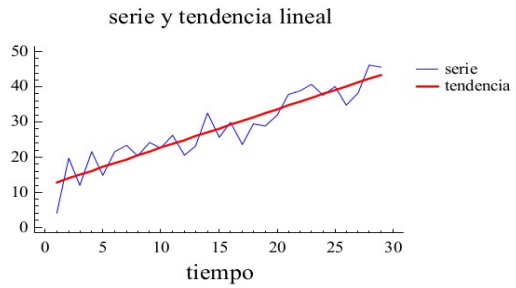


Figura 3.18: Serie temporal y su tendencia [14]

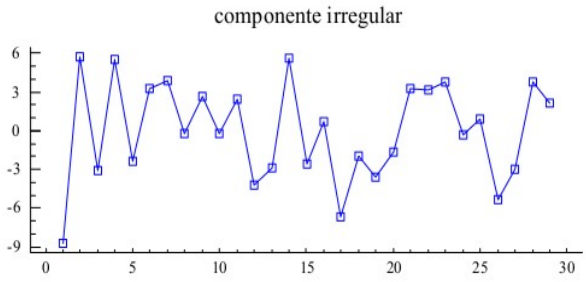


Figura 3.19: Componente irregular de la serie temporal tras eliminar su tendencia [14]

- **Tendencia evolutiva:** A menudo, la tendencia de la serie no sigue una recta y evoluciona a lo largo del tiempo. En ese caso, un método general de estimar la tendencia es suponer que evoluciona lentamente en el tiempo, y que se puede aproximar con una función sencilla para

intervalos cortos del tiempo. Para ello, suponemos que la representación de la tendencia por una recta es *válida para tres períodos consecutivos de tiempo* ***t-1***, ***t*** y ***t+1***, y representamos las tendencias en los tres periodos consecutivos de la siguiente manera:

$$T_{t-1} = T_t - \nabla T$$

$$T_t = T_t$$

$$T_{t+1} = T_t + \nabla T$$

Si hacemos la media de tres observaciones consecutivas

$$m_t = \frac{x_{t-1} + x_t + x_{t+1}}{3}$$

entonces

$$m_t = T_t + \frac{r_{t-1} + r_t + r_{t+1}}{3}$$

y como la componente irregular tiene **media cero**, la media de los tres valores de la componente irregular se puede suponer que es *despreciable frente a la tendencia*, y m_t representa la tendencia en ese instante. Esta operación se denomina *media móvil de orden tres*. Este método, denominado *medias móviles*, consiste fundamentalmente en agrupar sistemáticamente un número fijo ***k*** de valores de la serie y determinar para cada grupo su media.

- **Diferenciación de la serie:** Los dos métodos vistos con anterioridad sirven para estimar la tendencia y poder eliminarla de la serie temporal. Un tercer método más general para eliminar la tendencia consiste en suponer que la tendencia evoluciona lentamente en el tiempo, de manera que en el instante ***t*** la tendencia debe estar próxima a la tendencia en el instante ***t-1***. De esta forma, si restamos a cada valor de la serie el valor anterior, la serie resultante estará aproximadamente libre de tendencia. Esta operación se denomina **diferenciación de la serie** y consiste en pasar de la serie original x_t a la serie y_t mediante:

$$y_t = x_t - x_{t-1}$$

Estimación de la estacionalidad:

Para estimar la componente de *estacionalidad* de la serie se pueden utilizar varios métodos:

- **Variación de la media del período respecto de la media global:** Un método de estimar el efecto estacional (por ejemplo mensual) es considerar cómo varía la media del período (mes) respecto de la media global.
- **Diferenciación estacional de la serie:** Es un método más general que consiste en no hacer ninguna hipótesis sobre la forma general de la estacionalidad a corto plazo y suponer simplemente que evoluciona lentamente en el tiempo. Para ello, construimos una nueva serie diferenciando cada valor en un instante determinado con aquel valor anterior que se encuentra ***s*** instantes de tiempo hacia detrás:

$$y_t = x_t - x_{t-s}$$

Es decir, diferenciar estacionalmente la serie equivale a suponer que la estacionalidad en ***t*** es el valor de la serie en ***t-s:***

$$S_t = x_{t-s}$$

Esta serie se denomina ***serie diferenciada estacionalmente***.

3.2.4.5 Modelo ARIMA

Es una clase de modelos estadísticos para *analizar y pronosticar datos de series temporales*. Proporciona un método simple pero potente para hacer pronósticos hábiles de series temporales.

ARIMA es el acrónimo de *AutoRegressive Integrated Moving Average* (Modelo Autorregresivo Integrado de Media Móvil). Este acrónimo es descriptivo, capturando los aspectos clave del modelo mismo:

- **AR: Autoregresión**. Un modelo que utiliza la relación dependiente entre una observación y un cierto número de observaciones pasadas.
- **I: Integrado**. El uso de la diferenciación de observaciones brutas para que la serie temporal sea estacionaria.
- **MA: Media móvil**. Un modelo que utiliza la dependencia entre una observación y un error residual de un modelo de media móvil aplicado a

observaciones pasadas.

Cada uno de estos componentes se especifica explícitamente en el modelo como parámetro. Se utiliza la notación estándar ***ARIMA(p,d,q)***. Los parámetros del modelo ARIMA se definen de la siguiente manera:

- **Autorregresivo (p)**. Es el número de órdenes autorregresivos del modelo. Los órdenes autorregresivos especifican los valores previos de la serie utilizados para predecir los valores actuales. El proceso autorregresivo de orden **p**, representado por *ARIMA(p,0,0)* o simplemente por *AR(p)*, tiene la siguiente expresión matemática:

$$AR(p) \equiv X_t = \Phi_1 X_{t-1} + \Phi_2 X_{t-2} + ... + \Phi_p X_{t-p} + a_t$$

Los *procesos autorregresivos* presentan función de autocorrelación parcial (ACFP) con un número finito de valores distinto de cero. Un proceso **AR(p)** tiene los primeros ***p*** términos de la función de autocorrelación parcial distintos de cero y los demás son nulos. En la práctica se considera que una muestra dada proviene de un proceso autorregresivo de orden ***p*** si los términos de la función de autocorrelación parcial son casi cero a partir del que ocupa el lugar ***p***.

- **Diferencia (d)**. Especifica el orden de diferenciación aplicado a la serie antes de estimar los modelos. La diferenciación es necesaria si hay tendencias (las series con tendencias suelen ser no estacionarias y el modelado de ARIMA asume la estacionariedad) y se utiliza para eliminar su efecto.

- **Media móvil (q)**. Es el número de órdenes de media móvil presentes en el modelo. Los órdenes de media móvil especifican el modo en que se utilizan las desviaciones de la media de la serie para los valores previos con el fin de predecir los valores actuales. El proceso de medias móviles de orden **q**, representado por *ARIMA(0,0,q)* o también por *MA(q)*, viene dado por la expresión:

$$MA(q) \equiv X_t = a_t - \nu_1 a_{t-1} - \nu_2 a_{t-2} - ... - \nu_q a_{t-q}$$

Los *procesos de medias móviles* presentan función de autocorrelación con un número finito de valores distintos de cero. Un proceso **MA(q)** tiene los primeros ***q*** términos de la función de autocorrelación distintos de cero y los demás son nulos.

Una extensión natural de los modelos *AR(p)* y *MA(q)* es un tipo de modelos que incluyen tanto términos *autorregresivos* como de *medias móviles*

y se definen como **ARIMA(p, 0, q)**. Se representan por la ecuación:

$$ARMA(p,q) \equiv X_t = \Phi_1 X_{t-1} + \Phi_2 X_{t-2} + ... + \Phi_p X_{t-p} + a_t - v_1 a_{t-1} - v_2 a_{t-2} - ... - v_q a_{t-q}$$

A partir de esta fórmula se establece el modelo **ARIMA(p, d, q)**. Un modelo *ARIMA(0,d,0)* es una serie temporal que se convierte en ruido blanco (proceso puramente aleatorio) después de ser diferenciada **d** veces. El modelo ***(0, d, 0)*** se expresa mediante el operador de cambio retroactivo *B*:

$$(1-B)^d X_t = a_t$$

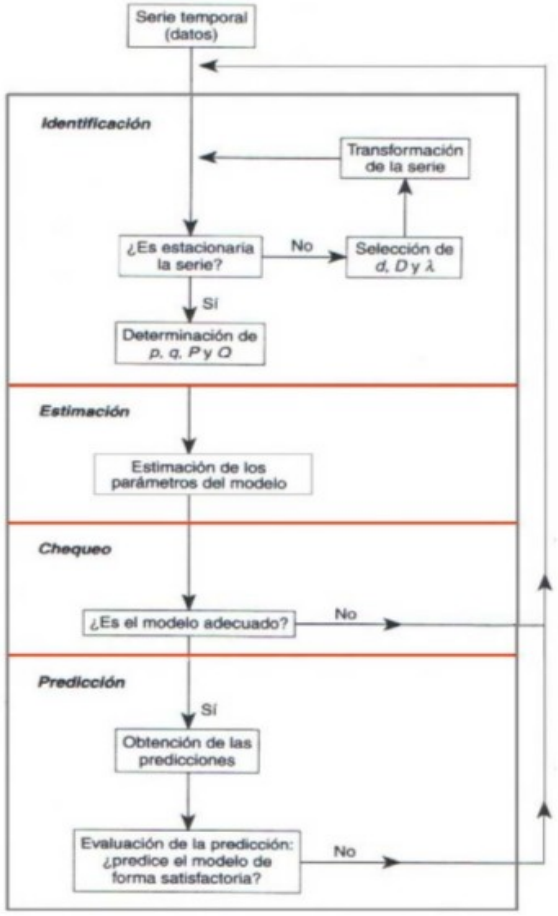
El modelo general *ARIMA(p, d, q)* toma la expresión:

$$ARIMA(p,d,q) \equiv (X_t - \Phi_1 X_{t-1} - \Phi_2 X_{t-2} - ... - \Phi_p X_{t-p})(1-B)^d = a_t - v_1 a_{t-1} - v_2 a_{t-2} - ... - v_q a_{t-q}$$

3.2.4.6 Fases del modelo ARIMA

La metodología de *Box y Jenkins* se resume en cuatro fases:

- **La primera fase** consiste en *identificar el posible modelo ARIMA que sigue la serie*, lo que requiere:
  - Decidir qué transformaciones aplicar para convertir la serie observada en una serie estacionaria.
  - Determinar un modelo ARMA para la serie estacionaria, es decir, los órdenes *p* y *q* de su estructura autorregresiva y de media móvil.
- **La segunda fase:** *seleccionar provisionalmente un modelo para la serie estacionaria.* Se pasa a la segunda etapa de estimación, donde los parámetros AR y MA del modelo se estiman y se obtienen sus errores estándar y los residuos del modelo.
- **La tercera fase** es el *diagnóstico*, donde se comprueba que los residuos no tienen estructura de dependencia y siguen un proceso de ruido blanco. Si los residuos muestran estructura se modifica el modelo para incorporarla y se repiten las etapas anteriores hasta obtener un modelo adecuado.
- **La cuarta fase** es la *predicción. U*na vez que se ha obtenido un modelo adecuado se realizan predicciones con el mismo.



[20]

Figura 3.20: Fases del modelo ARIMA

3.2.4.7 Regresión Lineal

La *Regresión Lineal* es un modelo matemático utilizado para aproximar la relación de dependencia entre una variable dependiente $\boldsymbol{Y}$, las variables dependientes $\boldsymbol{X}_i$ y un término aleatorio $\boldsymbol{\varepsilon}$. Este modelo puede ser expresado como:

$$Y_t = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon$$

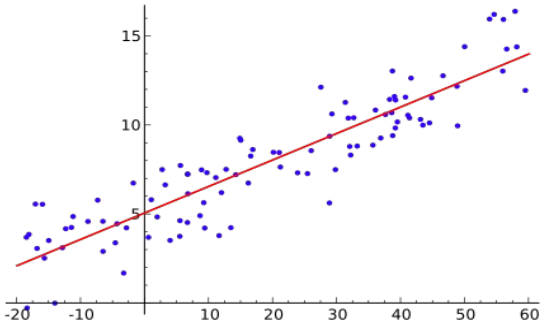
donde:

Y_t : variable dependiente o explicada

$X_1, X_2, \ldots, X_p$: variables independiente o explicativas

$\beta_0, \beta_1, \beta_2, \cdots, \beta_p$: parámetros, miden la influencia que las variables explicativas tienen sobre el regrediendo

donde β_0 es la **intersección** o **término "constante"**, las $\beta_i (i>0)$ son los parámetros respectivos a cada variable independiente, y p es el número de parámetros independientes a tener en cuenta en la regresión.



[21]

Figura 3.21: Regresión Lineal

Existen diversos algoritmos que se fundamentan en la *regresión lineal* y uno de los más conocidos es el de **mínimos cuadrados**. Este algoritmo, dados un conjunto de *pares ordenados* (variable independiente, variable dependiente) y una *familia de funciones*, se intenta encontrar la *función continua*, dentro de dicha familia, que mejor se aproxime a los datos de acuerdo con el criterio de *mínimo error cuadrático*. En su forma más simple, intenta minimizar la suma de cuadrados de las diferencias *en las ordenadas* (llamadas *residuos*) entre los puntos generados por la función elegida y los correspondientes valores en los datos.

3.2.5 Técnicas basadas en optimización

Este tipo de técnicas tienen como objetivo resolver un *problema de optimización de un modelo matemático*. Un **problema de optimización** trata de encontrar el máximo o el mínimo de una función sujeta a una serie de restricciones que pueden ser de igualdad o desigualdad. Existen diversos algoritmos que se fundamentan en esta técnica y uno de ellos se denomina ***Máquinas de Soporte Vectorial (SVM)***.

Las *Máquinas de Soporte Vectorial* son un algoritmo de aprendizaje supervisado que puede utilizarse tanto para desafíos de clasificación como de regresión; no obstante, se utiliza principalmente en problemas de clasificación. En este algoritmo trazamos cada elemento de datos (cada muestra de entrenamiento) como un punto en el espacio **n**-dimensional (donde **n** es el número de características del problema a resolver), con el valor de cada característica mapeándose al valor de una determinada coordenada. A continuación, dado este conjunto de ejemplos de entrenamiento, cada uno perteneciente a una clase, entrenamos una SVM para construir un modelo que prediga la clase de una nueva muestra mediante la construcción de un **hiperplano** que separe las clases de los datos de entrenamiento y maximice el margen entre esas clases (maximice la distancia entre los puntos más cercanos de cada clase al hiperplano de separación óptimo, llamados **vectores soporte**) en el espacio **n**-dimensional:

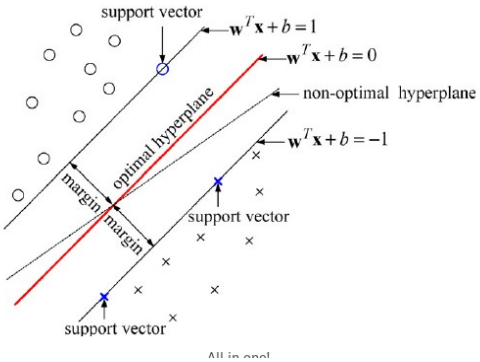


Figura 3.22: SVM [22]

Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de los espacios a los que pertenezcan, pueden ser clasificadas a una o la otra clase.

Las *Máquinas de Soporte Vectorial* también puede ser utilizado como un método de regresión (***SVR, Support Vector Regression***), manteniendo todas las características principales que caracterizan el algoritmo (margen máximo). En este caso, la idea es seleccionar el **hiperplano regresor** que mejor se ajuste a nuestro conjunto de datos de entrenamiento. Dado que en la práctica es muy difícil que los ejemplos de entrenamiento se ajusten al modelo lineal con un error de predicción igual a cero, se recurre al concepto de *margen blando*. De esta forma, se permite cierto ruido en los ejemplos de entrenamiento y, por tanto, se puede relajar la condición del error existente entre el valor predicho por la función y el valor real. Para llevar a cabo esto, se considera que todos los ejemplos que quedan confinados en la región *tubular* definida por $\pm\varepsilon$ no sean considerados vectores soporte. Además, se utilizan dos variables de holgura, ξ y ξ^*, para cuantificar la magnitud de dicho error (sumando el valor de esas variables):

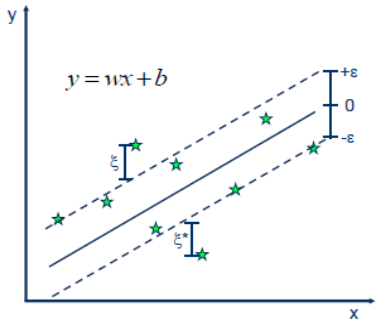


Figura 3.23: SVR [23]

La parte interesante de SVR es que puede utilizar un *kernel no lineal*. En este caso, se termina haciendo una regresión no lineal, es decir, ajustando una curva en lugar de una línea. Por lo tanto, la idea básica de SVR consiste en realizar un mapeo de los datos de entrenamiento $x \in X$, a un espacio de mayor dimensión F a través de un mapeo no lineal $\boldsymbol{\varphi}$*:* $X \rightarrow F$, donde podemos realizar una regresión lineal.

Capítulo 4
Proyecto de Minería de Datos

Con el objetivo de resolver el problema de predicción de flujos de tráfico en una red de carreteras hemos desarrollado un proyecto de minería de datos que hemos dividido en las siguientes fases:

- *Comprensión del problema de flujos de tráfico planteado por la competición KDDCup 2017.* En primer lugar se ha planteado llevar a cabo un estudio de las tareas encomendadas por la competición y de las reglas a aplicar en las predicciones a realizar.
- *Creación de bases de datos para almacenar los datos y modificación de las mismas para adecuarlas a las tareas de predicción.* Con el objetivo de poder acceder fácilmente y de forma flexible a la información suministrada, se ha planteado crear un conjunto de bases de datos que permita ordenar y clasificar los datos, de tal forma que se tenga una comprensión más acertada de ellos, se puedan realizar distintas combinaciones sobre los mismos y se genere nuevo conocimiento para llevar a cabo estimaciones. Una vez hecho esto, se propone modificar los esquemas de las tablas que componen dichos almacenes de datos para que la información contenida en ellos sea lo más manejable posible.
- *Visualización de los datos almacenados.* Con el fin de visualizar la información contenida en los datos suministrados y tener una comprensión global de la misma, se pretende desarrollar una serie de gráficas que nos permitan conocer las características de los datos y poder abordar las predicciones a realizar de la mejor forma posible.
- *Realización de predicciones del tiempo promedio de duración del recorrido y del volumen de tráfico*. Tras llevar a cabo la preparación preliminar de los datos a utilizar, se ha propuesto llevar a cabo diversas aproximaciones para predecir el **tiempo promedio de viaje** en las rutas y los intervalos de tiempo a estimar y el **volumen de tráfico** en las distintas barreras de peaje en las direcciones de

entrada y salida y en las ventanas de tiempo a pronosticar.

- *Desarrollo de un análisis comparativo de los resultados obtenidos a partir de la utilización de las distintas técnicas de minería de datos contempladas.* Tras utilizar diversas técnicas de minería de datos sobre los datos para realizar estimaciones del tiempo promedio de viaje y del volumen de tráfico, se ha propuesto llevar a cabo una comparación de los resultados de los distintos algoritmos empleados con el objetivo de obtener una visión general sobre la actuación de cada uno de ellos.

4.1 *Problema planteado por la competición KDDCup 2017*

4.1.1 Contexto

Los peajes de las autopistas son cuellos de botella bien conocidos en las redes de tráfico de vehículos. Durante las horas punta, las largas colas que se crean en los peajes pueden abrumar a las autoridades de gestión del tráfico. Por lo tanto, se desea implantar una serie de contramedidas efectivas preventivas para resolver este desafío. Tales contramedidas incluyen *agilizar el proceso de cobro de peaje* y *optimizar el flujo de tráfico futuro.* La optimización de la recaudación del peaje se podría llevar a cabo simplemente distribuyendo temporalmente recaudadores de peaje para abrir más carriles. Además, el futuro flujo de tráfico podría ser optimizado adaptando las señales de tráfico en función del flujo de tráfico presente en las intersecciones. Las contramedidas preventivas sólo funcionarán cuando las autoridades de gestión del tráfico reciban predicciones fiables para el flujo de tráfico futuro. Por ejemplo, si se predice un tráfico denso en la siguiente hora, los reguladores de tráfico podrían desplegar inmediatamente más recaudadores de peaje y/o desviar tráfico en las intersecciones.

Los patrones del flujo de tráfico varían debido a diferentes factores estocásticos , como las condiciones meteorológicas, los días festivos, la hora del día, etc. La predicción del futuro flujo de tráfico del ***ETA*** (Tiempo Estimado de Llegada) es un reto conocido. Una gran cantidad sin precedentes de datos de tráfico de aplicaciones móviles como *Waze* (en EE.UU.) o *Amap* (en China) puede ayudarnos a resolver este reto de forma más precisa.

4.1.2 Tareas

Las tareas propuestas por la competición *KDDCup2017* tienen que ver con la predicción de flujos de tráfico y son las que se enumeran a continuación:

- **Tarea 1**: Estimar el *tiempo promedio de viaje* desde las intersecciones designadas hasta las barreras de peaje. Para cada ventana de tiempo de 20 minutos, predecir el tiempo promedio de viaje de los vehículos para una ruta específica (ver la *Figura 4.1*):
    - Rutas desde la intersección *A* hasta las intersecciones *2* y *3*.
    - Rutas desde la intersección *B* hasta las intersecciones *1* y *3*.
    - Rutas desde la intersección *C* hasta las intersecciones *1* y *3*.

*Nota: El **tiempo estimado de viaje** (**ETA**) de una ventana de tiempo de 20 minutos para una ruta determinada es el tiempo medio de viaje de todas las trayectorias de vehículos que entran en la ruta en esa ventana de tiempo.*

- **Tarea 2**: Para cada ventana de tiempo de 20 minutos, predecir el *volumen de tráfico de entrada* en las barreras de peaje *1*, *2* y *3*. Hay que tener en cuenta que la barrera de peaje *2* sólo permite el tráfico de entrada a la autopista, mientras que las demás barreras de peaje permiten el tráfico en ambos sentidos (entrada y salida). Por lo tanto, necesitamos predecir el volumen de tráfico para 5 pares *barrera de peaje-dirección* en total.

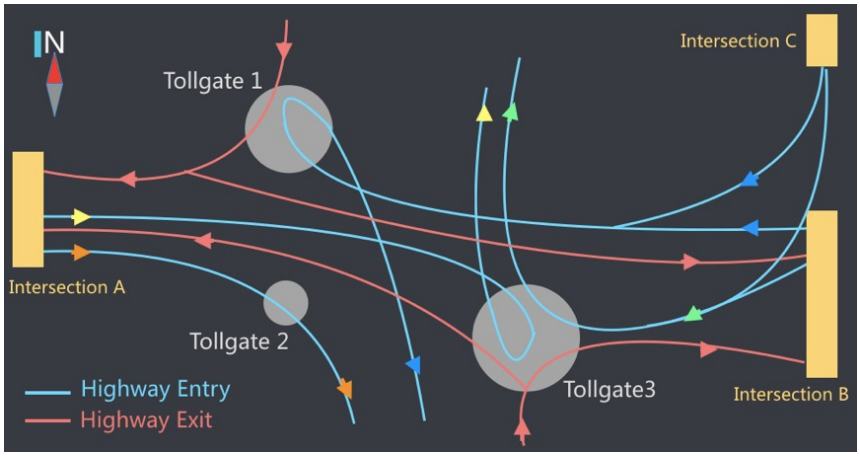


Figura 4.1: Topología de la red de carreteras de la zona objetivo [24]

4.1.3 Etapas y reglas de la competición

Las predicciones de tráfico se llevarán a cabo en las horas punta desde el

día 18 al 24 de octubre de 2016. Concretamente, se debe predecir el tráfico resultante durante las franjas horarias mostradas en la *Figura 4.2*; es decir, en las franjas horarias **8:00-10:00** y **17:00-19:00**.

Para la predicción del *tiempo de viaje*, el conjunto de entrenamiento inicial contiene los datos recogidos desde el día **19 de Julio** hasta el **17 de Octubre de 2016**. Con respecto a la estimación del *volúmen de tráfico*, el conjunto de entrenamiento inicial contiene los datos recogidos desde el día **19 de Septiembre** hasta el **17 de Octubre de 2016.**

Figura 4.2: Ventanas de tiempo para la predicción del tráfico [24]

En los conjuntos de datos de pruebas se proporcionan datos de tráfico durante las franjas horarias verdes mostradas en la ***Figura 4.2***. Esta información se puede utilizar como un indicador de tráfico en las próximas dos horas, que es lo que se va a proceder a predecir. En este aspecto, los concursantes no están restringidos a usar sólo los datos anteriores de las 2 horas antes de los intervalos de tiempo a estimar. No obstante, cada predicción está restringida a usar sólo los datos de tráfico **anteriores a la ventana de tiempo** predicha.

4.1.4 Métricas de evaluación

Para evaluar los resultados de las predicciones realizadas, se ha elegido la medida de error denominada **MAPE** (***Mean Absolute Percentage Error***, **Error Porcentual Absoluto Medio**). La fórmula para calcular esta medida de error es la siguiente:

$$MAPE = \frac{1}{R}\sum_{r=1}^{R}\frac{1}{T}\sum_{t=1}^{T}\frac{1}{D}\sum_{d=1}^{D}\left|\frac{d_{rtd} - p_{rtd}}{d_{rtd}}\right|$$

Figura 4.3: Cálculo del error de predicción del tiempo promedio de viaje

Esta fórmula se aplica a la primera tarea propuesta por la competición donde $\boldsymbol{R}$, $\boldsymbol{T}$, $\boldsymbol{D}$, $\boldsymbol{d_{rtd}}$ y $\boldsymbol{p_{rtd}}$ son el *número de rutas* , el n*úmero de ventanas de tiempo a predecir*, el *número de días en los que hay que predecir* y los v*alores actual y predicho del tiempo promedio de viaje* para una ruta r durante la ventana de tiempo t y el día d.

Para la segunda tarea propuesta, la fórmula es la que se muestra a continuación:

$$MAPE = \frac{1}{C}\sum_{c=1}^{C}\frac{1}{T}\sum_{t=1}^{T}\frac{1}{D}\sum_{d=1}^{D}\left|\frac{f_{ctd} - p_{ctd}}{f_{ctd}}\right|$$

Figura 4.4: Cálculo del error de predicción del

volumen de tráfico

donde $\boldsymbol{C}$, $\boldsymbol{T}$, $\boldsymbol{D}$, $\boldsymbol{f_{ctd}}$ y $\boldsymbol{p_{ctd}}$ son el *número de pares barrera de peaje-dirección* , el n*úmero de ventanas de tiempo a predecir*, el *número de días en los que hay que predecir* y los v*alores actual y predicho del volumen de tráfico* para un par barrera de peaje-dirección específico r durante la ventana de tiempo t y el día d.

4.2 Creación de bases de datos para almacenar los datos de la competición y modificaciones

Con el fin de almacenar los datos proporcionados por la competición de forma que sean fácilmente accesibles y permita una mejor comprensibilidad de los mismos, se ha utilizado el sistema de gestión de bases de datos ***PostgreSQL***.

4.2.1 Estructura de bases de datos propuesta

La información suministrada se ha almacenado en cuatro bases de datos distintas. Por un lado, se ha construido una base de datos con los datos de entrenamiento originales de la competición denominada ***tfgdatosoriginales***. Por otro lado, se ha procedido a crear una base de datos denominada ***tfgdatosmodificados*** similar a la anterior pero con los tipos de los atributos modificados para adecuarse a las tareas a realizar y otra serie de alteraciones

llevadas a cabo. Además, por conveniencia de separación de los datos de entrenamiento de los de testeo, se ha construido una base de datos con los datos de prueba denominada ***tfgtest***. Aparte, se ha generado otro almacén de datos denominado ***tfgrealvalues*** para guardar los datos reales de los días a predecir.

4.2.2 Base de datos con los datos originales

4.2.2.1 Tabla *vehicle_routes*

El esquema de la tabla original proporcionada por la competición es la que se expone a continuación:

vehicle_routes (***intersection_id***, ***tollgate_id***, ***link_seq***)

Esta tabla contiene los *tramos de carretera* que forman cada una de las rutas de la red de carreteras propuesta.

4.2.2.2 Tabla *road_links*

El esquema de la tabla original proporcionada por la competición es la siguiente:

road_links (**link_id, length, width, lanes, in_top, out_top, lane_width**)

Esta tabla contiene la **descripción de cada uno de los tramos** que forman las rutas. Para cada tramo de la ruta, el tráfico de vehículos proviene de uno o más *"enlaces viales entrantes"* y entra en uno o más *"enlaces viales salientes"* :

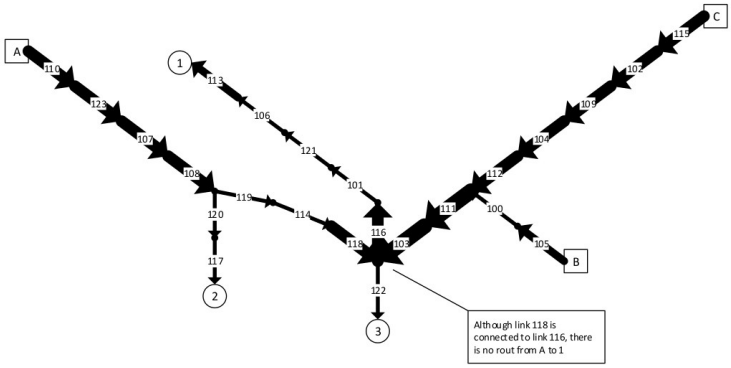


Figura 4.5: Rutas de la competición KDDCup2017 [25]

con los enlaces que las forman

4.2.2.3 Tabla *vehicle_trajectories_training*

El esquema de la tabla original proporcionada por la competición es la que se expone a continuación:

vehicle_trajectories_training (***intersection_id***, ***tollgate_id***, ***vehicle_id***, ***starting_time***, ***travel_seq***, ***travel_time***)

Esta tabla contiene cada uno de los vehículos que han viajado en algún momento, entre el **19 de Julio** y el **17 de Octubre**, por alguna de las rutas establecidas en la tabla *vehicle_routes*. Para cada vehículo se establece el momento en el que entró en una ruta, el tiempo que estuvo ese vehículo en cada uno de los tramos que forman dicha ruta y el tiempo total de viaje que tardó en realizar esa ruta.

4.2.2.4 Tabla *traffic_volume_tollgates_training*

El esquema de la tabla original proporcionada por la competición es la siguiente:

traffic_volume_tollgates_training (***time***, ***tollgate_id***, ***direction***, ***vehicle_model***, ***has_etc***, ***vehicle_type***)

En esta tabla se registran todos los vehículos que han pasado por alguna barrera de peaje situada en la topología de carreteras proporcionada por la competición en una dirección determinada. Con respecto al atributo *vehicle_type*, la competición no proporciona datos en esta columna, por lo que no se ha considerado. Los aforadores actuales, no obstante, permiten distinguir entre vehículos ligeros y pesados.

4.2.2.5 Tabla *weather_data*

El esquema de la tabla original proporcionada por la competición es la que se expone a continuación:

weather_data (***date***, ***hour***, ***pressure***, ***sea_pressure***, ***wind_direction***, ***wind_speed***, ***temperature***, ***rel_humidity***, ***precipitation***)

Esta tabla contiene los datos meteorológicos (como por ejemplo la *temperatura* y las *precipitaciones*)de cada una de las fechas contenidas en intervalos de 3 horas dentro del conjunto de entrenamiento.

4.2.2.6 Tabla *travel_time_intersection_to_tollgate*

El esquema de la tabla original proporcionado por la competición es la siguiente:

travel_time_intersection_to_tollgate (***intersection_id***, ***tollgate_id***, ***time_window***, ***avg_travel_time***)

Esta tabla se obtiene como resultado de la ejecución del script ***aggregate_travel_time.py*** proporcionado por la competición sobre la tabla *vehicle_trajectories_training* de tal forma que, a partir de esta tabla, para cada una de las rutas y ventana de tiempo de 20 minutos en el conjunto de entrenamiento, se agrupan los datos, se calcula el tiempo medio de viaje en esa ruta e intervalo de tiempo y se almacena en la tabla *travel_time_intersection_to_tollgate*.

4.2.2.7 Tabla *traffic_volume_tollgates*

El esquema de la tabla original proporcionado por la competición es la que se expone a continuación:

traffic_volume_tollgates (***tollgate_id***, ***time_window***, ***direction***, ***volume***)

Esta tabla se obtiene como resultado de la ejecución del script ***aggregate_volume.py*** proporcionado por la competición sobre la tabla *traffic_volume_tollgates_training* de manera que, a partir de esta tabla, para cada uno de los pares **barrera de peaje-dirección** e intervalo de tiempo de 20 minutos en el conjunto de entrenamiento, se agrupan los datos, se calcula el volumen de tráfico en ese par y ventana de tiempo y se almacena en la tabla *traffic_volume_tollgates*.

4.2.3 Base de datos con los tipos de datos modificados

4.2.3.1 Tabla *road_links_modified*

El esquema de la tabla se mantiene igual con respecto a la tabla de la base de datos originales exceptuando los tipos de algunos de los atributos: Uno de los cambios que se realizó fue cambiar el tipo de dato de la columna *link_id* a **smallint**. Por otra parte, los tipos de las columnas *in_top* y *out_top* se establecieron como **arrays de smallint** puesto que sus valores son conjuntos de enlaces.

4.2.3.2 Tabla *vehicle_routes_modified*

El esquema de la tabla es el mismo que en la tabla original solo que en esta tabla se cambió el tipo de la columna *tollgate_id* a **smallint** y la columna *link_seq* a **smallint**[].

4.2.3.3 Tabla *vehicle_trajectories_training_modified*

El esquema de la tabla permanece intalterable con respecto a la misma tabla en la base de datos originales a excepción de que en esta tabla se modificó los tipos de las columnas *tollgate_id* (a **smallint**), *vehicle_id* (a **int**) y *travel_seq* (a **link_object**[]). Esta última columna tiene un tipo compuesto con el objetivo de acceder mejor a la información, formado por los siguientes atributos:

- ***id*** : Identificador del enlace (**smallint**).
- ***entrance_time***: Momento del tiempo en el que el vehículo entra en ese enlace (**timestamp**).
- ***duration***: Tiempo que pasa el vehículo atravesando dicho enlace en segundos (**float**).

4.2.3.4 Tabla *traffic_volume_tollgates_training_modified*

El esquema de la tabla es el mismo que la tabla general. No obstante, en esta tabla se eliminaron las columnas *vehicle_model* y *vehicle_type* puesto que no son relevantes a la hora de realizar las predicciones pertinentes. Por otra parte, se modificaron los tipos de los atributos *tollgate_id* (a **smallint**), *direction* (a **smallint**) y *has_etc* (a **boolean**).

4.2.3.5 Tabla *weather_data_modified*

El esquema de la tabla no se modificó debido a que los tipos de las columnas en la tabla original son los correctos. Sin embargo, se alteraron aquellas filas en las que la columna *wind_direction* tenía el valor **999017** puesto que el valor que admite este atributo son **grados** y el valor debe estar entre *0* y *360* grados. La modificación que se procedió a realizar fue realizar *la media entre la dirección del viento del día anterior y del dia posterior*, con el objetivo de realizar una aproximación y no eliminar completamente una fila.

Por otro lado, se añadió una fila de una fecha que no existía en la tabla y

que es necesaria para combinarse con otras tablas (día *10/10/16*).

4.2.3.6 Tabla *travel_time_intersection_to_tollgate_modified*

En esta tabla se mantuvieron los mismos atributos que en la original, pero se modificaron los tipos de las columnas *tollgate_id* (a **smallint**) y *time_window* (a **timestamp ARRAY[2]**).

4.2.3.7 Tabla *traffic_volume_tollgates_modified*

El esquema de esta tabla se mantiene con respecto a la tabla original; sin embargo, se modificaron los tipos de las variables *tollgate_id* (a **smallint**), *time_window* (a **timestamp ARRAY[2]**) y *direction* (a **smallint**).

4.2.4 Base de datos con los datos relacionados con la fase de pruebas

Para separar la fase de entrenamiento de la fase de pruebas y estructurar mejor la información, se construyó una nueva base de datos para manejar los datos proporcionados por la competición para la fase de pruebas.

4.2.4.1 Tabla *travel_time_intersection_to_tollgate_test*

El esquema de la tabla es la siguiente:

travel_time_intersection_to_tollgate_test (***intersection_id***, ***tollgate_id***, ***time_window***, ***avg_travel_time***)

Para comprender el sentido de esta relación, es necesario observar la **Figura 4.2**. En esta tabla se especifica el tiempo promedio de viaje en intervalos de tiempo de 20 minutos contenidos dentro de los intervalos de tiempo de 2 horas previos a los intervalos de tiempo a predecir. En este caso, los datos proporcionados corresponden a los días desde el **18** al **24 de Octubre de 2016** en los intervalos de tiempo de **6:00 a 8:00** (2 horas antes del intervalo a predecir de 8:00 a 10:00) y de **15:00 a 17:00** (2 horas antes del intervalo a predecir de 17:00 a 19:00).

4.2.4.2 Tabla *traffic_volume_tollgates_test*

El esquema de la tabla es la que se expone a continuación:

traffic_volume_tollgates_test (***tollgate_id***, ***time_window***, ***direction volume***)

En esta tabla se proporciona el *volumen de tráfico* de cada una de las barreras de peaje en las direcciones de entrada y salida (menos la barrera de peaje 2 que no tiene dirección de salida) en intervalos de tiempo de 20 minutos contenidos dentro de los intervalos de tiempo de 2 horas previos a los intervalos de tiempo a predecir. Los datos proporcionados corresponden a los mismos días establecidos para la predicción del tiempo promedio de viaje.

4.2.4.3 Tabla *tabla_resultado_average_travel_time*

El esquema de la tabla es la siguiente:

tabla_resultado_average_travel_time(***intersection_id***, ***tollgate_id***, ***time_window***, ***avg_travel_time***)

Esta tabla contiene las predicciones realizadas con respecto al tiempo promedio de viaje en los intervalos que nos insta la competición a predecir en la *fase de testeo* (**Figura 4.2**).

4.2.4.4 Tabla *tabla_resultado_traffic_volume*

El esquema de la tabla es la que se expone a continuación:

traffic_volume_tollgates_test (***tollgate_id***, ***time_window***, ***direction volume***)

. Esta tabla contiene las predicciones realizadas con respecto al volumen de tráfico en los intervalos que nos insta la competición a predecir en la fase de testeo (**Figura 4.2**).

4.2.5 Base de datos con los datos reales de los valores a predecirse

Con el fin de verificar la precisión obtenida por el desempeño de las diferentes técnicas de minería de datos en las tareas de predicción tanto del *tiempo promedio de viaje* como del *volumen de tráfico*, procedimos a construir una base de datos en la que se alojaran los valores reales de aquellos intervalos de tiempo requeridos para pronosticar tales variables de tráfico. En los subsiguientes apartados se exponen las tablas que conforman dicha base de datos.

4.2.5.1 Tabla *travel_time_intersection_to_tollgate_real*

El esquema de la tabla es el mismo que el de la tabla *travel_time_intersection_to_tollgate_test*. Esta tabla contiene los datos reales del *tiempo promedio de viaje* en las ventanas de tiempo, rutas y días que nos insta la competición a predecir en la fase de pruebas (**Figura 4.2**).

4.2.5.2 Tabla *traffic_volume_tollgates_real*

El esquema de la tabla es la misma que el de la tabla *traffic_volume_tollgates_test*. Esta tabla contiene los datos reales del *volumen de tráfico* en las ventanas de tiempo, barreras de peaje y direcciones de conducción (entrada y salida) que nos insta la competición a predecir en la fase de pruebas (**Figura 4.2**).

Nota: Los esquemas detallados de todas las tablas utilizadas pueden ser accedidos a través de [26].

4.3 Creación de gráficas para visualizar los datos almacenados

Tras realizar un almacenado de los datos suministrados por la competición, es fundamental presentar dicha información en *gráficas* con el fin de facilitar la comprensión de los mismos y descubrir las relaciones subyacentes en la información contenida en ellos. En los siguientes apartados se visualizan distintos gráficos generados a partir de las tablas mencionadas anteriormente.

4.3.1 Gráficas del tiempo promedio de viaje de todos los días por rutas

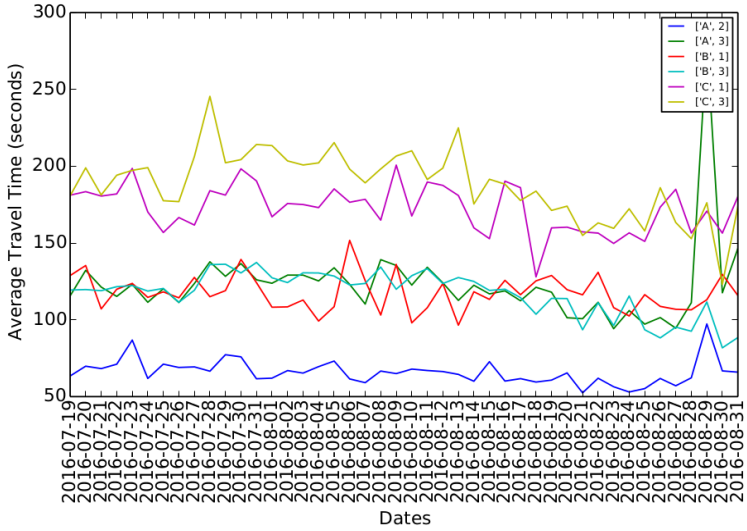


Figura 4.6: Tiempo promedio de viaje medio en cada uno de los días de entrenamiento de los meses de julio y agosto de 2016

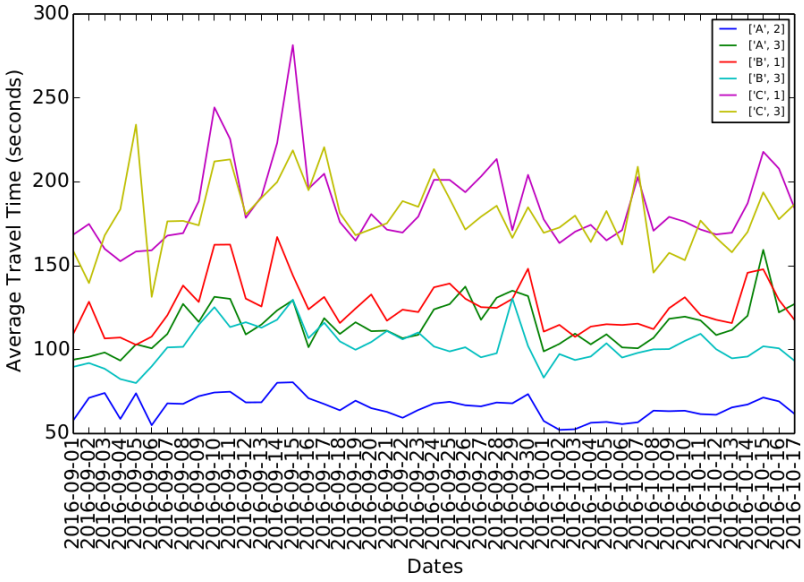


Figura 4.7: Tiempo promedio de viaje medio en cada uno de los días de entrenamiento de los meses de septiembre y octubre de 2016

En estas gráficas se representa el valor medio de los tiempos promedios de viaje de todos los intervalos de tiempo de 20 minutos de entrenamiento por cada uno de los días y rutas proporcionados por la competición. Como se puede apreciar en ellas, cada una de las rutas tiene su propio patrón de valores del tiempo promedio de viaje, de tal forma que en rutas como la ***C-1***

y ***C-3*** presentan valores mayores, mientras que las demás adquieren valores más bajos, siendo la ruta A-***2*** la que menos tarda en recorrerse. Este comportamiento de la gráfica es normal debido a que estos valores tienen que ver con la capacidad, anchura y longitud de las carreteras que las forman (**Figura 4.5**).

4.3.2 Gráficas del tiempo promedio de viaje de algunos días en todas las horas en cada una de las rutas

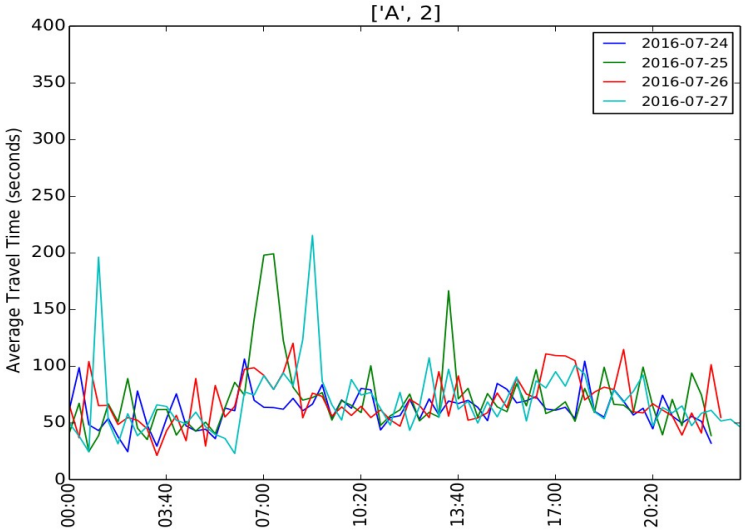


Figura 4.8: Tiempo promedio de viaje por horas en algunos días en la ruta A-2

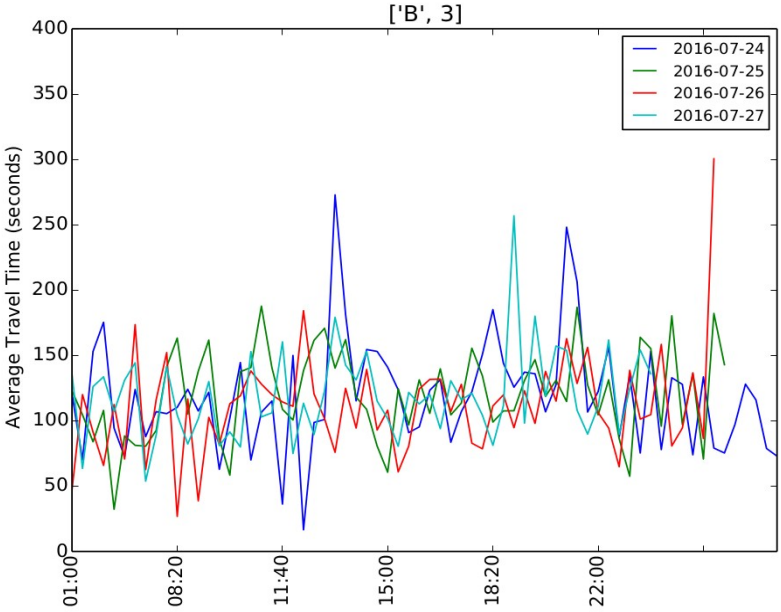


Figura 4.9: Tiempo promedio de viaje por horas en algunos días en la ruta B-3

En estas imágenes se muestra el tiempo promedio de viaje por horas de algunos días en dos rutas propuestas por la competición. A través de la superposición de las series temporales de varios días se puede contemplar que todas ellas se asemejan en forma debido a la propiedad de estacionalidad que presentan las series temporales relacionadas con el tráfico en las carreteras. No obstante, puesto que las características de tráfico poseen una naturaleza bastante estocástica (ya que son muchos los factores que influyen en el mismo y tienen una componente aleatoria) y hay una falta de datos de algunas intervalos de tiempo, las series tienen una serie de diferencias que dificultan el descubrimiento de un comportamiento totalmente estacional en las mismas.

4.3.3 Gráficas del volumen de tráfico de todos los días en todas las horas en cada una de los pares barrera de peaje-dirección

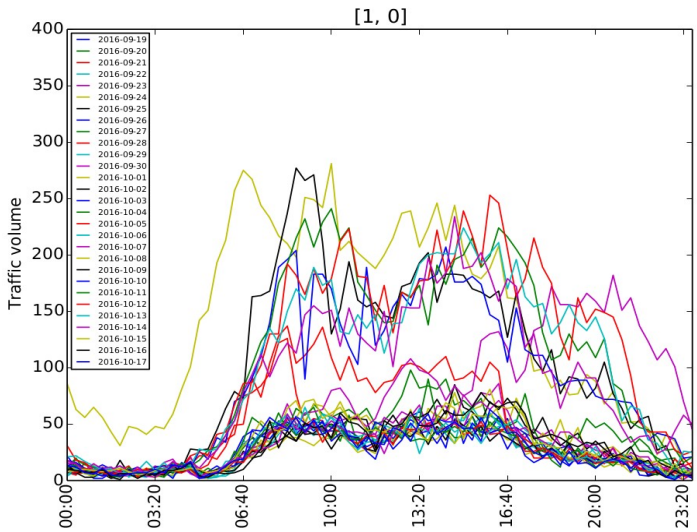


Figura 4.10: Barrera de peaje 1 en la dirección de entrada

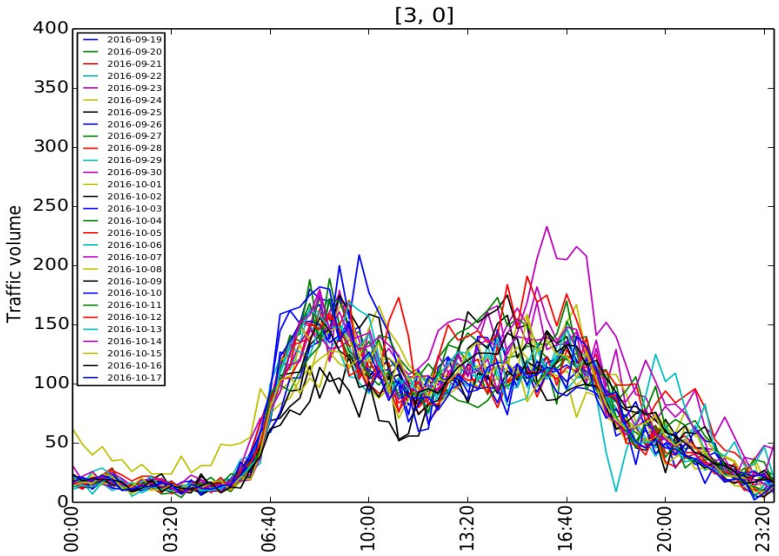


Figura 4.11: Barrera de peaje 3 en la dirección de entrada

Como podemos visualizar en las gráficas de ejemplo de dos pares *barrera de peaje-dirección* anteriores, cada uno de los pares de la competición posee un determinado patrón en cuanto al número de vehículos a lo largo de un día. No obstante, si nos fijamos detenidamente, se aprecia que en todas las gráficas la forma de las gráficas es la misma: en la primeras horas del día hay poco tráfico; en las primeras horas de la mañana hay un pico en el número de vehículos y disminuye en las últimas horas del horario de mañana y, a medida que avanza la tarde, el tráfico va disminuyendo. Esto se corresponde con un día típico.

En algunas gráficas (como la que se visualiza en la **Figura 4.10**) se observa que en algunos días se sigue el patrón del número de vehículos normal pero no la forma de las gráficas de la mayoría de los días. Aquellos días que poseen dicho comportamiento se considera *ruido* y es necesario eliminarlo para que esos valores no influyan negativamente a la hora de realizar las predicciones del volumen de tráfico.

4.4 Predicciones del tiempo promedio de viaje

El primer objetivo a conseguir propuesto por la competición es estimar el ***tiempo promedio de viaje*** que se va a producir en una serie de intervalos de 20 minutos en unos días determinados (según la **Figura 4.2**). Para llevar a cabo esto, se ha procedido a desarrollar una serie de aproximaciones en las que se aplican diferentes técnicas de aprendizaje

automático y se analiza y evalúa el comportamiento de los mismos sobre los datos contenidos en las bases de datos creadas para su tratamiento.

4.4.1 Primera aproximación: Algoritmos diversos de Machine Learning

La primera aproximación de predicciones del tiempo promedio de viaje desarrollada parte de la *generación de unas vistas minables* sobre las bases de datos de los datos modificados y de los datos de prueba. Con el propósito de ejecutar el primer conjunto de predicciones, se llevó a cabo la creación de una nueva estructura de los datos para pasarla como entrada a los algoritmos a utilizar para estimar valores. El esquema de dicha estructura es la siguiente:

vista_minable (***type_day***, ***twenty_min_previous***, ***forty_min_previous***, ***sixty_min_previous***, ***eighty_min_previous***, ***onehundred_min_previous***, ***onehundredtwenty_min_previous***, ***pressure***, ***sea_pressure***, ***wind_direction***, ***wind_speed***, ***temperature***, ***rel_humidity***, ***precipitation***, ***avg_travel_time***)

Esta nueva estructura de los datos almacenados en las bases de datos creadas se ha generado con el objetivo de que albergue los datos de entrenamiento y los datos de testeo para suministrar a las técnicas de aprendizaje automático pertinentes. Este esquema se sustenta en dos características principales: los *datos meteorológicos* y la *evolución del tiempo promedio de viaje durante las dos horas previas a la ventana de tiempo*. Por un lado, se tiene en cuenta el tiempo meteorológico puesto que es un factor que influye notablemente en el flujo de tráfico, de tal forma que si por ejemplo un día es muy lluvioso, entonces ese día el tráfico va a circular con menor velocidad debido a las medidas de precaución y el *tiempo promedio de viaje* será mayor. Por otro lado, si proporcionamos a los algoritmos de minería de datos la evolución de los cambios que sufre el tiempo promedio de viaje en las dos horas previas a cada una de las ventanas de tiempo, entonces éstos pueden hallar un patrón en los datos y realizar las tareas de predicción de forma más precisa. Además, en esta aproximación se tuvo en cuenta el *tipo de día* (si era laborable o fin de semana) puesto que no es lo mismo el tiempo promedio de viaje un día laborable en horario de mañana que un día en fin de semana en el mismo horario.

4.4.1.1 Creación de las vistas minables

Para poder crear el esquema de datos comentado anteriormente, primero fue necesario crear un script *sql* que creara las columnas que componen la evolución del tiempo promedio de viaje en las dos horas previas a la ventana de tiempo correspondiente en la **tabla *travel_time_intersection_to_tollgate_modified***. Concretamente, este script genera dichas columnas sobre los datos de entrenamiento. El bloque de código principal que las crea es el siguiente:

```
DO $$
<<block>>
DECLARE
  termina boolean DEFAULT FALSE;
  rutas_intervalos tipo_fila ARRAY;
  rutaintervalo_anterior tipo_fila;
  contador integer DEFAULT 1;
  routes ruta ARRAY;
  tiempos time ARRAY;
  route ruta;
  tiempo time;
BEGIN

rutas_intervalos := ARRAY(SELECT '(' ||intersection_id || ', ' || tollgate_id || ', ' || time_window[1] || ')' FROM  travel_time_intersection_to_tollgate_modified WHERE
(time_window[1].time BETWEEN TIME '08:00:00' AND TIME '09:40:00') OR (time_window[1].time BETWEEN TIME '17:00:00' AND TIME '18:40:00')
ORDER BY intersection_id, tollgate_id, time_window);

 WHILE contador <= ARRAY_LENGTH(rutas_intervalos, 1) LOOP
       termina := FALSE;
       PERFORM create_firstrow_route_interval(rutas_intervalos[contador]);
       rutaintervalo_anterior = rutas_intervalos[contador];
       contador = contador + 1;
        WHILE NOT(termina) LOOP
            IF (rutas_intervalos[contador].intersection = rutaintervalo_anterior.intersection AND rutas_intervalos[contador].tollgate = rutaintervalo_anterior.tollgate AND
(rutas_intervalos[contador].left_side_interval - rutaintervalo_anterior.left_side_interval) = INTERVAL '20 min') THEN
                PERFORM actualizar_filaactual_con_filaanterior(rutas_intervalos[contador], rutaintervalo_anterior);
                 rutaintervalo_anterior = rutas_intervalos[contador];
                contador := contador + 1;
            ELSE
                termina := TRUE;
            END IF;
        END LOOP;
  END LOOP;
```

Figura 4.12: Bloque principal de código que crea la evolución de las 2 horas previas a la ventana de tiempo correspondiente en los datos de entrenamiento

En primer lugar, obtenemos las diferentes rutas de la competición junto con las ventanas de tiempo comprendidas en los intervalos de tiempo que se nos pide predecir. Esto se lleva a cabo debido a que, en esta aproximación de predicciones, se van a tener en cuenta sólo aquellos datos de entrenamiento cuya ventana de tiempo esté contenida en los intervalos a predecir:

```
rutas_intervalos := ARRAY(SELECT '(' ||intersection_id || ', ' || tollgate_id || ', ' || time_window[1] || ')' FROM  travel_time_intersection_to_tollgate_modified WHERE
(time_window[1].time BETWEEN TIME '08:00:00' AND TIME '09:40:00') OR (time_window[1].time BETWEEN TIME '17:00:00' AND TIME '18:40:00')
ORDER BY intersection_id, tollgate_id, time_window);
```

Figura 4.13: Obtención de las rutas de tráfico junto con las ventanas de tiempo comprendidas en los intervalos de tiempo a predecir

A continuación, para cada uno de los pares *ruta-ventana de tiempo* obtenidos, se crean las columnas que indican la evolución del tiempo promedio de viaje las dos horas previas. Para ello, en la primera iteración se llama a una función (*create_firstrow_route_interval*) que crea el primer conjunto de valores de estas columnas para el primer par *ruta-ventana de tiempo*. Una vez hecho esto, en la siguiente iteración, si la ruta del nuevo par *ruta-ventana de tiempo* corresponde con el de la anterior iteración y las ventanas de tiempo se diferencian en 20 minutos (es decir, son intervalos de tiempo contiguos), entonces se crean los valores de las columnas con datos de la anterior iteración (con la función *actualizar_filaactual_con_filaanterior*). Esto se lleva a cabo debido a que, por ejemplo, el valor de la columna que indica el *tiempo promedio de viaje 60 minutos previos* con respecto a la ventana de tiempo de la actual iteración corresponde con el valor de la columna que indica el *tiempo promedio de viaje 40 minutos previos* a la ventana de tiempo de la iteración anterior.

Tras varias iteraciones, como hemos obtenido los datos de los intervalos a predecir (***8:00 – 10:00*** y de ***17:00 a 19:00***) y éstos están ordenados por la ventana de tiempo, en la siguiente iteración ya los intervalos no avanzan de 20 en 20 minutos puesto que pasamos de la franja horaria de *8:00-10:00* a 17*:00-19:00*. También se puede dar el caso de que haya un intervalo de tiempo de 20 minutos que falte en esas franjas horarias debido a la falta de datos proporcionados por la competición. En estas dos situaciones, por lo tanto, como no tenemos filas previas con las que rellenar las columnas de la actual iteración (debido al cambio de franja horaria), es necesario volver a rellenar las columnas con la función *create_firstrow_route_interval* para crear el primer conjunto de valores de estas columnas en el primer intervalo de tiempo de 20 minutos de dicha franja horaria. A partir de este paso el proceso anterior se repite:

```
WHILE contador <= ARRAY_LENGTH(rutas_intervalos, 1) LOOP
      termina := FALSE;
      PERFORM create_firstrow_route_interval(rutas_intervalos[contador]);
      rutaintervalo_anterior = rutas_intervalos[contador];
      contador = contador + 1;
       WHILE NOT(termina) LOOP
          IF (rutas_intervalos[contador].intersection = rutaintervalo_anterior.intersection AND rutas_intervalos[contador].tollgate = rutaintervalo_anterior.tollgate AND  (rutas_intervalos
[contador].left_side_interval - rutaintervalo_anterior.left_side_interval) = INTERVAL '20 min') THEN
              PERFORM actualizar_filaactual_con_filaanterior(rutas_intervalos[contador], rutaintervalo_anterior);
               rutaintervalo_anterior = rutas_intervalos[contador];
              contador := contador + 1;
          ELSE
              termina := TRUE;
          END IF;
      END LOOP;
  END LOOP;
```

Figura 4.14: Estructura iterativa para crear las columnas relacionadas con la evolución del tráfico en las dos horas previas a la ventana de tiempo considerada

Como se comentó anteriormente, para rellenar las columnas mencionadas previamente, los valores del primer intervalo de 20 minutos de cada una de las franjas horarias de los distintos días de entrenamiento se establecen con la función *create_firstrow_route_interval*. Para entender cómo se rellena cada una de estas columnas, se visualiza la siguiente imagen:

```
UPDATE  travel_time_intersection_to_tollgate_modified AS thistable
         SET twenty_min_previous = othertable.avg_travel_time
        FROM travel_time_intersection_to_tollgate_modified othertable
        WHERE othertable.time_window[1] = (rutaintervalo.left_side_interval - INTERVAL '20 minute') AND othertable.time_window[2] = (rutaintervalo.left_side_interval)
        AND othertable.intersection_id = rutaintervalo.intersection AND othertable.tollgate_id = rutaintervalo.tollgate AND
        thistable.time_window[1] = rutaintervalo.left_side_interval AND  thistable.time_window[2] = (rutaintervalo.left_side_interval + INTERVAL '20 minute')
        AND  thistable.intersection_id = rutaintervalo.intersection AND  thistable.tollgate_id = rutaintervalo.tollgate;

        PERFORM checkAttributeValue(array['twenty_min_previous','20']::text[], rutaintervalo);
```

Figura 4.15: Consulta SQL que rellena la columna del tiempo promedio de viaje 20 minutos antes de la ventana de tiempo considerada

En esta consulta SQL se actualiza el valor de la columna que aloja el *tiempo promedio de viaje* de los 20 minutos previos a la ventana de tiempo que se considere en ese momento buscando aquella fila de la tabla con los datos de entrenamiento del *tiempo promedio de viaje* que se corresponda con el intervalo de tiempo que coincide con los 20 minutos previos y obteniendo el valor de su *tiempo promedio de viaje*. Tras esto, se comprueba si el valor establecido en la columna no es **nulo** y esta comprobación se realiza debido a dos razones. Por un lado, al intentar buscar el valor, puede ocurrir que no haya datos en la tabla que correspondan a un intervalo de tiempo pasado ya que no se han proporcionado más datos.

Por otra parte, nos hemos percatado de que, al construir la **tabla *travel_time_intersection_to_tollgate_modified***, hay intervalos de tiempo que no existen en la tabla y, por tanto, no disponemos de su *tiempo promedio de viaje* para rellenar el valor de estas columnas. En el caso de que ocurriera alguna de estas dos circunstancias, esta columna adquiría el valor medio de los tiempos promedios de viaje de aquellas filas de la tabla cuyo intervalo de tiempo y el tipo de día (día de la semana) fuera el mismo que el de la fila en consideración. Por ejemplo, si la fila tomada en consideración corresponde a un **lunes** en el intervalo de tiempo **9:20-9:40** y la columna que corresponde al *tiempo promedio de viaje* 20 minutos antes de esa ventana de tiempo no se puede rellenar debido a que el dato no existe, entonces se establece el valor medio de los tiempos promedios de viaje de aquellas filas de entrenamiento que tengan el mismo intervalo de tiempo y el día sea un lunes.

El procedimiento explicado anteriormente se sigue para los demás valores de la evolución del *tiempo promedio de viaje* en las dos horas previas a

la ventana de tiempo que se esté teniendo en consideración.

Una vez llevado a cabo este proceso, para rellenar los siguientes intervalos de tiempo de 20 minutos a predecir contiguos a la ventana de tiempo rellenada con la función anterior (por ejemplo, rellenar los valores de las columnas de los intervalos de tiempo de 20 minutos comprendidos entre las **8:20** y las **10:00** a partir del intervalo **8:00-8:20**), se realiza lo siguiente:

```
UPDATE  travel_time_intersection_to_tollgate_modified AS actual
            SET twenty_min_previous = before.avg_travel_time,
                forty_min_previous = before.twenty_min_previous,
                sixty_min_previous = before.forty_min_previous,
                eighty_min_previous = before.sixty_min_previous,
                onehundred_min_previous = before.eighty_min_previous,
                onehundredtwenty_min_previous = before.onehundred_min_previous
            FROM  travel_time_intersection_to_tollgate_modified before
            WHERE actual.intersection_id = rutaintervalo_actual.intersection AND actual.tollgate_id = rutaintervalo_actual.tollgate AND actual.time_window[1]=
rutaintervalo_actual.left_side_interval AND before.intersection_id = rutaintervalo_anterior.intersection AND before.tollgate_id = rutaintervalo_anterior.tollgate AND before.time_window[1] =
rutaintervalo_anterior.left_side_interval;
```

Figura 4.16: Consulta SQL que establece los valores de las columnas de la evolución del tiempo promedio de viaje de las 2 horas previas a la ventana de tiempo en consideración utilizando los valores de las columnas del anterior intervalo de tiempo

En esta consulta SQL se accede a los valores de las columnas del intervalo de tiempo de 20 minutos previos a la ventana de tiempo en consideración para actualizar las columnas del intervalo de tiempo actual.

Tras rellenar los valores de las columnas que corresponden a la evolución del *tiempo promedio de viaje* en las dos horas previas a cada uno de los intervalos de tiempo a predecir (en los datos de entrenamiento), se procedió a ejecutar lo que se visualiza en la siguiente imagen:

```
CREATE OR REPLACE VIEW tiempo_con_intervalos AS SELECT *
FROM weather_data_modified JOIN (SELECT *
FROM travel_time_intersection_to_tollgate_modified
WHERE (time_window[1].time BETWEEN TIME '08:00:00' AND TIME '09:40:00')  OR (time_window[1].time BETWEEN TIME '17:00:00' AND TIME '18:40:00')
ORDER BY intersection_id, tollgate_id, time_window
) t ON date_ = time_window[1].date AND CEIL(EXTRACT(HOUR FROM time_window[1])/3) * 3 = hour
ORDER BY intersection_id, tollgate_id, time_window;
```

Figura 4.17: Combinación de la tabla con los datos de entrenamiento del tiempo promedio de viaje junto con los datos meteorológicos

En esta consulta SQL se combinaron aquellas filas de la **tabla *travel_time_intersection_to_tollgate_modified*** (que contienen las columnas creadas) cuyos intervalos de tiempo correspondían con aquellas ventanas de tiempo a predecir con los datos meteorológicos de esos días (vista

tiempo_con_intervalos).

Por último, para crear las vistas minables que se suministran como datos de entrenamiento a los algoritmos de minería de datos, se ejecutó el código de la imagen que se muestra a continuación:

```
FOREACH route IN ARRAY routes LOOP
    FOREACH tiempo IN ARRAY tiempos LOOP
       EXECUTE('CREATE TABLE  ' || route.intersection ||'_' ||route.tollgate || '_' ||EXTRACT(HOUR FROM tiempo) || '_' ||EXTRACT(MINUTE FROM tiempo) || ' AS
              SELECT EXTRACT(isodow FROM time_window[1].date) AS type_day, twenty_min_previous, forty_min_previous, sixty_min_previous, eighty_min_previous, onehundred_min_previous,
onehundredtwenty_min_previous, pressure,sea_pressure,wind_direction,wind_speed,temperature,rel_humidity,precipitation,avg_travel_time FROM tiempo_con_intervalos  WHERE intersection_id = ''' ||
route.intersection|| ''' AND tollgate_id = '|| route.tollgate || 'AND time_window[1].time = ''' ||
              tiempo || ''' ORDER BY intersection_id, tollgate_id, time_window');

       EXECUTE('UPDATE ' || route.intersection ||'_' ||route.tollgate || '_' ||EXTRACT(HOUR FROM tiempo) || '_' ||EXTRACT(MINUTE FROM tiempo) || ' SET type_day = 1
              WHERE type_day BETWEEN 1 AND 5');
       EXECUTE('UPDATE ' || route.intersection ||'_' ||route.tollgate || '_' ||EXTRACT(HOUR FROM tiempo) || '_' ||EXTRACT(MINUTE FROM tiempo) || ' SET type_day = 0
              WHERE type_day IN (6,7)');
    END LOOP;
END LOOP;
```

Figura 4.18: Código SQL que crea una vista para cada ruta e intervalo a partir de la vista con los datos combinados

En este código SQL, se genera una vista por cada uno de los pares *ruta-intervalo de tiempo* a predecir a partir de la vista *tiempo_con_intervalos*. Esta creación de distintas vistas por cada ruta e intervalo de tiempo a estimar se realiza debido a que, si se quiere predecir el *tiempo promedio de viaje* en una ruta e intervalo de tiempo determinado, es intuitivo pensar que van tener una gran influencia los datos de entrenamiento que se correspondan con esa ruta e intervalo de tiempo.

El procedimiento visto para crear las vistas minables necesarias como entrada de entrenamiento de los algoritmos de minería de datos en esta primera aproximación se aplica también a los datos que corresponden a las 2 horas previas a los intervalos a predecir, cuyas vistas se unen a las vistas de entrenamiento previos (esto se verá más detalladamente más adelante). Por otra parte, también se crean las vistas con la estructura de la vista *tiempo_con_intervalos* para las ventanas de tiempo a predecir con el objetivo de que el algoritmo pertinente obtenga las mismas columnas que los datos de entrenamiento.

4.4.1.2 Realización de predicciones a partir de las vistas

Para llevar las estimaciones oportunas del *tiempo promedio de viaje*, se ha procedido a utilizar diversos algoritmos de minería de datos vistos en apartados anteriores; estos algoritmos son *XGBoost*, *LightGBM*, *Regresión Lineal*, *Redes Neuronales*, *Máquinas de Soporte Vectorial* y *k-Vecinos Más*

Cercanos.

El primer paso para poder utilizar estas técnicas es entrenarlas con datos de entrenamiento. Como mencionamos anteriormente, para cada uno de las rutas e intervalos a predecir, se ha generado una vista con el fin de proporcionar a los algoritmos de minería de datos el conjunto de datos de entrenamiento que corresponde a cada ruta e intervalo. Por lo tanto, para cada par *ruta-intervalo*, se han seguido los siguientes pasos:

- En primer lugar se accede a la base de datos ***tfgdatosmodificados*** para obtener la vista que contiene los datos de entrenamiento correspondientes a la ruta y al intervalo que estamos considerando:

```
conn = psycopg2.connect("dbname='tfgdatosmodificados' user='javisunami' host='localhost' password='javier123'")
cur = conn.cursor()
query = "select * from " + route[0].lower() + "_" + str(route[1]) + "_" + str(interval.hour) + "_" + str(interval.minute) + ";"
cur.execute(query)
rows = cur.fetchall()
dataframe_traveltime = pd.DataFrame(rows, columns=colnames)
X_train = dataframe_traveltime.iloc[:, 0:14]
y_train = dataframe_traveltime.iloc[:, 14]
```

Figura 4.19: Obtención de los datos de entrenamiento para la ruta e intervalo en consideración

- A continuación se accede a la base de datos ***tfgtest*** para obtener la vista que contiene los datos de los diferentes días de predicción en el par *ruta-intervalo* del que se obtuvieron los datos de entrenamiento anteriormente:

```
conn = psycopg2.connect("dbname='tfgtest1' user='javisunami' host='localhost' password='javier123'")
cur = conn.cursor()
query = "select * from " + route[0].lower() + "_" + str(route[1]) + "_" + str(interval.hour) + "_" + str(interval.minute) + ";"
cur.execute(query)
rows = cur.fetchall()
dataframe_traveltime = pd.DataFrame(rows, columns=colnames)
X_test = dataframe_traveltime.iloc[:, 0:14]
```

Figura 4.20: Obtención de los datos de prueba para la ruta e intervalo en consideración

- Después realizamos la fase de entrenamiento y de predicción para cada uno de los modelos de aprendizaje automático contemplados previamente:

```
model = XGBRegressor()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Figura 4.21: Entrenamiento y predicción del tiempo promedio de viaje en los distintos días de predicción con el modelo XGBoost

- Calculamos el error de estimación de cada una de las técnicas con la fórmula mencionada en la **Figura 4.3**. Para ello, para cada algoritmo, primero calculamos el *error correspondiente a los días de predicción para una ruta e intervalo determinado* (tercer sumatorio de la fórmula):

```
for index, fila in travel_time_dataframe.iterrows():
        y_test_sum += abs((fila['avg_travel_time'] - y_pred[fila['date'].day - 18]) / fila['avg_travel_time'])
y_test_sum /= len(travel_time_dataframe);
```

Figura 4.22: Cálculo del error medio de los días de predicción para una ruta e intervalo determinado

Este error se acumula para cada uno de los *intervalos de una ruta* con el objetivo de calcular el *error correspondiente a los intervalos de una ruta* (segundo sumatorio):

```
errores_predicciones_intervalos["XGBoost"] += y_test_sum
```

Figura 4.23: Acumulación de los errores correspondiente a los días a predecir de cada uno de los intervalos a estimar

```
errores_predicciones_intervalos[key]/len(time_intervals)
```

Figura 4.24: Cálculo del segundo sumatorio de la fórmula del error MAPE para un algoritmo

Nota: El key es cualquier algoritmo contemplado, en este caso XGBoost.

Una vez hecho esto se acumula, a su vez, el *error relacionado con cada una de las rutas de la competición* (primer sumatorio). Esta acumuluación va sumando los errores del *segundo sumatorio* :

```
errores_predicciones_rutas[key] += errores_predicciones_intervalos[key]/len(time_intervals)
```

Figura 4.25: Acumulación de los errores del segundo sumatorio de la fórmula del error MAPE

```
errores_predicciones_rutas[key] = errores_predicciones_rutas[key]/len(routes);
```

Figura 4.26: Cálculo del primer sumatorio de la fórmula del error MAPE para un algoritmo

A la hora de calcular el *error MAPE* para cada uno de los intervalos y rutas de la competición, se tuvo que realizar una pequeña modificación. La causa de dicha modificación consistía en que la competición no proporcionaba los valores del *tiempo promedio de viaje* de todas las rutas e intervalos en los días predecir. Es decir, la competición proporcionaba datos de tráfico de los días a estimar pero, al agrupar dichos datos en intervalos de 20 minutos había intervalos de los que no se disponían datos. Por lo tanto, a la hora de comparar las predicciones con los valores reales, se compararon aquellos intervalos de los que se disponía el valor real, dejando sin equiparar aquellos de los que no había datos reales.

Para cada una de las técnicas de minería de datos utilizadas se han dado los siguientes ***errores MAPE:***

XGBoost	**KNN**	**LightGBM**	**MLP**	**SVR**	**Linear Regression**
0.2416	0.2107	0.2194	0.4639	0.2065	0.2613

Tabla 4.1: Errores MAPE de la primera aproximación de predicciones del tiempo promedio de viaje

Como se puede apreciar en los valores de *error MAPE* anteriores, el algoritmo de aprendizaje automático que ha experimentado mejor rendimiento ha sido las *Máquinas de Soporte Vectorial aplicadas a la Regresión*, con un error de ***0.2065***. No obstante, todos los algoritmos han estado prácticamente en la misma línea, exceptuando las *Redes Neuronales.* Este modelo computacional, para que proporcione buenos resultados, requiere de una considerable cantidad de datos, hecho que no está presente en esta aproximación.

Con respecto a estos errores, cabe señalar que las técnicas de minería de datos utilizadas se ven afectadas, al igual que las *Redes Neuronales*, por el limitado número de instancias de entrenamiento que se les proporciona, a lo que se añade la falta de datos de tráfico de una serie de intervalos de tiempo en el conjunto de entrenamiento, lo que provoca que éste tenga menor tamaño: no superan las **100** instancias.

4.4.2 Segunda aproximación: ARIMA

Para realizar la segunda aproximación de predicciones del tiempo promedio de viaje se ha empleado un modelo estadístico para la *predicción de series temporales* denominado ***ARIMA***. Para aplicar dicho modelo a los datos de tráfico proporcionados, se ha escogido crear dos modelos *ARIMA* para cada una de las rutas, días y franjas de tiempo de 2 horas a predecir. La realización de estimaciones mediante este modelo se explica con detalles en los subsiguientes apartados.

4.4.2.1 Preparación de los datos

Con el objetivo de comparar las estimaciones que lleve a cabo cada uno de los modelo *ARIMA* a construir, es imprescindible tener los valores reales del *tiempo promedio de viaje*. Para ello, el primer paso para efectuar las predicciones es obtener dichos valores reales de la base de datos oportuna:

```
cur = conn.cursor()
cur.execute("""SELECT * FROM travel_time_intersection_to_tollgate_training2 WHERE (time_window[1].time BETWEEN TIME '08:00:00' AND TIME '09:40:00') OR (time_window[1].time
BETWEEN TIME '17:00:00' AND TIME '18:40:00') ORDER BY intersection_id, tollgate_id, time_window """)
rows = cur.fetchall()
colnames = ['intersection_id', 'tollgate_id', 'time_window', 'avg_travel_time']
```

Figura 4.27: Obtención de los valores reales del tiempo promedio de viaje para las rutas e intervalos de tiempo a predecir

Para poder aplicar el modelo *ARIMA* sobre nuestros datos, es necesario proporcionarle como entrada una ***serie temporal***; es decir suministrarle como entrada datos de entrenamiento del *tiempo promedio de viaje* en intervalos de tiempo ordenados cronológicamente. No obstante, de forma similar a lo que sucedía con la falta de datos de tráfico reales en las rutas e intervalos mencionados en la primera aproximación de predicciones, en los datos de entrenamiento también hay ventanas de tiempo de los que no existen datos del *tiempo promedio de viaje*. Por lo tanto, para proporcionarle al modelo una entrada de entrenamiento coherente, se procedió a rellenar los intervalos faltantes con datos medios de los demás días de entrenamiento. Este código establece la **media** de los valores de la serie temporal cuya ventana de tiempo es igual al intervalo de tiempo que se quiere añadir a la serie temporal:

```
minimum_date = min(df1.date)
maximum_date = max(df1.date)
date_aux = minimum_date
while (date_aux != maximum_date):
       if (not((date_aux == df1['date']).any())):
         valores_avg_travel = []
         for row in df1.values:
                if (row[0].time() == date_aux.time()):
                       valores_avg_travel.append(row[1])
         df1.loc[len(df1)] = [date_aux, np.mean(valores_avg_travel)]
       date_aux += datetime.timedelta(minutes=20)
df1 = df1.sort_index()
```

Figura 4.28: Cálculo del valor del tiempo promedio de viaje de una parte de aquellas rutas e intervalos de las que no disponemos datos

4.4.2.2 Realización de estimaciones

A continuación, para una *ruta* y *día* concretos, se procede a realizar las estimaciones en los intervalos de tiempo a predecir. Es decir, como se trata de predicciones de series temporales, se deben hacer por cada par *ruta-día* para estimar la evolución de la serie temporal en las ventanas de tiempo de dicho par. Para ello, por cada par *ruta-día*, se crean dos modelos *ARIMA*: uno para predecir los datos de la serie temporal en los intervalos de tiempo de 20 minutos incluidos en la ventana de tiempo ***8:00-10:00*** y otro modelo para estimar aquellos datos en los intervalos de tiempo de 20 minutos incluidos en la ventana de tiempo **17:00-19:00**. Para llevar a cabo esto, se realiza lo siguiente:

- De los días y rutas en los que se quiere estimar el *tiempo promedio de viaje*, la competición solo nos proporciona datos reales de los intervalos de tiempo de 20 minutos incluidos en las 2 horas previas a los intervalos a predecir. Al disponer de estos datos, por cada uno de los días, rutas e intervalos a predecir, se añaden los datos de esas 2 horas previas a los datos de entrenamiento correspondientes a una ruta determinada:

```
try:
   conn = psycopg2.connect("dbname='tfgtest1' user='javisunami' host='localhost' password='javier123'")
except:
   print("I am unable to connect to the database")
cur = conn.cursor()
query = "select time_window[1], avg_travel_time from travel_time_intersection_to_tollgate_test1 where intersection_id = '"+ str(route[0]) +"' AND tollgate_id = " + str(route[1])
+ " AND (time_window[1].time BETWEEN " + intervalo1 + ") AND (time_window[1].date = DATE '"+str(day)+"') order by time_window;"
cur.execute(query)
rows = cur.fetchall()
df2 = pd.DataFrame.from_records(rows, columns=['date','avg_travel_time'])
result_dataframe = pd.concat([df1_aux,df2])
```

Figura 4.29: Concatenación de los datos de entrenamiento de una ruta determinada con los datos reales de las dos horas previas a un intervalo a predecir en un día y ruta determinada

- Una vez realizado este paso, se prueban diferentes modelos *ARIMA*

para realizar las predicciones de un intervalo de tiempo de 2 horas a predecir en un día y ruta determinadas. El paso llevado a cabo previamente es necesario para que el modelo *ARIMA* pueda realizar las estimaciones de los siguientes 6 valores de la serie a partir de los datos reales de las dos horas previas a los intervalos a predecir. Estos 6 valores de la serie temporal son los 6 intervalos de 20 minutos incluidos en la ventana de tiempo de 2 horas a estimar:

```
for p in range(3,10):
  for d in range(3):
    for q in range(5):
        print("ORDEN : ", (p,d,q))
        orderr = (p,d,q)
        try:
                model = ARIMA(serie, order=orderr)
                model_fit = model.fit(disp=0)
                forecast = model_fit.forecast(steps=6)[0]
                new_forecast=[]
                for element in rows2:
                        new_forecast.append(forecast[((datetime.datetime(2018,1,1,element[0].hour,element[0].minute, 0)-hora_de_referencia)/1200).seconds])
                mse = mean_squared_error(valores_reales, new_forecast)
                print("MSE : ", mse, " BEST_SCORE: ", best_score)
                if mse < best_score:
                      print("BEST_SCORE : ", best_score)
                      best_score, best_cfg = mse, orderr
        except:
                continue
print('Best ARIMA%s MSE=%.3f' % (best_cfg, best_score))
```

Figura 4.30: Cálculo del mejor modelo ARIMA para una ruta, día y ventana de tiempo a estimar

Para elegir el mejor modelo *ARIMA* que se adapta a cada ruta, día e intervalo de 2 horas a predecir se comparan las predicciones llevadas a cabo con los valores reales. Debido a la falta de algunos valores reales del *tiempo promedio de viaje* en los intervalos a predecir, solo se ha podido tener en cuenta el error de predicción en aquellas estimaciones de las que disponíamos datos verídicos.

- Por último, se ha procedido a calcular el *error MAPE* de las predicciones desarrolladas por cada uno de los modelos *ARIMA* construidos para cada ruta, día e intervalo que se requerían estimar. El ***error MAPE*** generado por los valores pronosticados es **0.163**.

Como habíamos comentado con anterioridad, por cada par *ruta-día* se modelan dos modelo ARIMA: uno para predecir los datos de la serie temporal en los intervalos de tiempo de 20 minutos incluidos en la ventana de tiempo ***8:00-10:00*** y otro para las ventanas de tiempo de 20 minutos presentes en la franja de tiempo ***17:00-19:00***. De esta forma, se crea un modelo ARIMA que identifica la progresión del tiempo del *volumen de tráfico* de cada *ruta-día-intervalo de tiempo* (8:00-10:00 y 17:00-19:00).

Tras alcanzar un grano fino en cuanto a la creación de modelos ARIMA, se obtiene un *error MAPE* más pequeño que en cualquiera de los algoritmos utilizados en la primera aproximación de predicciones puesto que se identifican pequeños patrones dentro de los datos que los diferentes modelos ARIMA consiguen modelar de forma más óptima que el empleo de técnicas de minería de datos de forma más genérica. Aparte de esto, a la hora de construir los modelos ARIMA, se entrenan los modelos con una mayor cantidad de datos de entrenamiento puesto que, en lugar de agrupar los datos por ruta, día e intervalo de tiempo como se llevo a cabo en la primera aproximación, los modelos ARIMA son entrenados con la progresión temporal de una determinada ruta a lo largo del tiempo, lo que propicia una comprensión de los datos más precisa.

4.4.3 Tercera aproximación: Ventana deslizante

Para desarrollar la tercera aproximación de estimaciones del tiempo promedio de viaje se ha convertido la serie temporal de tiempos promedios de viaje proporcionada por la competición en un problema de *aprendizaje supervisado*. Esta transformación de los datos de la serie temporal nos permite acceder al conjunto de algoritmos estándar de aprendizaje de máquinas lineales y no lineales que hemos visto en apartados anteriores. Para ejecutar dicha conversión, se ha hecho uso de un método denominado ***ventana deslizante***, que se expone con detalle en los siguientes apartados.

4.4.3.1 Preparación de los datos

Para poder aplicar el método de la *ventana deslizante* sobre los datos temporales, es imprescindible añadir a dichos datos aquellos intervalos de tiempo de los que no se dispone ningún dato sobre el tiempo promedio de viaje en los mismos. Para ello, es fundamental aplicar la transformación vista en la **Figura 4.28**.

```
            except:
                print("I am unable to connect to the database")
            cur = conn.cursor()
            query = "select time_window[1], avg_travel_time from travel_time_intersection_to_tollgate_test1  where intersection_id = '" + str(route[0]) + "' AND tollgate_id = " + str(route[1]) +
" AND extract(day from time_window[1]) = " + str(day) + " AND extract(hour from time_window[1]) BETWEEN " + str(interval[0]) + " AND " + str((interval[1] - 1)) + " order by time_window;"
            cur.execute(query)
            row_2hoursintervals_before = cur.fetchall()
            dates_traveltime_2hoursintervals_before = pd.DataFrame.from_records(row_2hoursintervals_before, columns=['date','avg_travel_time'])
            dates_traveltime_filled = pd.concat([dates_traveltime_filled,dates_traveltime_2hoursintervals_before])
```

Figura 4.31: Concatenación de las dos horas previas a los intervalos de tiempo a predecir un día determinado con la serie temporal de una ruta

A continuación, concatenamos a los tiempos promedios de viaje de la serie temporal de una ruta determinada aquellos datos de las dos horas previas de un día a predecir, de manera que poseemos una serie temporal que alcanza hasta el primer intervalo de tiempo a estimar en ese día:

Tras este paso, se procede a utilizar el método de la *ventana deslizante.* Este método consiste en, dada una secuencia de números para un conjunto de datos de series temporales, reestructurar los datos para que parezcan un *problema de aprendizaje supervisado*. Podemos llevar a cabo esto usando **instantes de tiempo anteriores** como *variables de entrada* y usar el **siguiente instante de tiempo** como variable de salida. Para ello, debemos elegir un determinado *retraso de tiempo* para establecer el número de instantes de tiempo hacia detrás que se van a emplear como atributos explicativos. Tras probar con diferentes *retrasos de tiempo*, el que mejor resultado nos dio fue aquél con valor **5**:

```
dates_traveltime_supervised = pd.DataFrame()
number_time_steps_previous = 5
for i in range(number_time_steps_previous,0,-1):
      dates_traveltime_supervised['t-'+str(i)] = series_dates_traveltime_filled.shift(i)
dates_traveltime_supervised['t'] = series_dates_traveltime_filled .values
dates_trafficvolume_supervised = dates_trafficvolume_supervised[number_time_steps_previous:]
```

Figura 4.32: Serie temporal a problema de aprendizaje supervisado

En la última línea de la **Figura 4.32** se eliminan las primeras 5 líneas debido a que, como hemos elegido un retraso de tiempo de 5 instantes de tiempo, los primeros 5 intervalos de tiempo van a tener datos faltantes puesto que no se disponen de más datos hacia atrás (por ejemplo el cuarto intervalo de tiempo no va a tener el *tiempo promedio de viaje* en el instante **t-5** puesto que no se proporciona dicho valor). Un ejemplo de la estructura se genera a partir del método de la *ventana deslizante* es el siguiente:

t-5	t-4	t-3	t-2	t-1	t
58.05	56.87	77.74	42.64	40.17	41.92
56.87	77.74	42.64	40.17	41.92	39.43
77.74	42.64	40.17	41.92	39.43	48.13
42.64	40.17	41.92	39.43	48.13	62.11
40.17	41.92	39.43	48.13	62.11	46.12
41.92	39.43	48.13	62.11	46.12	49.56
39.43	48.13	62.11	46.12	49.56	54.84
48.13	62.11	46.12	49.56	54.84	58.08
62.11	46.12	49.56	54.84	58.08	46.36
46.12	49.56	54.84	58.08	46.36	48.59
49.56	54.84	58.08	46.36	48.59	66.64
54.84	58.08	46.36	48.59	66.64	64.68
58.08	46.36	48.59	66.64	64.68	85.68
46.36	48.59	66.64	64.68	85.68	58.97
48.59	66.64	64.68	85.68	58.97	81.60
66.64	64.68	85.68	58.97	81.60	80.21
64.68	85.68	58.97	81.60	80.21	63.45
85.68	58.97	81.60	80.21	63.45	78.05
58.97	81.60	80.21	63.45	78.05	69.04
81.60	80.21	63.45	78.05	69.04	69.66

Figura 4.33: Estructura

generada con el método de la

ventana deslizante

Una vez creada la estructura de los datos pertinentes, se proporciona como entrada a los algoritmos de minería de datos.

A la hora de generar las predicciones de los intervalos de tiempo a predecir, se accede a la última fila creada de la estructura construida anteriormente y se *desplaza una posición a la izquierda* , de tal forma que tenemos los valores de *tiempo promedio de viaje* desde el instante de tiempo **t-5** (en este ejemplo) al instante de tiempo **t-1**, dejando el instante de tiempo **t** sin dato puesto que es el primer instante de tiempo a predecir. Tras estimar el valor de la primera ventana de tiempo a predecir, se rellena el espacio vacío en el instante **t** con el objetivo de volver a desplazar los valores de esta fila hacia la izquierda para volver a dejar vacío el instante **t** con el fin de predecir el segundo intervalo de tiempo a estimar. Es decir, se utiliza el valor predicho como instante anterior al siguiente valor a predecir.

Para cada ruta, día e intervalo a estimar, se sigue el proceso comentado previamente. Después de obtener todos los valores a predecir, se obtiene el *error MAPE* correspondiente comparando los datos reales de la base de datos *tfgrealvalues* con los valores obtenidos. Los *errores MAPE* generados por los distintos algoritmos de minería de datos son los siguientes:

XGBoost	**KNN**	**LightGBM**	**MLP**	**SVR**	**Linear Regression**
0.212	0.229	0.22	0.229	0.2544	0.212

Tabla 4.2: Errores MAPE de la tercera aproximación de predicciones del tiempo promedio de viaje

En esta aproximación, los algoritmos que han proporcionado mejores resultados han sido *XGBoost* y *Regresión Lineal*. Con respecto al *error MAPE* obtenido en las *Redes Neuronales*, cabe mencionar que este algoritmo ha experimentado una notable mejoría con respecto a su empleo en la primera aproximación. Esto se debe a que en esta aproximación se le han proporcionado una mayor cantidad de instancias de entrenamiento, por lo que ha podido entrenarse de una forma más eficaz. Por otra parte, el algoritmo *SVR* ha empeorado con respecto a la primera aproximación y los demás se han mantenido prácticamente en la misma línea.

4.5 Predicciones del volumen de tráfico

El segundo objetivo a conseguir propuesto por la competición es predecir el **volumen de tráfico** que se va a producir en una serie de

intervalos de 20 minutos en unos días determinados (según la **Figura 4.2**). Debido a que este parámetro del flujo de tráfico a predecir también evoluciona en el tiempo (al igual que el tiempo promedio de viaje), se ha procedido a aplicar de forma similar las aproximaciones de predicción desarrolladas para la estimación del tiempo promedio de viaje puesto que en este caso también se trata con **series temporales**. Sin embargo, en este caso el objetivo no es estimar valores de volumen de tráfico por ruta, sino predecir tales valores por cada uno de los pares *barrera de peaje-dirección* contemplados en la topología de la red de carreteras dada por la competición. A continuación, se expone detalladamente el proceso de construcción de las diferentes técnicas de minería de datos escogidas para la tarea en cuestión.

4.5.1 Primera aproximación: Algoritmos diversos de Machine Learning

La primera aproximación de predicciones del volumen de tráfico desarrollada se basa en la misma idea propuesta para la estimación del *tiempo promedio de viaje*: construir unas vistas minables sobre las base de datos de los datos modificados y de los datos de testeo.

A la hora de generar la estructura de dichas vistas para albergar los datos a utilizar para llevar a cabo la labor de predicción se tuvieron en cuenta los mismos principios de construcción de la misma que para el *tiempo promedio de viaje*. No obstante, en este caso, se accedieron a otras tablas de las bases de datos distintas a las utilizadas para el *tiempo promedio de viaje* con el fin de acceder a los datos pertinentes a la estimación de la segunda variable de tráfico. Estas vistas se generaron en base a la estructura representada a continuación:

vista_minable (***type_day***, ***twenty_min_previous***, ***forty_min_previous***, ***sixty_min_previous***, ***eighty_min_previous***, ***onehundred_min_previous***, ***onehundredtwenty_min_previous***, ***pressure***, ***sea_pressure***, ***wind_direction***, ***wind_speed***, ***temperature***, ***rel_humidity***, ***precipitation***, ***volume***)

Como podemos apreciar, el esquema se asemeja bastante al de la vista minable creada para la primera aproximación de predicciones del *tiempo promedio de viaje*. La única diferencia es la variable de tráfico a predecir.

Los resultados obtenidos utilizando la presente aproximación de predicciones son los siguientes:

XGBoost	KNN	LightGBM	MLP	SVR	Linear Regression
0.31	0.247	0.32	0.3467	0.3815	0.33

Tabla 4.3: Errores MAPE de la primera aproximación de predicciones del volumen de tráfico

En esta primera aproximación, el mejor algoritmo ha sido el *k-Vecinos Más Cercanos*, con un error MAPE de **0.247**. En general, el desempeño de los distintos algoritmos ha sido más deficiente que en la misma aproximación que para el *tiempo promedio de viaje*. La cuestión primordial de este suceso es la **cantidad tan pequeña de datos de entrenamiento** que les hemos proporcionado a las diferentes técnicas de minería de datos. Para llevar a cabo las estimaciones para cada par *barrera de peaje-dirección* y ventana de tiempo, los algoritmos han recibido alrededor de **30 filas de entrenamento** puesto que solo nos proporcionaron un mes de datos para dicha fase. Por lo tanto, el rendimiento de los mismos ha sido menos eficaz.

De la misma forma, los algoritmos han tenido la dificultad añadida de que los datos de entrenamiento tienen muchas variables explicativas, lo que obstaculiza la tarea de predicción.

4.5.2 Segunda aproximación: ARIMA

Al igual que la primera aproximación de estimaciones del *volumen de tráfico* comentada anteriormente, para desarrollar la segunda aproximación para estimar el *volumen de tráfico* se llevaron a cabo los mismos procedimientos de realización de predicciones abordadas en la segunda aproximación relacionada con el *tiempo promedio de viaje*. Es decir, en esta aproximación, para una *barrera de peaje* y *dirección* de conducción concretos, se procede a realizar las estimaciones en los intervalos de tiempo a predecir. No obstante, es preciso mencionar que existen algunas diferencias entre las dos segundas aproximaciones.

Por un lado, en los datos proporcionados por la competición sobre el *volumen de tráfico* en intervalos de tiempo de 20 minutos hay bastantes menos ventanas de tiempo que faltan en las tablas relacionadas con esta variable de tráfico. De esta forma, no ha sido necesario calcular tantas veces, para cada ventana de tiempo faltante, el *volumen de tráfico* de aquellos intervalos de tiempo que no lo tenían. No obstante, el inconveniente encontrado en dicha información es la cantidad suministrada de la misma. Mientras que para las tareas de predicción del *tiempo promedio de viaje* se proporcionaron datos de

4 meses (desde julio hasta octubre), para la estimación del *volumen de tráfico* se suministraron datos de 2 meses (septiembre y octubre). Por lo tanto, la información de entrenamiento que se ha aportado al entrenamiento de los modelos de aprendizaje automático es bastante menor en este caso y, de este modo, ha conllevado una dificultad añadida a la fase de predicción del *volumen de tráfico*.

Por otra parte, el patrón que se da en los datos proporcionados del *volumen de tráfico* se mantiene bastante más similar entre días que en la información suministrada para la predicción del *tiempo promedio de viaje*. Para apreciar esto, como ejemplo podemos visualizar la **Figura 4.11**.

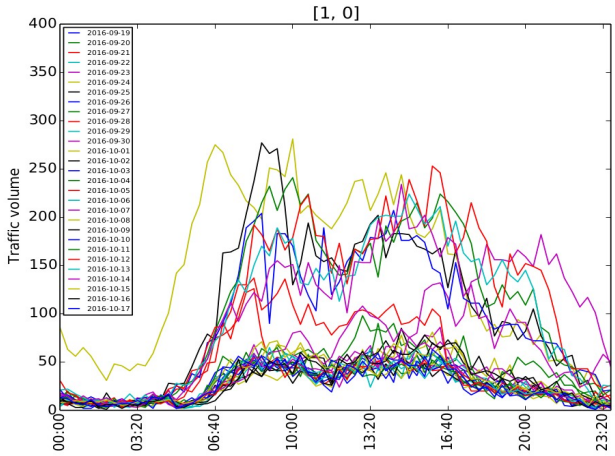


Figura 4.34: Ejemplo de gráfica del volumen de tráfico

en una ruta determinada en los días de entrenamiento

En esta gráfica podemos observar que en una gran parte de los días de entrenamiento se repite el mismo patrón (en la superposición de una gran cantidad de gráficas de línea). Aquellos días que no siguen dicho patrón se considera *ruido*, por lo que es conveniente eliminarlo. Para ello, es necesario identificar aquellos días que no se comportan de manera similar a la mayoría de días de entrenamiento. Con el fin de llevar a cabo esto, se ejecuta lo siguiente:

```
dates_out_1_0 = ['2016-09-21','2016-09-28','2016-09-30','2016-10-01', '2016-10-02','2016-10-03','2016-10-04','2016-10-05','2016-10-06', '2016-10-07']
dates_out_1_1 = ['2016-10-01', '2016-10-02','2016-10-03','2016-10-04','2016-10-05','2016-10-06', '2016-10-07']
dates_out_2_0 = ['2016-09-28', '2016-10-01', '2016-10-02','2016-10-03','2016-10-04','2016-10-05','2016-10-06', '2016-10-07']
if (pair[0] == 1 and pair[1] == 0):
 booleanos = crearBooleanos(dates_out_1_0)
 df1 =  df1[booleanos]
elif ((pair[0] == 1 and pair[1] == 1) or (pair[0] == 3 and pair[1] == 1)):
 booleanos = crearBooleanos(dates_out_1_1)
 df1 =  df1[booleanos]
elif (pair[0] == 2 and pair[1] == 0):
 booleanos = crearBooleanos(dates_out_2_0)
 df1 =  df1[booleanos]
```

Figura 4.35: Eliminación del ruido de los datos proporcionados para

la predicción del volumen de tráfico

Para aquellos pares *barreras de peaje-dirección de conducción* que presentan ruido en sus gráficas, se determinan aquellos días que lo generan y se filtran en la estructuras pertinentes para su eliminación. Tras el proceso de exclusión de los datos ruidosos ésta se queda de la siguiente manera:

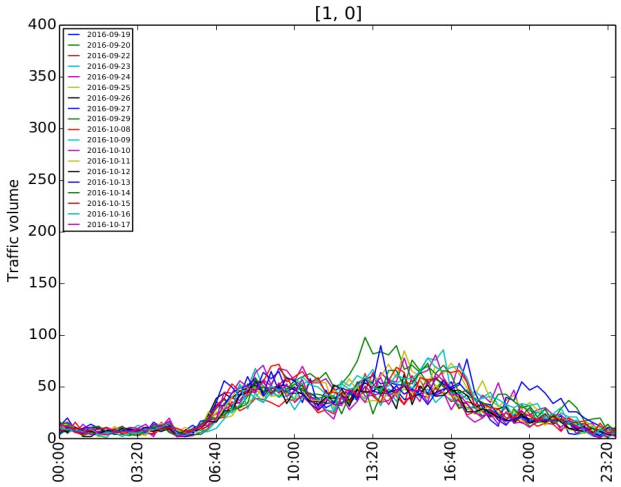


Figura 4.36: Gráfica resultante de la

eliminación de ruido

Como se puede apreciar en esta figura, podemos ver con notoriedad la ausencia de ruido en las series temporales con respecto a la **Figura 4.34**. La eliminación de dichos valores que no siguen la tendencia de la mayoría de los datos en ese par *barrera de peaje-dirección de conducción* propicia que se puedan crear modelos más detallados y precisos para llevar a cabo las estimaciones oportunas del *volumen de tráfico*.

Tras realizar el filtrado, se proporciona la información suministrada con respecto al *volumen de tráfico* a las técnicas de minería de datos siguiendo las mismas pautas que para el *tiempo promedio de viaje* con el objetivo de que ejecuten las fases de entrenamiento y testeo para predecir esta variable de tráfico. A partir de esto, se consiguió un *error MAPE* de **0.152**. Este *error MAPE* supera al error obtenido con el mismo algoritmo en la predicción del *tiempo promedio de viaje*. Por una parte, esto se debe a que las series temporales relacionadas con el *volumen de tráfico* presentan más **estacionalidad** durante los días que aquellas que tienen que ver con el *tiempo promedio de viaje*, por lo que al algoritmo le cuesta menos sacar patrones de los mismos. Por otra parte, en los datos relacionados con el *volumen de tráfico* había menos datos faltantes, por lo que se ha podido trabajar con más datos reales que a la hora de predecir el *tiempo promedio de viaje*.

4.5.3 Tercera aproximación

Los pasos llevados a cabo para implementar la tercera aproximación con el fin de pronosticar la variable de tráfico *volumen de tráfico* para los pares *barrera de peaje-dirección de conducción* se equiparan a los seguidos para las estimaciones del *tiempo promedio de viaje*: utilizar el método de la ***ventana deslizante***.

Previamente a la aplicación de dicho método sobre los datos de entrenamiento suministrados para la predicción del *volumen de tráfico* se ejecuta el filtrado de datos mencionado en la **Figura 4.35** para suprimir el ruido presente en las series temporales. A continuación de esto, se genera la estructura apropiada mediante la ejecución del método indicado en la **Figura 4.32**, originando de este modo la distribución de los datos visualizados en la **Figura 4.33**. Seguidamente, se provee a las técnicas de minería de datos de la información indispensable para que puedan desempeñar su función de la mejor forma posible.

Tras la ejecución de los algoritmos de aprendizaje automático pertinentes, se obtuvieron los *errores MAPE* que se exponen a continuación:

XGBoost	**KNN**	**LightGBM**	**MLP**	**SVR**	**Linear Regression**
0.24	0.252	0.272	0.25	0.269	0.276

Tabla 4.4: Errores MAPE de la tercera aproximación de predicciones del volumen de tráfico

En comparación con los errores obtenidos en la primera aproximación de predicciones del *tiempo promedio de viaje*, es preciso mencionar que, en líneas generales, el rendimiento de los algoritmos en este caso ha sido peor. La razón principal de esto es que las técnicas de minería de datos en este caso han recibido una menor cantidad de datos de entrenamiento (de un mes en lugar de tres como con el *tiempo promedio de viaje*) y no han podido entrenarse lo suficiente.

Nota: Los valores de las predicciones obtenidas pueden ser accedidos a través de [27].

Capítulo 5
Conclusiones y líneas futuras

5.1.1 Conclusiones

En este TFG se ha desarrollado un proyecto de minería de datos para la predicción del tiempo promedio de viaje y volumen de tráfico de una red de carreteras. Para llevarlo a cabo, se han utilizado distintas técnicas de minería de datos: *Regresión Lineal*, *SVM*, *k-vecinos más cercanos*, *Redes Neuronales*, *XGBoost, LightGBM*, *modelo ARIMA* y *ventana deslizante*. Como herramienta para las labores de construcción de estos modelos de predicción se ha empleado el lenguaje de programación *Python* y para almacenar los datos necesarios para la ejecución de las técnicas se ha usado el software *PostgreSQL*.

Por otra parte, el desempeño de nuestro trabajo ha sido bastante aceptable en comparación con los mejores *errores MAPE* de la competición tanto para la predicción del *tiempo promedio de viaje* como para el *volumen de tráfico*. En el caso de la primera variable de tráfico, el mejor resultado que obtuvimos fue con el **modelo ARIMA** (con un error MAPE de **0.163**), seguido del **algoritmo SVR** en la primera aproximación (con un error MAPE de **0.2065**) y de los **algoritmos XGBoost y Regresión Lineal** (con un error MAPE de **0.212**). Con respecto a esta variable, el mejor resultado de la competición fue un error MAPE de **0.1748** [28][29]. En relación a este resultado, hay que tener en cuenta que, a pesar de haber obtenido un mejor resultado que el primero de la competición con el modelo ARIMA, es necesario precisar que, como comentamos anteriormente, nos faltan datos reales de los intervalos a predecir, por lo que a la hora de calcular el error hay intervalos de tiempo que no tuvimos en cuenta. Por lo tanto, el error que hubiéramos obtenido con todos los datos reales hubiera sido mayor que el obtenido en este caso.

En el caso del *volumen de tráfico*, al igual que con el *tiempo promedio de viaje,* el algoritmo con mejor desempeño fue con el **modelo ARIMA** (con un error MAPE de **0.152**), seguido del **XGBoost** en la primera aproximación

(con un error de **0.24**) y del **KNN** en la segunda aproximación (con un error de **0.247**). En cuanto a los resultados obtenidos por los participantes de la competición con respecto al *volumen de tráfico*, el mejor *error MAPE* que se obtuvo fue **0.1203**, el segundo **0.1298** y el tercero **0.1326** [30][31]. En este caso, el error que obtuvimos no se encuentra entre los tres primeros clasificados pero, a pesar de esto, es bastante razonable dada la complejidad del problema.

5.1.2 Líneas futuras

El planteamiento futuro que se pretende llevar a cabo sobre este proyecto consiste en aplicar las técnicas de minería de datos empleadas en la predicción de flujo de tráfico a la *red de carreteras de la isla de Tenerife*. Con la colaboración del Cabildo de Tenerife se persigue obtener la mayor cantidad de información de tráfico posible sobre la circulación de vehículos en Tenerife y realizar tareas de estimación de parámetros de tráfico con el objetivo de controlar aquellos aspectos del mismo que son críticos y de especial relevancia en la isla.

Para llevar a cabo este proyecto futuro se pretende aplicar las mismas técnicas de minería de datos utilizadas en el TFG sobre los datos de tráfico de la isla, así como realizar un estudio más exhaustivo de estos algoritmos con el fin de refinarlos y optimizarlos e incluso aplicar nuevas técnicas. También se procederá a analizar otras formas de preprocesar los datos para que aumente el rendimiento de las distintas técnicas empleadas.

Capítulo 6
Summary and Conclusions

6.1.1 Conclusions

In this *End-of-Degree Project* a data mining project has been developed to predict the average travel time and traffic volume of a road network. To do this, different data mining techniques have been used: *Linear Regression*, *SVM*, *k-Nearest Neighbors*, *Neural Networks*, *XGBoost*, *LightGBM*, *ARIMA model* and *sliding window*. The *Python* programming language has been used as a tool for the construction of these prediction models and the *PostgreSQL* database management system to store the data required for the execution of the techniques.

On the other hand, the performance of our work has been quite acceptable in comparison to the best *MAPE errors* in the competition for both the prediction of *average travel time* and *traffic volume*. In the case of the first traffic variable, the best result we obtained was with the **ARIMA model** (with a MAPE error of **0.163**), followed by the **SVR algorithm** in the first approximation (with a MAPE error of **0.2065**) and the **XGBoost** and **Linear Regression** algorithms (with a MAPE error of **0.212**). For this variable, the best result of the competition was a MAPE error of **0.1748** [28] [29]]. In relation to this result, it is necessary to bear in mind that, despite having obtained a better result with the ARIMA model than the first classified in the competition, it is necessary to specify that, as we commented above, we lack real data at the time intervals to be predicted, so when calculating the error there are time intervals that we did not take into account. Therefore, the error we would have obtained with all the real data would have been greater than the error we obtained in this case.

In the case of traffic volume, as with the *average travel time*, the best-performing algorithm was the **ARIMA model** (with a MAPE error of **0.152**), followed by **XGBoost** in the first approach (with an error of **0.24**) and **KNN** in the second approach (with an error of **0.247**). As for the results obtained by the participants of the competition with respect to *traffic volume*,

the best *MAPE error* obtained was **0.1203**, the second **0.1298** and the third **0.1326**[30][31]. In this case, the error we got was not among the top three but, despite this, it is quite reasonable given the complexity of the problem.

6.1.2 Future lines

The future approach to this project is to apply the data mining techniques used to predict traffic flow in the *road network of the island of Tenerife*. With the collaboration of the town council of Tenerife, the aim is to obtain as much traffic information as possible about the circulation of vehicles in Tenerife and to carry out traffic parameter estimation tasks with the objective of controlling those aspects of it that are critical and of particular relevance to the island.

The aim of this future project is to apply the same data mining techniques used in the End of Degree Project on the island's traffic data, as well as to carry out a more thorough study of these algorithms in order to refine and optimise them and even apply new techniques. Other ways of pre-processing the data to improve the performance of the various techniques used will also be explored.

Capítulo 7
Presupuesto

En este apartado se presenta una tabla con el presupuesto de todos los elementos que intervinieron en el desarrollo este proyecto:

Elemento	**Cantidad**	**Coste unitario [€/u]**	**Coste**
Portátil *Lenovo IdeaPad Z510*	1	900 €	900 €
Ratón *Logitech m170*	1	11€	11€
Monitor *Asus VW193D*	1	60€	60€
Total			971€

Tabla 7.1: Presupuesto del material utilizado en el proyecto

Por otra parte, se expone el coste total del número de horas requeridas en la ejecución de este proyecto:

Personal	**Horas de trabajo**	**Coste/Hora**	**Total**
1	300	12€	3600€

A partir de estas tablas se calcula el presupuesto total utilizado:

Suma presupuestos (Material y Personal)	**Total**
971€ + 10500€	11471€

Bibliografía

[1] Y. Yin and P.Shang, "Forecasting traffic time series with multivariate predicting method", *Applied Mathematics and Computation*, vol. 291 , pp. 266-278, 2016. [Online]. Disponible en: https://goo.gl/5gkEfP

[2] P. Yuan and X. Lin , "How long will the traffic flow time series keep efficacious to forecast the future?", *Physica A: Statistical Mechanics and its Applications*, vol. 467 , pp. 419-437, 2017. [Online]. Disponible en: https://drive.google.com/open?id=1r4ZF2wughH4nZcRUsQqqn_hmrSSR6mw8

[3] H. A. Sevilla, "Predicción de tráfico en las carreteras de la red de la Generalitat Valenciana", Trabajo fin de carrera, Dep. De Sistemas Informáticos y Computación, Escola Tècnica Superior d'Enginyeria Informàtica, Universitat Politècnica de València, Valencia, 2015. [Online]. Disponible en: https://drive.google.com/open?id=1usrXMdc7c-J3-1Rt2CjzAHLvZII3eeEO

[4] M. Goletz, I. Feige and D. Heinrichs, "What Drives Mobility Trends: Results from Case Studies in Paris, Santiago de Chile, Singapore and Vienna", *Transportation Research Procedia*, vol. 13 , pp. 49-60, 2016. [Online]. Disponible en: https://drive.google.com/open?id=1590qIrBgZvJjANFsTvHYZlLkWRz9vfFN

[5] C. Gloves, R. North, R. Johnston and G. Fletcher, "Short Term Traffic Prediction on the UK Motorway Network Using Neural Networks", *Transportation Research Procedia*, vol. 13 , pp. 184-195, 2016. [Online]. Disponible en: https://drive.google.com/open?id=1rDz5GfONXSf7ZFhLcgQv_FFfSnrmj55F

[6]J.H. Orallo, Mª J.R. Quintana and C.F. Ramírez, Introducción a la Minería de Datos. 1ªed.: Pearson, 2004

[7] http://rtdibermatica.com/?tag=kdd

[8]A. Moujahid, I. Inza and P. Larrañaga http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t9knn.pdf

[9] https://tex.stackexchange.com/questions/132444/diagram-of-an-artificial-

neural-network

[10] https://advancedtech.wordpress.com/2008/08/07/redes-neuronales-rna/

[11] https://www.researchgate.net/figure/Figura-4-Arbol-de-regresion-para-la-densidad-poblacional-en-Alsophila-fi-rma-Las_fig4_263114868

[12] http://www.mdpi.com/2072-4292/9/12/1299/htm

[13] http://www.escuela-verano.otrasenda.org/wp-content/uploads/2015/06/curso_series.pdf

[14] http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/EDescrip/tema7.pdf

[15] https://www.ibm.com/support/knowledgecenter/es/SS3RA7_sub/modeler_mainhelp_client_ddita/components/dt/timeseries_trend.html

[16] https://www.ibm.com/support/knowledgecenter/es/SS3RA7_sub/modeler_mainhelp_client_ddita/components/dt/timeseries_seasonal.html

[17] https://www.ibm.com/support/knowledgecenter/es/SS3RA7_sub/modeler_mainhelp_client_ddita/components/dt/timeseries_pulses.html

[18] http://www5.uva.es/estadmed/datos/series/series2.htm

[19] http://finanzaszone.com/analisis-y-prediccion-de-series-temporales-con-r-ii-estacionariedad-y-raices-unitarias/

[20] http://www.estadistica.net/ECONOMETRIA/SERIES-TEMPORALES/modelo-arima.pdf

[21] https://es.wikipedia.org/wiki/Regresi%C3%B3n_lineal

[22] https://medium.com/data-science-group-iitr/support-vector-machines-svm-unraveled-e0e7e3ccd49b

[23] http://www.saedsayad.com/support_vector_machine_reg.htm

[24] https://tianchi.aliyun.com/competition/information.htm?raceId=231597

[25] https://github.com/lianjiecao/kdd-cup 2017/blob/master/dataSets/Route_KDD.pdf

[26] https://drive.google.com/open?

id=1zt2gNDVpM9rWXRlOQuMq5oWKT5KU_cnG

[27] https://drive.google.com/drive/u/1/folders/1rjJAqEoT06edKbC04flo8WKTo62lvGbn

[28] https://tianchi.aliyun.com/competition/rankingList.htm?spm=5176.11165320.0.0.6c5441050t1CtX&season=1&raceId=231597&pageIndex=1

[29] K.Hu, P.Huang, H.Chen and P. Yan, "KDD CUP 2017 Travel Time Prediction Predicting Travel Time - The Winning Solution of KDD Cup 2017", presented at the KDD 2017, Halifax, Nova Scotia - Canada, 2017. [Online]. Disponible en: https://drive.google.com/open?id=12a4pBbxA6h_zy517ARtHTtn61w7uZ7jJ

[30] https://tianchi.aliyun.com/competition/rankingList.htm?spm=5176.11165320.0.0.39454105uKtxEh&raceId=231597&season=0

[31] K.Hu, P.Huang, H.Chen and P. Yan, "KDDCUP 2017 Volume Prediction Step by step modeling for travel volume prediction", presented at the KDD 2017, Halifax, Nova Scotia-Canada, 2017. [Online]. Disponible en: https://drive.google.com/open?id=1uK8IwRTe061NVbWQn4FQFxH7BUx6b6mq