



Trabajo de Fin de Grado

Grado en Ingeniería Informática

Predicción de flujos de tráfico mediante técnicas de Machine Learning

*Prediction of traffic flows using Machine Learning
techniques.*

Javier Ramos Fernández

La Laguna, 24 de abril 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:

“Predicción de flujos de tráfico mediante técnicas de Machine Learning”

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 24 de abril 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 que, a pesar del alejamiento, me hagan pasar unos veranos inolvidables y 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

* Si quiere permitir que se compartan las adaptaciones de tu obra mientras se comparta de la misma manera y quieres permitir usos comerciales de tu obra (licencia de Cultura Libre) indica:



© Esta obra está bajo una licencia de Creative Commons Reconocimiento-CompartirIgual 4.0 Internacional.

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. Problemática y estado del arte.....	2
1.2 Objetivos.....	6
1.3 Organización de la memoria.....	7
Capítulo 2 Tecnologías.....	8
2.1 Almacenamiento de datos.....	8
2.1.1 En el proyecto.....	8
2.1.1.1 PostgreSQL.....	8
2.1.2 Otras tecnologías posibles.....	9
2.1.2.1 MySQL.....	10
2.1.2.2 Oracle.....	11
2.1.2.3 Microsoft SQL Server.....	11
2.2 Ciencia de datos.....	13
2.2.1 En el proyecto.....	13
2.2.1.1 Python.....	13
2.2.2 Otras tecnologías posibles.....	13
2.2.2.1 RapidMiner.....	14
2.2.2.2 Lenguaje R.....	14
2.2.2.3 Weka.....	15
Capítulo 3 La minería de datos.....	17

3.1	Introducción.....	17
3.2	Técnicas de minería de datos.....	20
3.2.1	XGBOOST.....	20
3.2.1.1	Definición.....	20
3.2.1.2	Gradient boosting.....	21
3.2.1.3	Implementación del algoritmo.....	24
3.2.2	LightGBM.....	25
3.2.2.1	Definición.....	25
3.2.2.2	Ventajas.....	26
3.2.3	Perceptrón multicapa (Redes neuronales).....	27
3.2.3.1	Definición.....	27
3.2.3.2	Neuronas.....	27
3.2.3.3	Redes de neuronas.....	29
3.2.3.4	Entrenamiento de una red neuronal.....	30
3.2.4	Modelo ARIMA.....	32
3.2.4.1	Definición de una serie temporal.....	32
3.2.4.2	Características de las series temporales.....	32
3.2.4.3	Componentes de las series temporales.....	37
3.2.4.4	Tipos de esquemas para series temporales.....	38
3.2.4.5	Clasificación descriptiva de las series temporales.....	40
3.2.4.6	Funciones de autocorrelación y autocorrelación parcial.....	43
3.2.4.7	Estimación de las componentes de una serie temporal.....	45
Capítulo 4	Fases del proyecto.....	47
4.1	Problema planteado por la competición KDDCup 2017.....	48
4.1.1	Contexto.....	48
4.1.2	Tareas.....	49
4.1.3	Etapas y reglas de la competición.....	50
4.1.4	Métricas de evaluación.....	51

4.2 Creación de bases de datos para almacenar los datos de la competición y modificaciones.....	53
4.2.1 Estructura de bases de datos propuesta.....	53
4.2.2 Base de datos con los datos originales.....	54
4.2.2.1 Tabla vehicle_routes (“routes_table4.csv”).....	54
4.2.2.2 Tabla road_links (“links_table3.csv”).....	54
4.2.2.3 Tabla vehicle_trajectories_training (“trajectories_table 5_training.csv”).....	56
4.2.2.4 Tabla traffic_volume_tollgates_training (“volume_table6_training.csv”).....	58
4.2.2.5 Tabla weather_data (“weather (table 7)_training.csv”).....	59
4.2.2.6 Tabla travel_time_intersection_to_tollgate (“trajectories_table5_training_20min_avg_travel_time.csv”).....	61
4.2.2.7 Tabla traffic_volume_tollgates (“volume_table 6_training_20min_avg_volume.csv”).....	61
4.2.3 Base de datos con los datos modificados.....	63
4.2.3.1 Tabla road_links_modified (“links_table3.csv”).....	63
4.2.3.2 Tabla vehicle_routes_modified (“routes_table4.csv”).....	64
4.2.3.3 Tabla vehicle_trajectories_training_modified (“trajectories_table 5_training.csv”).....	64
4.2.3.4 Tabla traffic_volume_tollgates_training_modified (“volume_table6_training.csv”).....	66
4.2.3.5 Tabla weather_data_modified (“weather (table 7)_training.csv”).....	67
4.2.3.6 Tabla travel_time_intersection_to_tollgate_modified (“trajectories_table5_training_20min_avg_travel_time.csv”).....	67
4.2.3.7 Tabla traffic_volume_tollgates_modified (“volume_table 6_training_20min_avg_volume.csv”).....	68
4.2.4 Base de datos con los datos relacionados con la fase de pruebas....	70
4.2.4.1 Tabla travel_time_intersection_to_tollgate_test1	

("test1_20min_avg_travel_time.csv").....	70
4.2.4.2 Tabla traffic_volume_tollgate_test1 ("test1_20min_avg_volume.csv").....	71
4.2.4.3 Tabla tabla_resultado_average_travel_time.....	72
4.2.4.4 Tabla tabla_resultado_traffic_volume.....	73
4.3 Creación de gráficas para visualizar los datos almacenados.....	74
4.3.1 Gráficas del tiempo promedio de viaje de todos los días por rutas.....	75
4.3.2 Gráfica del tiempo promedio de viaje de algunos días en todas las horas en cada una de las rutas.....	77
4.4 Predicciones del tiempo promedio de viaje.....	81
4.4.1 Primera aproximación.....	81
4.4.1.1 Creación de las vistas minables.....	84
4.4.1.2 Realización de predicciones a partir de las vistas.....	89
4.4.2 Segunda aproximación.....	92
4.4.2.1 Preparación de los datos.....	92
4.5 Predicciones del volumen de tráfico.....	94
Capítulo 5 Conclusiones y líneas futuras.....	95
Capítulo 6 Summary and Conclusions.....	96
Capítulo 7 Presupuesto.....	97
7.1 Sección Uno.....	97
Capítulo 8 Título del Apéndice 1.....	98
8.1 Algoritmo XXX.....	98
8.2 Algoritmo YYY.....	98
Capítulo 9 Título del Apéndice 2.....	100
9.1 Otro apéndice: Sección 1.....	100
9.2 Otro apéndice: Sección 2.....	100

Índice de figuras

Figura 2.1: Consola interactiva de PostgreSQL.....	9
Figura 2.2: Entorno gráfico de MySQL Workbench.....	10
Figura 2.3: Entorno gráfico del entorno de desarrollo Oracle JDeveloper.....	11
Figura 2.4: Entorno gráfico de Microsoft SQL Server Management Studio.....	12
Figura 2.5: Entorno gráfico de RapidMiner.....	14
Figura 2.6: Entorno gráfico de Weka.....	16
Figura 3.1: El proceso de descubrimiento de conocimiento en bases de datos (KDD).....	17
Figura 3.2: Cuatro de las tareas principales de minería de datos.....	19
Figura 3.3: Crecimiento level-wise del árbol en XGBoost.....	26
Figura 3.4: Crecimiento leaf-wise del árbol en LightGBM.....	26
Figura 3.5: Estructura de una neurona artificial.....	28
Figura 3.6: Estructura de una red neuronal.....	29
Figura 3.7: Ejemplo de serie temporal.....	32
Figura 3.8: Ejemplo de una serie temporal con tendencia.....	33
Figura 3.9: Ejemplo de una serie temporal con ciclos estacionales.....	34
Figura 3.10: Serie temporal con pulsos.....	34
Figura 3.11: Distintos tipos de valores atípicos.....	36
Figura 3.12: Comparación de una serie estacionaria con una no estacionaria con respecto a la autocovarianza.....	39
Figura 3.13: Serie temporal con tendencia agregada a la estacionalidad.....	40

Figura 3.14: Serie temporal con tendencia multiplicada por la estacionalidad.....	40
Figura 3.15: Ejemplo de serie estacionaria.....	41
Figura 3.16: Ejemplo de serie no estacionaria.....	41
Figura 3.17: Comparación de una serie estacionaria con una no estacionaria con respecto a la media.....	42
Figura 3.18: Comparación de una serie estacionaria con una no estacionaria con respecto a la varianza.....	42
Figura 3.19: Comparación de una serie estacionaria con una no estacionaria con respecto a la autocovarianza.....	43
Figura 3.20: Ejemplo de un gráfico de autocorrelación.....	44
Figura 3.21: Serie temporal y su tendencia.....	46
Figura 3.22: Componente regular de la serie temporal tras eliminar su tendencia.....	46
Figura 4.1: Cálculo del error de predicción del tiempo promedio de viaje. .	50
Figura 4.2: Topología de la red de carreteras de la zona objetivo.....	50
Figura 4.3: Ventanas de tiempo para la predicción del tráfico.....	51
Figura 4.4: Cálculo del error de predicción del volumen de tráfico.....	52
Figura 4.5: Rutas de la competición KDDCup2017 con los enlaces que las forman.....	56
Figura 4.6: Tiempo promedio de viaje medio en cada uno de los días en las diferentes rutas.....	75
Figura 4.7: Continuación de la gráfica de la Figura 4.4.....	76
Figura 4.8: Tiempo promedio de viaje por horas en algunos días en la ruta A-2.....	77
Figura 4.9 Tiempo promedio de viaje por horas en algunos días en la ruta B-1.....	78
Figura 4.10: Tiempo promedio de viaje por horas en algunos días en la ruta A-3.....	78
Figura 4.11: Tiempo promedio de viaje por horas en algunos días en la ruta	

B-3.....	79
Figura 4.12: Tiempo promedio de viaje por horas en algunos días en la ruta C-1.....	79
Figura 4.13: Tiempo promedio de viaje por horas en algunos días en la ruta C-3.....	80
Figura 4.14: Bloque principal de código que crea la evolución de las 2 horas previas a la ventana de tiempo correspondiente en los datos de entrenamiento.....	84
Figura 4.15: Obtención de las rutas de tráfico junto con las ventanas de tiempo comprendidas en los intervalos de tiempo a predecir.....	85
Figura 4.16: Estructura iterativa para crear las columnas relacionadas con la evolución del tráfico en las dos horas previas a la ventana de tiempo considerada.....	86
Figura 4.17: Consulta SQL que rellena la columna del tiempo promedio de viaje 20 minutos antes de la ventana de tiempo considerada.....	86
Figura 4.18: 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.....	87
Figura 4.19: Combinación de la tabla con los datos de entrenamiento del tiempo promedio de viaje junto con los datos meteorológicos.....	88
Figura 4.20: Código SQL que crea una vista para cada ruta e intervalo a partir de la vista con los datos combinados.....	88
Figura 4.21: Obtención de los datos de entrenamiento para la ruta e intervalo en consideración.....	90
Figura 4.22: Obtención de los datos de prueba para la ruta e intervalo en consideración.....	90
Figura 4.23: Entrenamiento y predicción del tiempo promedio de viaje en los distintos días de predicción con el modelo XGBoost.....	90
Figura 4.24: Cálculo del error medio de los días de predicción para una ruta e intervalo determinado.....	91

Figura 4.25: Acumulación de los errores correspondiente a los días a predecir de cada uno de los intervalos a estimar.....	91
Figura 4.26: Cálculo del segundo sumatorio de la fórmula del error MAPE para un algoritmo.....	91
Figura 4.27: Acumulación de los errores del segundo sumatorio de la fórmula del error MAPE.....	91
Figura 4.28: Cálculo del primer sumatorio de la fórmula del error MAPE para un algoritmo.....	92
Figura 4.29: Obtención de los valores reales del tiempo promedio de viaje para las rutas e intervalos de tiempo a predecir.....	93
Figura 4.30: Cálculo del valor del tiempo promedio de viaje de una parte de aquellas rutas e intervalos de las que no disponemos datos.....	93

Índice de tablas

Tabla 4.1: Tabla vehicle_routes.....	54
Tabla 4.2: Tabla road_links.....	55
Tabla 4.3: Tabla vehicle_trajectories_training.....	57
Tabla 4.4: Tabla traffic_volume_tollgates_training.....	58
Tabla 4.5: Tabla weather_data.....	60
Tabla 4.6: Tabla travel_time_intersection_to_tollgate.....	61
Tabla 4.7: Tabla traffic_volume_tollgates.....	62
Tabla 4.8: Tabla road_links_modified.....	63
Tabla 4.9: Tabla vehicle_routes_modified.....	64
Tabla 4.10: Tabla vehicle_trajectories_training_modified.....	65
Tabla 4.11: Tabla traffic_volume_tollgates_training_modified.....	66
Tabla 4.12: Tabla travel_time_intersection_to_tollgate_modified.....	68
Tabla 4.13: Tabla traffic_volume_tollgates_modified.....	69
Tabla 4.14: Tabla travel_time_intersection_to_tollgate_test1.....	70
Tabla 4.15: Tabla traffic_volume_tollgates_test1.....	71
Tabla 4.16: Tabla tabla_resultado_average_travel_time.....	72
Tabla 4.17: Tabla tabla_resultado_traffic_volume.....	73
Tabla 4.18: Vista creada sobre los datos a utilizar para entrenamiento y testeo de la primera aproximación de predicciones.....	83

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.

Este aumento exponencial del volumen y variedad de información que se ha presentado en las últimas décadas gracias a la era de la información ha propiciado que se generen 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.

Hoy en día, la minería de datos se considera un campo multidisciplinar que se ha desarrollado en paralelo o como prolongación de otras tecnologías, por lo que la investigación y los avances en la minería de datos se nutren de los que se producen en una serie de áreas relacionadas como las *bases de datos*, la *recuperación de información*, la *visualización de*

datos, la *estadística*, el *aprendizaje automático*, etcétera. De esta manera, este campo de conocimiento es ampliamente utilizado en diversas áreas, que cada vez son más a medida que la tecnología sigue avanzando en este campo y que lo han integrado en su actividad para obtener información de gran valor. Algunas de ellas son el *análisis de datos financieros*, la *industria minorista*, la *industria de las telecomunicaciones*, el *análisis de datos biológicos*, *tráfico*, *educación*, etcétera.

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 pausado, 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 sumariaización que permite transformar los datos en otros datos agregados o cruzados de manera sofisticada.

En el caso del trabajo que nos ocupa, la minería de datos es un concepto crucial a emplear en los datos de los que parte el mismo, proporcionados por la competición KDDCup 2017. En este proyecto se pretende utilizar las técnicas y los algoritmos que nos proporciona este área de conocimiento para aplicarlos sobre los datos de tráfico de la competición y obtener previsiones acerca de la tendencia del mismo en una serie de intervalos de tiempo que se nos propone predecir. Así, con la ayuda de dichas estimaciones se puede realizar un control el tráfico a través de la ejecución de decisiones proactivas sobre la red de carreteras (anticiparse al inicio de la congestión del tráfico).

1.1 Antecedentes. Problemática y estado del arte

La realización de predicciones de flujo de tráfico tiene un papel relevante en la sociedad actual debido al gran impacto que tiene el tráfico en

la vida diaria de la gente. En muchas ocasiones nos encontramos con un problema recurrente en las carreteras: la **congestión vehicular**. Actualmente se están realizando una gran cantidad de estudios sobre este tema dada la gran complejidad que presenta puesto que intervienen un considerable número de factores. Las investigaciones que se llevan a cabo en este aspecto son fundamentales para *diseñar las topologías de las carreteras* (intersecciones viales, planes semafóricos, demarcaciones de las vías, señalizaciones verticales, etcétera) de forma óptima y desarrollar *estrategias de planificación* que permitan gestionar de una forma eficaz y eficiente el tráfico; es decir, llevar un control dinámico del mismo que permita realizar y actualizar continuamente *predicciones de los factores* que intervienen en su desarrollo. Esto propiciará un efecto positivo en la economía, el comportamiento de los viajeros, el uso del terreno y otros aspectos.

Algunos ejemplos de estudio en este ámbito son los siguientes:

1. En el artículo [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 el artículo [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 . Para ello, recopilan los datos de las series temporales de flujo de tráfico con diferentes granularidades y realizan una serie de análisis y métodos como analizar la propiedad de memoria larga de las series temporales del flujo de tráfico mediante el cálculo del exponente Hurst, además de un conjunto de comparaciones.
3. El artículo [3] es 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 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) 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 el artículo [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 a 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. El tema del artículo [5] consiste en realizar 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.

Además de los artículos anteriormente mencionados, la competición KDDCup 2017 tiene publicadas las presentaciones de los ganadores de la misma. A continuación se exponen dichos documentos:

1. El documento [6] presenta los resultados del equipo que acabó en el primer puesto de predicción del tiempo promedio de viaje. Para realizar las predicciones, se basaron en unos cuantos modelos de aprendizaje automático como XGBoost, LightGBM y Multilayer Perceptron y se sirven de técnicas combinadas de aprendizaje basadas en funciones de pérdidas, transformaciones de logaritmos y otros métodos.
2. El documento [7] presenta los resultados del equipo que acabó en el segundo puesto de predicción del tiempo promedio de viaje. Se basa principalmente en utilizar XGBoost para realizar las estimaciones de tiempo promedio de viaje, haciendo hincapié en cómo resolver la falta de datos en los conjuntos de datos proporcionados por la competición.
3. El documento [8] presenta los resultados del equipo que acabó en el tercer puesto de predicción del tiempo promedio de viaje. En este trabajo se sigue un adecuado orden de ejecución de fases para realizar las predicciones, desde el preprocesado de datos hasta la realización de métodos combinados de aprendizaje, pasando por la extracción de características y el análisis y elección de modelos de predicción. Aparte de utilizar XGBoost para realizar las predicciones, utilizan el modelo ARIMA, que es muy utilizado para la predicción de series temporales.
4. El documento [9] presenta los resultados del equipo que acabó en el primer puesto de predicción del volumen de tráfico. Este equipo es el mismo que quedó en el primer puesto en la tarea de predicción del tiempo promedio de viaje. Los modelos que utilizan son prácticamente los mismo que utilizaron para estimar el tiempo promedio de viaje, y prestaron gran atención a las características estadísticas de los datos.
5. El documento [10] presenta los resultados del equipo que acabó en el segundo puesto de predicción del volumen de tráfico. En este trabajo se realizaron pruebas sobre muchos modelos de predicción, entre los que se incluyen algunos mencionados en los trabajos anteriores y el modelo

KNN (k-Nearest Neighbours).

6. El documento [11] presenta los resultados del equipo que acabó en el tercer puesto de predicción del volumen de tráfico. El aspecto más relevante de este trabajo es que el autor realiza las predicciones con un sólo modelo, que es la regresión lineal. Para que este modelo pudiera dar buenos resultados, se hizo hincapié en la eliminación de ruido sobre los datos.

Para poder desarrollar este trabajo de predicción de flujos de tráfico, se ha considerado utilizar los datos proporcionados por la competición denominada **KDDCup**, concretamente la edición del año 2017. Esta competición es una competición anual de *Minería de Datos y Descubrimiento de Conocimiento* organizada por **SIGKDD (Association for Computing Machinery's Special Interest Group on Knowledge Discovery and Data Mining)**, la organización profesional líder de minería de datos. En dicha competición se plantea una problemática, se proporcionan los datos necesarios para solventarla y se premian las mejores soluciones propuestas.

El tema elegido por la organización de la KDD Cup 2017 para este año estaba directamente relacionado con la predicción de los flujos de tráfico de las autopistas de peaje en China (concretamente en **Hangzhou**, provincia de **Zhejiang**). El objetivo final era ofrecer a los responsables de la gestión del tráfico medidas preventivas basadas en datos y preparar el camino hacia una solución holística y realista a los cuellos de botella del tráfico.

1.2 Objetivos

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

- *Realización de predicciones del tiempo promedio de viaje.* Uno de los dos objetivos propuestos por la competición es predecir el tiempo promedio de viaje en las rutas y los intervalos de tiempo requeridos por la competición.
- *Realización de varias aproximaciones de predicciones del volumen de tráfico.* El otro objetivo establecido a cumplir es llevar a cabo estimaciones del volumen de tráfico en las distintas barreras de peaje en las direcciones de entrada y salida en los intervalos de

tiempo solicitados por la competición.

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 de la competición 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 que existen en nuestros datos.
- En el capítulo 4 se abordan de forma detallada las distintas fases del proyecto, desde la creación de las bases de datos hasta la comparación de resultados de los distintos algoritmos de aprendizaje automático.

Capítulo 2

Tecnologías

La elección de las herramientas y tecnologías empleadas para el desarrollo del proyecto se ha realizado cautelosamente, de tal forma que nos ha permitido concentrarnos en el problema en cuestión en lugar de tener que instruirnos exhaustivamente en ellas y que aumentara la complejidad del trabajo. 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

2.1.1 En el proyecto

2.1.1.1 PostgreSQL

PostgreSQL es un potente sistema de base de datos objeto-relacional de código abierto basado en el paquete POSTGRES, desarrollado en el Departamento de Informática de la Universidad de California en Berkeley y liberado bajo la licencia BSD. 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. Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una sola compañía, sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo; dicha comunidad es denominada el *PGDG* (*PostgreSQL Global Development Group*).

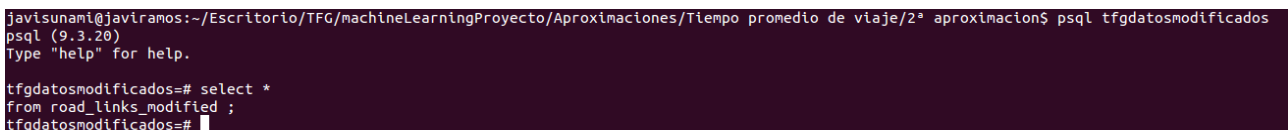
Este sistema de base de datos soporta una gran parte del estándar SQL y ofrece muchas características modernas incluyendo las siguientes:

- Consultas SQL complejas
- Sub-selecciones SQL

- Claves externas
- Disparadores
- Vistas actualizables
- Transacciones
- Control de concurrencia multi-versión (MVCC)
- Integridad transaccional
- Replicación de Streaming (a partir de 9.0)
- Espera en caliente (a partir de 9.0)
- Etc.

Además, PostgreSQL puede ser ampliado por el usuario de muchas maneras, por ejemplo, añadiendo nuevo

- tipos de datos
- funciones
- operadores
- funciones agregadas
- métodos de indexación
- lenguajes procedimentales



```
javisunam@javiramos:~/Escritorio/TFG/machineLearningProyecto/Aproximaciones/Tiempo promedio de viaje/2* aproximacion$ psql tfgdatosmodificados
psql (9.3.20)
Type "help" for help.

tfgdatosmodificados=# select *
from road_links_modified ;
tfgdatosmodificados=#
```

Figura 2.1: Consola interactiva de PostgreSQL

2.1.2 Otras tecnologías posibles

Existen multitud de herramientas, aparte de la mencionada anteriormente, para almacenar los datos sistemáticamente para su posterior uso. A continuación se mencionan algunas destacadas.

2.1.2.1 MySQL

MySQL 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 para cubrir las necesidades de cualquier tipo de organización, desde pequeños establecimientos comerciales a grandes empresas y organismos administrativos. 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. Puede desarrollar sus propias aplicaciones de base de datos en la mayor parte de los lenguajes de programación utilizados en la actualidad y ejecutarlos en casi todos los sistemas operativos. Además, utiliza el *lenguaje de consulta estructurado* (SQL). Se trata del lenguaje utilizado por todas las bases de datos relacionales.

Comparado con *PostgreSQL*, este sistema de administración de bases de datos se ha enfocado tradicionalmente en aplicaciones web de lectura mayormente, usualmente escritas en PHP, 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. Tiene un planificador de consultas extremadamente sofisticado, que es capaz de unir cantidades relativamente grandes de tablas eficientemente.

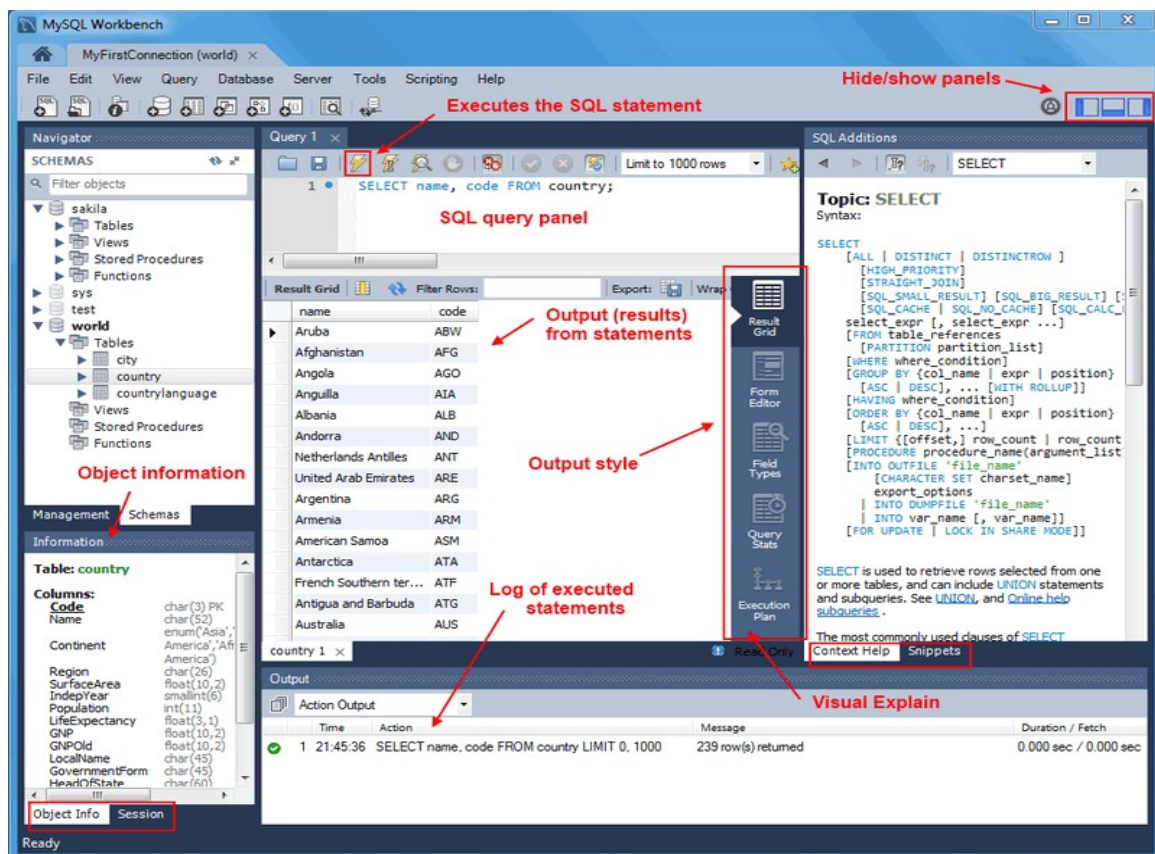


Figura 2.2: Entorno gráfico de MySQL Workbench

2.1.2.2 Oracle

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 *sopore 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.

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. Por otra parte, *PostgreSQL* hace más sencillo el análisis de datos y tiene una mayor seguridad, pero *Oracle* si soporta consultas en paralelo.

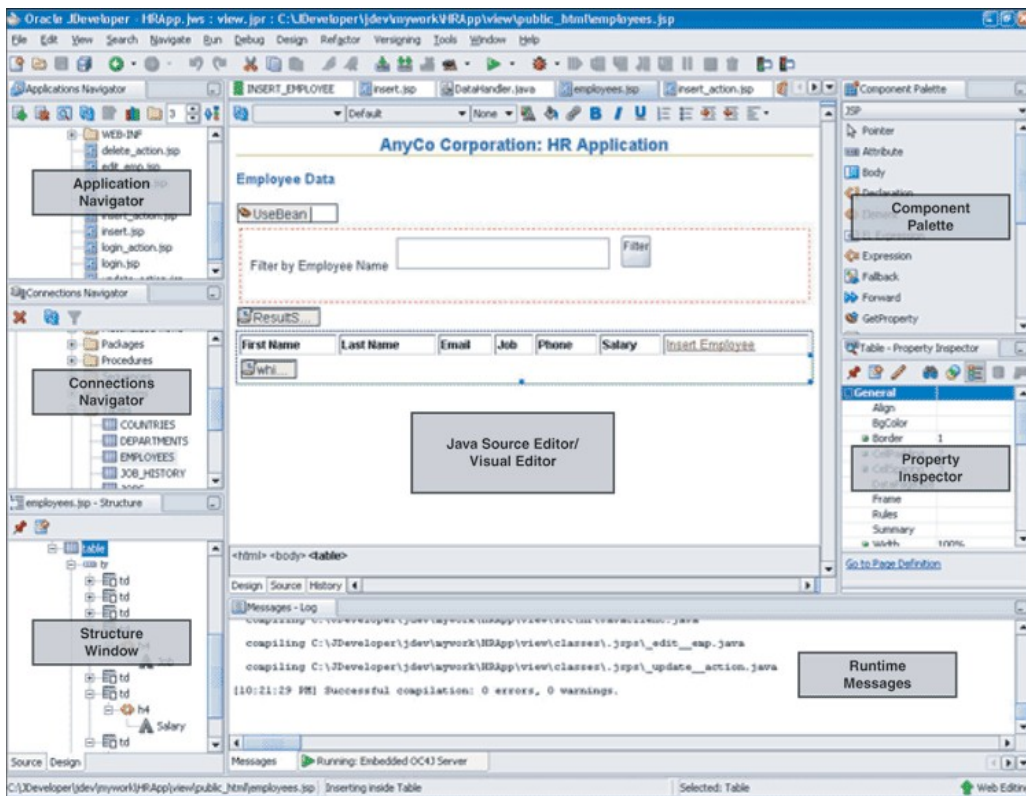


Figura 2.3: Entorno gráfico del entorno de desarrollo Oracle JDeveloper

2.1.2.3 Microsoft SQL Server

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*, una aplicación de las normas ANSI / ISO estándar Structured Query Language (SQL). Las características principales de este sistema son las siguientes:

- Soporte de transacciones.

- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor, donde la información y los datos se alojan en el servidor y las terminales o clientes de la red acceden a la información.
- Permite administrar información de otros servidores de datos.

Una de las diferencias principales de *PostgreSQL* con respecto a *Microsoft SQL Server* es que es **multiplataforma**; el primero puede ejecutarse en Linux, BSD y Windows, pero el segundo solo se puede ejecutar en Windows. Además, el primero posee una facilidad de uso mayor que el segundo. No obstante, *PostgreSQL* es más lento que *Microsoft SQL Server*.

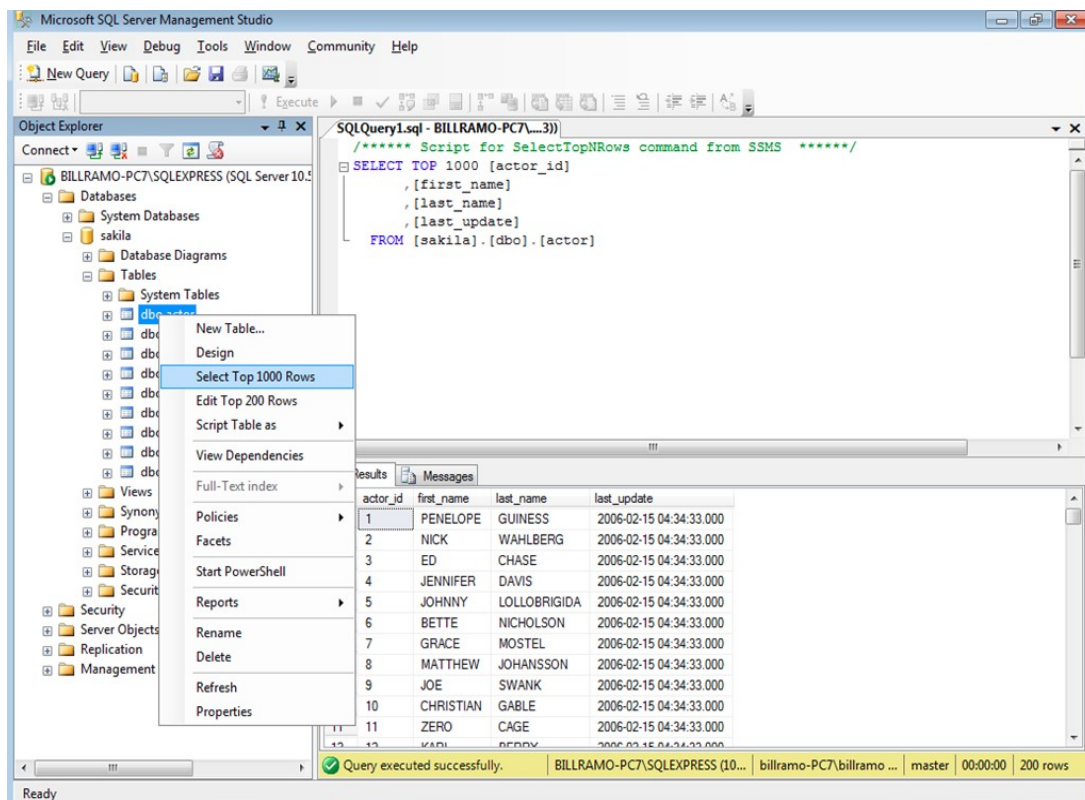


Figura 2.4: Entorno gráfico de Microsoft SQL Server Management Studio

2.2 Ciencia de datos

2.2.1 En el proyecto

2.2.1.1 Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación *multiparadigma*, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Aparte de esto, es un lenguaje *dinámico* y *multiplataforma*, muy adecuado para el desarrollo interactivo y la creación rápida de prototipos con la capacidad de soportar el desarrollo de grandes aplicaciones. También 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**: Un módulo para SciPy que permite trabajar eficientemente con datos en vectores y matrices.
- **Matplotlib**: Una biblioteca que permite crear gráficos en 2D y gráficos a partir de datos.
- **Pandas**: Una biblioteca que 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**. Se basa en el ecosistema de SciPy y lo requiere. 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*. Al igual que Python y SciPy, **scikit-learn** es de código abierto y es utilizable comercialmente bajo la licencia BSD.

2.2.2 Otras tecnologías posibles

Existen muchas más herramientas, aparte de la mencionada anteriormente, para poner en práctica el aprendizaje automático. A

continuación se nombran algunas de las más importantes.

2.2.2.1 RapidMiner

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. Se usa en investigación, educación, capacitación, creación rápida de prototipos y en aplicaciones empresariales. Las características principales de este software es que está *desarrollado en Java*, es *multiplataforma*, utiliza ficheros XML para la *representación interna de los procesos de análisis de datos*, permite el *desarrollo de programas a través de un lenguaje script*, puede *usarse de diversas maneras* (a través de GUI, en línea de comandos, en lotes y desde otros programas a través de llamadas a sus bibliotecas), es *extensible*, incluye gráficos y herramientas de *visualización de datos* y dispone de un *módulo de integración con el lenguaje R*.

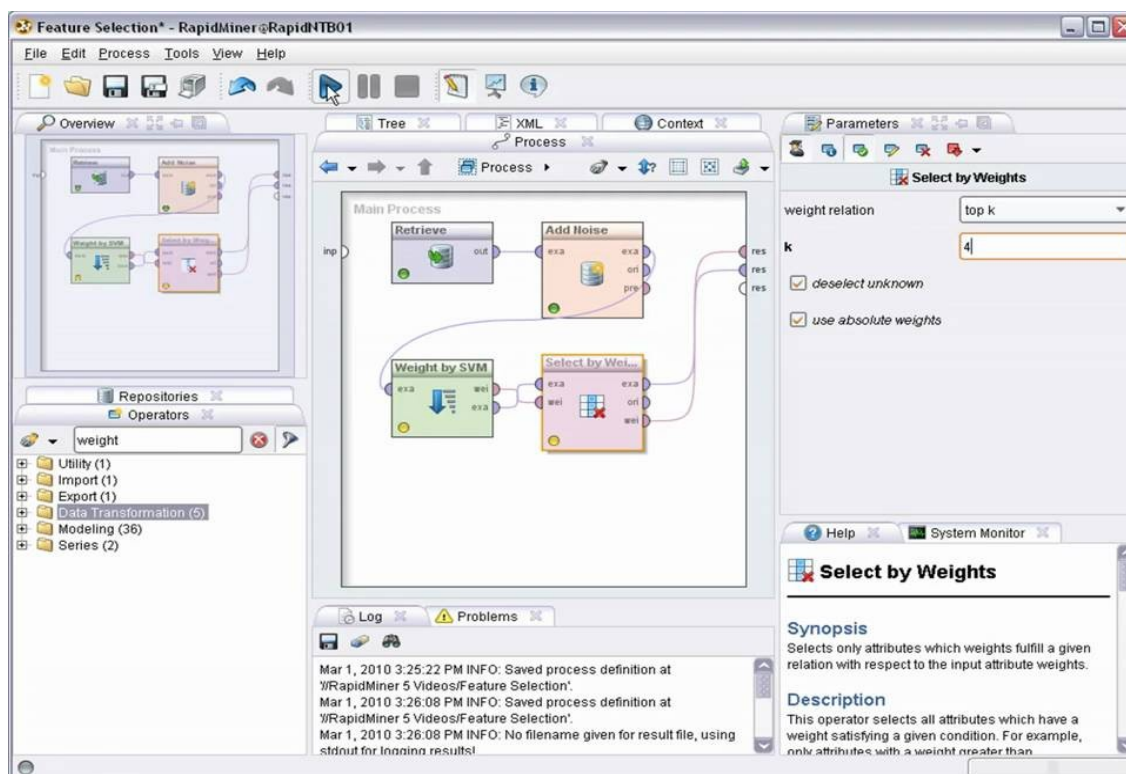


Figura 2.5: Entorno gráfico de RapidMiner

2.2.2.2.2.2 Lenguaje R

R es un entorno y un lenguaje de programación enfocado en el **análisis estadístico** de los más utilizados en el campo de la minería de datos que pueden aplicarse a gran variedad de disciplinas. Este lenguaje es un *proyecto*

colaborativo y abierto, por lo que los desarrolladores pueden descargar el código de forma gratuita y modificarlo para incluir mejoras. Por otra parte, es un *lenguaje interpretado*, funciona mediante comandos, proporciona una *amplia gama de herramientas estadísticas* que incluyen análisis de datos y generación de gráficos de alta calidad. Gracias a este lenguaje de programación los ingenieros de datos pueden *manejar grandes volúmenes 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.

2.2.2.3 Weka

Weka es un software libre y de código abierto basado en Java, licenciado bajo la GPL de GNU y disponible para su uso en Linux, Mac OS X y Windows. 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, permite importar los datos sin procesar desde varios formatos de archivo y soporta algoritmos bien conocidos para diferentes acciones de minería de datos como *filtrado, agrupación, clasificación y selección de atributos*. Este software también proporciona un *Java Appetiser* para su uso en aplicaciones y puede conectarse a bases de datos utilizando *CJD*.

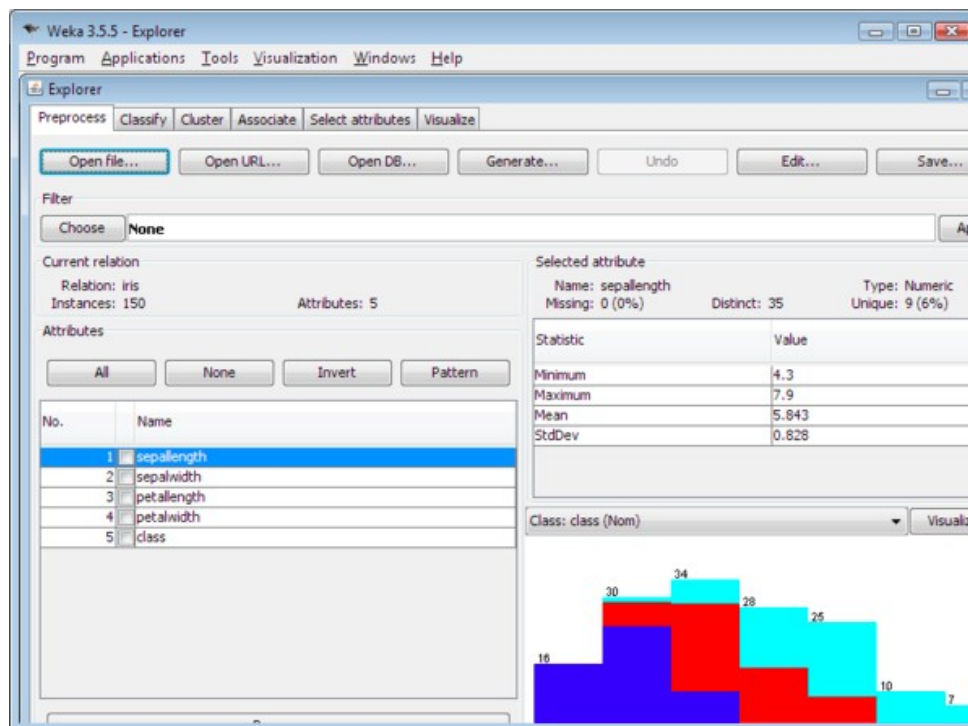


Figura 2.6: Entorno gráfico de Weka

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. 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, Knowledge Discovery in Databases**), que es el proceso general de conversión de datos brutos en información útil, como se muestra en la Figura 3.1. Este proceso consiste en una serie de pasos de transformación, desde el preprocesamiento de datos hasta el postprocesamiento de los resultados de la minería de datos.

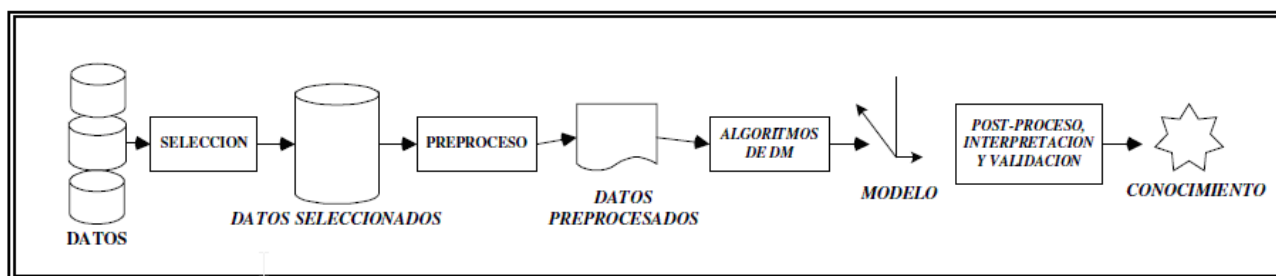


Figura 3.1: El proceso de descubrimiento de conocimiento en bases de datos (KDD)

- Los **datos de entrada** pueden almacenarse en una variedad de

formatos (archivos planos, hojas de cálculo o tablas relacionales) y pueden residir en un repositorio de datos centralizado o distribuirse en varios sitios.

- El propósito del **preprocesamiento** es *transformar los datos* de entrada brutos en un formato apropiado para el análisis posterior. Los pasos involucrados en el preprocesamiento de datos incluyen la fusión de datos de múltiples fuentes, la limpieza de datos para eliminar el ruido y la duplicación de observaciones, y la selección de registros y características que son relevantes para la tarea de minería de datos a mano. Debido a las muchas maneras en que los datos pueden ser recolectados y almacenados, el *preprocesamiento* de datos es quizás el paso más laborioso y que consume más tiempo en el proceso general de descubrimiento de conocimiento.
- El objetivo del siguiente proceso es **integrar los resultados de la minería de datos** en los sistemas de apoyo a la toma de decisiones. Es la **fase de modelamiento** propiamente tal, en donde métodos inteligentes son aplicados con el objetivo de extraer patrones previamente desconocidos, válidos, nuevos, potencialmente útiles y comprensibles y que están contenidos u “ocultos” en los datos.
- La fase anterior (de integración de resultados) requiere una etapa de **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*.
- **Tareas descriptivas.** El objetivo es derivar patrones (correlaciones, tendencias, clusters, trayectorias y anomalías) que resumen las relaciones subyacentes en los datos. Las tareas descriptivas de minería de datos son a menudo de naturaleza exploratoria y con frecuencia requieren técnicas de postprocesamiento para validar y explicar los

resultados

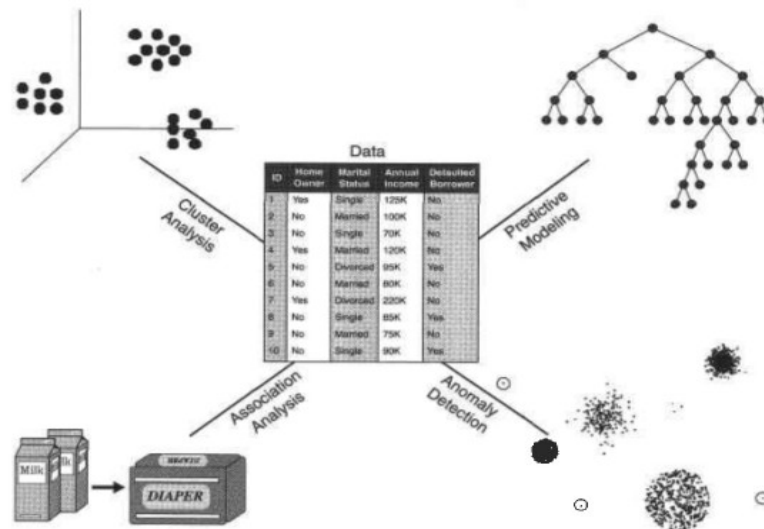


Figura 3.2: Cuatro de las tareas principales de minería de datos

En estas dos categorías se engloban cuatro de las tareas de minería de datos principales:

- **Modelado predictivo:** Se refiere a la tarea de construir un modelo para la variable objetivo en función de una serie de variables explicativas. Existen dos tipos de tareas de modelado predictivo: la *clasificación*, que se utiliza para las variables objetivo discretas, y la *regresión*, que se utiliza para las variables objetivo continuas. Por ejemplo, predecir si un usuario Web realizará una compra en una librería en línea es una tarea de clasificación porque la variable objetivo es binaria. Por otra parte, la previsión del precio futuro de un stock es una tarea de regresión porque el precio es un atributo de valor continuo. El objetivo de ambas tareas es aprender un modelo que minimice el error entre el valor predicho y el verdadero de la variable objetivo.
- **Análisis de asociaciones:** Se utiliza para descubrir patrones que describan características fuertemente asociadas en los datos. Los patrones descubiertos se representan típicamente en forma de reglas de implicación o subconjuntos de características. Debido al tamaño exponencial de su espacio de búsqueda, el objetivo del análisis de asociaciones es extraer los patrones más interesantes de manera eficiente. Las aplicaciones útiles del análisis de asociaciones incluyen la búsqueda de grupos de genes que tienen funcionalidad relacionada, la identificación de páginas Web a las que se accede de forma conjunta o la

comprensión de las relaciones entre los diferentes elementos del sistema climático de la Tierra.

- **Análisis de clústeres:** El objetivo es 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. La agrupación en clústeres se ha utilizado para agrupar conjuntos de clientes relacionados, encontrar áreas de océanos que tienen un impacto significativo en el clima de la Tierra y comprimir datos.
- **Detección de anomalías:** Es la tarea de identificar observaciones cuyas características son significativamente diferentes del resto de los datos. Estas observaciones se conocen como *anomalías* o *valores atípicos*. El objetivo de un algoritmo de detección de anomalías es descubrir las anomalías reales y evitar etiquetar falsamente los objetos normales como anómalos. En otras palabras, un buen detector de anomalías debe tener un alto índice de detección y un bajo índice de falsas alarmas. Las aplicaciones de detección de anomalías incluyen la detección de fraudes, intrusiones en la red, patrones inusuales de enfermedades y perturbaciones en los ecosistemas.

3.2 Técnicas de minería de datos

Para realizar las predicciones del tiempo promedio de viaje y el volumen de tráfico oportunas propuestas por la competición *KDDCup 2017* 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 XGBOOST

3.2.1.1 Definición

XGBoost (*eXtreme Gradient Boosting*) es una implementación de árboles de decisión potenciados por gradientes, diseñados para lograr una velocidad y un rendimiento dominantes y competitivos en el aprendizaje automático. Por lo tanto, la biblioteca está centrada en la velocidad de cálculo y el rendimiento del modelo. Este algoritmo es realmente rápido en comparación con otras implementaciones de potenciación del gradiente.

Para entender esta técnica de minería de datos es de gran importancia comprender en qué consiste la **potenciación del gradiente** (o *gradient boosting*).

3.2.1.2 Gradient boosting

La **potenciación del gradiente** (o *gradient boosting*) es una de las técnicas más poderosas para construir modelos predictivos. 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, normalmente *árboles de decisión*. Construye el modelo de forma escalonada como hacen otros métodos de refuerzo, y los generaliza permitiendo la optimización arbitraria de una función de pérdida diferenciable.

Este algoritmo es un algoritmo 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.

La potenciación del gradiente implica tres elementos:

- Una **función coste** a optimizar. Esta función asigna un *evento* o valores de una o más variables en un número real que representa intuitivamente algún "coste" asociado al evento. Ésta debe ser *diferenciable* (una función diferenciable de una variable real es una función cuya derivada existe en cada punto de su dominio). Un problema de optimización busca minimizar una función de pérdida.
- Un **weak learner** para hacer predicciones. Los árboles de decisión se utilizan como el **weak learner** en la potenciación del gradiente. Específicamente se utilizan árboles de regresión que emiten valores reales para las particiones y cuya salida puede sumarse, permitiendo que

las salidas de los modelos subsiguientes se sumen y "corrijan" los residuos en las predicciones. Los árboles se construyen de una manera codiciosa, eligiendo los mejores puntos de división en función de las puntuaciones de pureza como *Gini* o para minimizar el error. No obstante, es común restringir los **weak learners** de manera específica (un número máximo de capas, nodos, divisiones o nodos hoja) para asegurar que permanezcan débiles pero que aún puedan ser construidos de una manera codiciosa, como veremos más adelante. Es necesario que los modelos que se vayan generando permanezcan débiles ya que, si se sobreajusta a los datos, no habrá ningún residuo o error para los modelos subsiguientes sobre los que construir.

- Un **modelo aditivo** para añadir **weak learners** para minimizar la función de error. Los árboles se añaden uno a la vez y los árboles existentes en el modelo no se modifican. Se utiliza un procedimiento de *descenso por gradiente* para minimizar el error al añadir árboles. Tradicionalmente, el *descenso en gradiente* se utiliza para minimizar un conjunto de parámetros, como los coeficientes en una ecuación de regresión o los pesos en una red neuronal. Después de calcular el error, los pesos se actualizan para minimizar ese error. En lugar de parámetros, tenemos *sub-modelos de aprendizaje débiles* o, más específicamente, árboles de decisión. Después de calcular el error, para realizar el procedimiento de descenso por gradiente, debemos añadir un árbol al modelo que reduzca la pérdida (es decir, seguir el gradiente). Hacemos esto parametrizando el árbol, luego modificando los parámetros del árbol y después moviéndonos en la dirección correcta reduciendo la pérdida residual. Se agrega un número fijo de árboles o se detiene el entrenamiento una vez que la pérdida alcanza un nivel aceptable o ya no mejora un conjunto de datos de validación externa.

La potenciación del gradiente es un algoritmo codicioso y puede sobreajustar rápidamente un conjunto de datos de entrenamiento. Ante esto, esta técnica de aprendizaje automático puede beneficiarse de los **métodos de regularización** que penalizan varias partes del algoritmo y, en general, mejoran el rendimiento del algoritmo al reducir el sobreajuste. Algunas mejoras que se aplican a la potenciación del gradiente son las siguientes:

- **Restricciones de los árboles:** Es importante que los **weak learners** tengan destreza pero permanezcan débiles. Hay varias maneras en que los árboles pueden ser restringidos. Una buena heurística general es que mientras más restringida sea la creación de árboles, más árboles necesitará en el modelo, y al revés, cuanto

menos árboles individuales sean restringidos, menos árboles se necesitarán. Algunas de las restricciones que se imponen a la construcción de estos árboles son el *número de árboles*, *profundidad del árbol*, *número de nodos* o *número de hojas* y el *número de observaciones por división* (impone una restricción mínima en la cantidad de datos de entrenamiento en un nodo de entrenamiento antes de que se pueda considerar una división)

- **Velocidad de aprendizaje:** Las predicciones de cada árbol se suman secuencialmente y la contribución de cada árbol a esta suma puede ser ponderada para ralentizar el aprendizaje por el algoritmo. Esta ponderación se denomina *velocidad de aprendizaje* y cada actualización se escala por el valor de este parámetro. El efecto es que el aprendizaje se ralentiza, lo que a su vez requiere que se añadan más árboles al modelo, lo que a su vez lleva más tiempo entrenar, proporcionando un compromiso de configuración entre el *número de árboles* y el *ritmo de aprendizaje*. Esto es, disminuir el valor del ritmo de aprendizaje aumenta el mejor valor para el número de árboles; es común tener valores pequeños en el rango de 0.1 a 0.3, así como valores menores a 0.1. De esta manera, el parámetro de *velocidad de aprendizaje* reduce la influencia de cada árbol individual y deja espacio para que los árboles futuros mejoren el modelo.
- **Muestreo aleatorio:** En cada iteración se extrae una submuestra de los datos de entrenamiento al azar (sin reemplazo) del conjunto completo de datos de entrenamiento. La submuestra seleccionada al azar se utiliza entonces, en lugar de la muestra completa, para adaptarse al **base learner**. El beneficio de esto es que reduce la correlación entre los árboles en la secuencia en modelos de *potenciación del gradiente*. Esta variación de la potenciación del gradiente se denomina *potenciación del gradiente estocástico* y algunas variantes que se pueden utilizar de este algoritmos son el *submuestreo de filas antes de crear cada árbol*, *submuestreo de columnas antes de crear cada árbol* y el *submuestreo de columnas antes de considerar cada división*. En general, el submuestreo agresivo, como seleccionar sólo el 50% de los datos, ha demostrado ser beneficioso.
- **Aprendizaje penalizado:** Se pueden imponer restricciones adicionales a los árboles parametrizados aparte de modificar su estructura. Los árboles de decisión clásicos no se utilizan como **weak learners**, sino que se utiliza una forma modificada denominada *árbol de regresión* que tiene valores numéricos

(denominados *pesos*) en los nodos hoja (también llamados *nodos terminales*). Como tal, los valores de peso de las hojas de los árboles pueden ser regularizados usando una serie de *funciones de regularización*, tales como la *regularización L1 en los pesos* y la *regularización L2 en los pesos*. El término adicional de regularización ayuda a suavizar los pesos finales aprendidos para evitar sobreajustes.

3.2.1.3 Implementación del algoritmo

La implementación del algoritmo fue diseñada para conseguir la mejor utilización eficiente de los recursos de tiempo y memoria de computación. para entrenar el modelo. Algunas de las características clave de implementación del algoritmo incluyen:

- **Implementación de un algoritmo Sparse Aware** que puede tratar matrices dispersas, ahorrando memoria (sin necesidad de matrices densas) y tiempo de computación (los valores a cero se manejan de una forma especial).
- **Paralelización de la construcción de árboles** utilizando todos los núcleos de la CPU durante el entrenamiento.
- **Computación distribuida** para entrenar conjuntos de datos muy grandes utilizando un clúster de máquinas.
- **Computación fuera del núcleo** en una sola máquina para tratar grandes conjuntos de datos que no caben en la memoria. Se utiliza una solución de almacenamiento de datos denominado *bloque de columnas*. Esta solución organiza los datos por columnas, de tal forma que se ahorra tiempo al extraer los datos del disco tal y como lo espera el algoritmo de optimización (que funciona en columnas vectoriales).
- **Optimización de las estructuras de datos y algoritmos de la memoria caché** para un mejor uso del hardware.
- **Estructura de bloques** para soportar la paralelización de la construcción de árboles.
- **Entrenamiento continuado** para que se pueda seguir impulsando un modelo ya ajustado con nuevos datos.
- **Tratamiento de datos que faltan** de una forma efectiva. Otros métodos de combinación de árboles requieren primero datos faltantes con el objetivo de desarrollar una ramificación apropiada del árbol para tratar dichos valores. *XGBoost*, en su lugar, primer ajusta todos los valores no faltantes y, despues de haber creado la ramificación de la

variable, decide qué rama es la mejor que deben escoger otros valores faltantes para minimizar el error de predicción. Esta aproximación conduce tanto a árboles más compactos como a una estrategia de imputación eficaz, lo que conlleva un mayor poder de predicción.

3.2.2 LightGBM

3.2.2.1 Definición

LightGBM es un framework de potenciación del gradiente que 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 el otro algoritmo 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). Es decir, dado que **LightGBM** se basa en *algoritmos de árbol de decisión*, divide la hoja del árbol **leaf-wise** con el mejor ajuste, mientras que otros algoritmos de refuerzo dividen la profundidad del árbol **depth-wise** o **level-wise** en lugar de **leaf-wise**.

En comparación con el crecimiento **depth-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 que raramente puede lograrse con cualquiera de los algoritmos de *boosting* existentes. Además, es sorprendentemente muy 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:

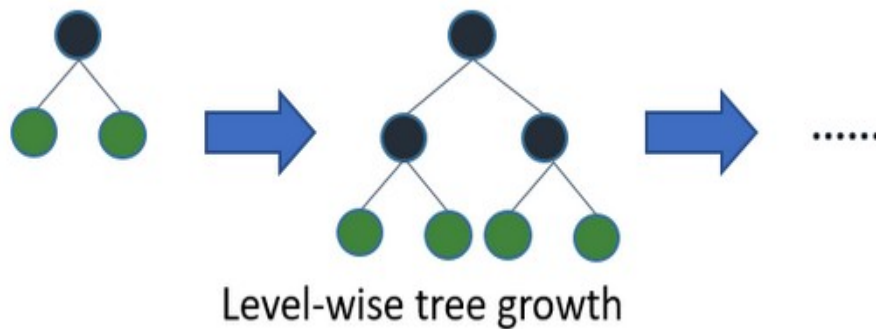


Figura 3.3: Crecimiento level-wise del árbol en XGBoost

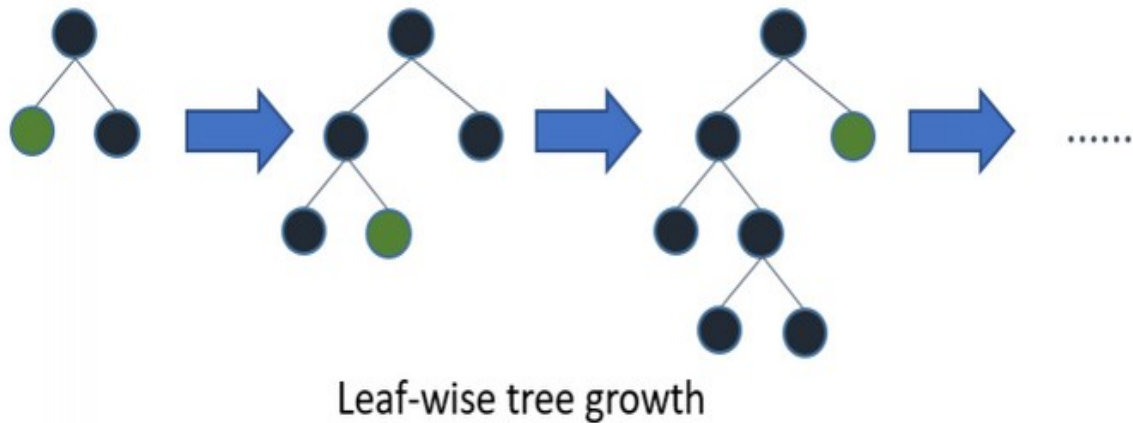


Figura 3.4: Crecimiento leaf-wise del árbol en LightGBM

3.2.2.2 3.2.2.2 Ventajas

Las ventajas que presenta *LightGBM* con respecto a otros algoritmos de *boosting* son las siguientes:

- **Mayor velocidad de entrenamiento y mayor eficiencia:** *LightGBM* utiliza un *algoritmo basado en histogramas*, es decir, almacena valores de características continuas en contenedores discretos que fijan el procedimiento de entrenamiento.
- **Menor uso de memoria:** Reemplaza valores continuos a contenedores discretos, lo que resulta en un menor uso de memoria.
- **Mayor precisión que cualquier otro algoritmo de *boosting*:** Genera árboles mucho más complejos al seguir un enfoque de división

leaf-wise en lugar de un enfoque **level-wise**, que es el factor principal para lograr una mayor precisión. Sin embargo, a veces puede llevar a un sobreajuste del modelo que puede evitarse configurando el parámetro de *profundidad máxima*.

- **Compatibilidad con grandes conjuntos de datos:** Es capaz de funcionar igual de bien con grandes conjuntos de datos con una reducción significativa del tiempo de entrenamiento en comparación con XGBoost.
- **Aprendizaje paralelo soportado.**

No es aconsejable utilizar *LightGBM* en pequeños conjuntos de datos. Este algoritmo es sensible al sobreajuste y puede sobreajustar datos pequeños fácilmente. No hay umbral en el número de filas, pero es recomendable usarlo sólo para datos con más de 10.000 filas.

3.2.3 Perceptrón multicapa (Redes neuronales)

3.2.3.1 Definición

El campo de las redes neuronales artificiales a menudo se llama simplemente **redes neuronales** o **perceptrones multicapa**. Un *perceptrón* es un modelo de una sola neurona que fue precursor de redes neuronales de mayor tamaño. Es un campo que investiga cómo modelos simples de cerebros biológicos pueden ser usados para resolver tareas computacionales difíciles como las tareas de modelado predictivo. El objetivo no es crear modelos realistas del cerebro, sino desarrollar algoritmos robustos y estructuras de datos que podamos usar para modelar problemas difíciles.

El poder de las redes neuronales proviene de su capacidad para aprender la representación en sus datos de entrenamiento y cómo relacionarla mejor con la variable de salida que desea predecir. En este sentido las redes neuronales aprenden un **mapeo**. Matemáticamente, son capaces de aprender cualquier función de mapeo y han demostrado ser un algoritmo de aproximación universal.

3.2.3.2 Neuronas

El bloque de construcción de las redes neuronales son las *neuronas artificiales*. Éstas son unidades computacionales simples que ponderan las señales de entrada y producen una señal de salida usando una función de

activación.

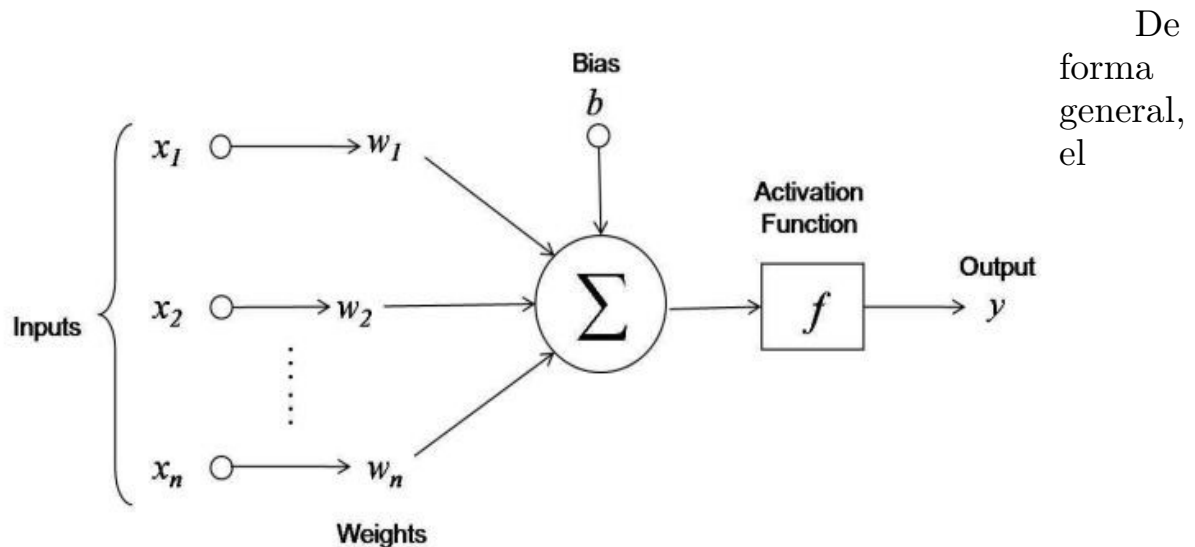


Figura 3.5: Estructura de una neurona artificial
funcionamiento de una neurona artificial es el siguiente:

1. Recibe una serie de **entradas**, que pueden ser características de un conjunto de entrenamiento o salidas de otras neuronas.
2. A continuación, se aplican unos **pesos** a las entradas. Estos pesos se inicializan a menudo a valores aleatorios pequeños, como valores en el rango de 0 a 0.3 , aunque se pueden utilizar esquemas de inicialización más complejos. Es deseable mantener pesos pequeños en la red y se pueden utilizar técnicas de regularización para ello.
3. Después, las entradas ponderadas se suman junto con un *sesgo* 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) y 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. Se llama función de activación porque gobierna el umbral a partir del cual se activa la neurona y la fuerza de la señal de salida. 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. Tradicionalmente se utilizan funciones de activación no

lineales puesto que permiten a la red neuronal combinar las entradas de maneras más complejas y, a su vez, proporcionar una mejor capacidad en las funciones que pueden modelar.

3.2.3.3 Redes de neuronas

Las neuronas están dispuestas en **redes neuronales**. Una fila de neuronas se denomina **capa** y una red puede tener múltiples capas. La arquitectura de las neuronas en la red a menudo se denomina **topología de red**.

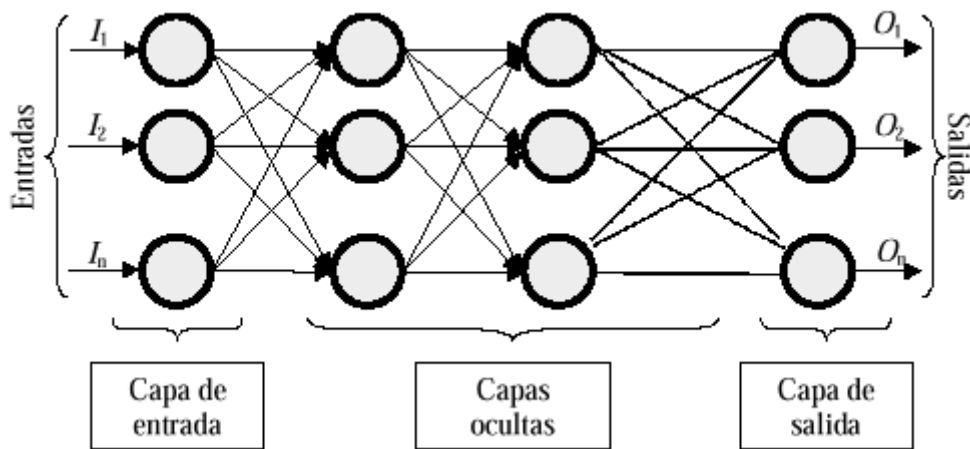


Figura 3.6: Estructura de una red neuronal

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

- **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. A menudo una red neuronal se representa con una capa visible con una neurona por valor de entrada o columna en el conjunto de datos. 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. La estructura de red neuronal más simple es tener

una sola neurona en la capa oculta que produzca directamente el valor de salida. Dado el aumento de la potencia de cálculo y la eficiencia de las bibliotecas, se pueden construir redes neuronales muy profundas. El *aprendizaje profundo* se refiere a tener muchas capas ocultas en una red neuronal. Son profundas porque habrían sido inimaginablemente lentas para entrenar históricamente, pero pueden tardar segundos o minutos para entrenar dichas redes neuronales usando técnicas y hardware modernos.

- **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. Por ejemplo, un problema de *regresión* puede tener una neurona de salida única y la neurona puede no tener función de activación. Otro ejemplo sería un problema de *clasificación binaria*, que puede tener una neurona de salida única y utilizar una función de **activación sigmoide** para emitir un valor entre 0 y 1 para representar la probabilidad de predecir un valor para la clase 1. Esto se puede convertir en un valor de clase definido utilizando un umbral de 0.5 y ajustar valores inferiores al umbral a 0 y superiores a 1.

3.2.3.4 Entrenamiento de una red neuronal

Para comenzar a entrenar una red neuronal, es imprescindible primero preparar los datos. Éstos deben ser *numéricos*; si los datos son *categoricos*, como un atributo de sexo con los valores "masculino" y "femenino", se puede convertir en una representación de valores reales denominada **codificación en caliente**. Aquí es donde se añade una nueva columna para cada valor de clase (dos columnas en el caso del sexo de hombres y mujeres) y un 0 o 1 para cada fila dependiendo del valor de clase para esa fila.

Esta misma codificación en caliente se puede utilizar en la variable de salida en *problemas de clasificación* con más de una clase. Esto crearía un

vector binario a partir de una sola columna que sería fácil de comparar directamente con la salida de la neurona en la capa de salida de la red neuronal que, como se describió anteriormente, produciría un valor para cada clase.

Las redes neuronales requieren que la entrada esté escalada de manera consistente. Se puede rescalar al rango entre 0 y 1 y esto se denomina **normalización**. Otra técnica bastante utilizada es **estandarizarla** 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. Con la *normalización* y la *estandarización* el proceso de entrenamiento de la red neuronal se realiza con mucha mayor velocidad.

El clásico y aún preferido algoritmo de entrenamiento para redes neuronales se denomina ***descenso por gradiente estocástico***, en el que una fila de datos se expone a la red neuronal a la vez como entrada. 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. Es el tipo de propagación que también se utiliza después de que la red neuronal es entrenada para hacer predicciones sobre nuevos datos.

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. Esta propagación del error hacia atrás se denomina el *algoritmo de retropropagación*. El proceso se repite para todos los ejemplos de los datos de entrenamiento y un 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***. Puede resultar en cambios rápidos pero también caóticos en la red. 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*** y a menudo es más estable.

Típicamente, debido a que los conjuntos de datos son tan grandes y a las eficiencias computacionales, el *tamaño del lote* (el número de ejemplos que se le muestra a la red neuronal antes de una actualización), a menudo se reduce a un pequeño número, como decenas o cientos de ejemplos. 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 o 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.

3.2.4 Modelo ARIMA

Para comprender el funcionamiento del modelo ARIMA, resulta relevante entender los conceptos englobados dentro de lo que se denominan *series temporales*.

3.2.4.1 Definición de una serie temporal

Una **serie temporal** es una colección ordenada de mediciones tomadas en intervalos regulares; por ejemplo, los precios diarios de las acciones o los datos de ventas semanales. Los intervalos pueden representar cualquier unidad de tiempo, pero debe utilizarse un mismo intervalo para todas las mediciones. Además, si algún intervalo no tiene ninguna medición, debe definirse en el valor perdido. De esta forma, el número de intervalos con mediciones (incluidos los que tienen valores perdidos) define la duración del período histórico de los datos. Un ejemplo de serie temporal es el siguiente:

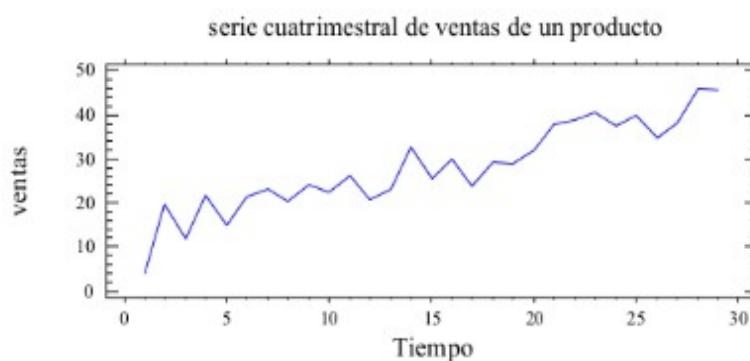


Figura 3.7: Ejemplo de serie temporal

3.2.4.2 Características de las series temporales

Estudiar el comportamiento pasado de una serie temporal ayuda a

identificar los patrones y realizar mejores previsiones. Cuando se representan, muchas series temporales muestran una o varias de estas características:

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

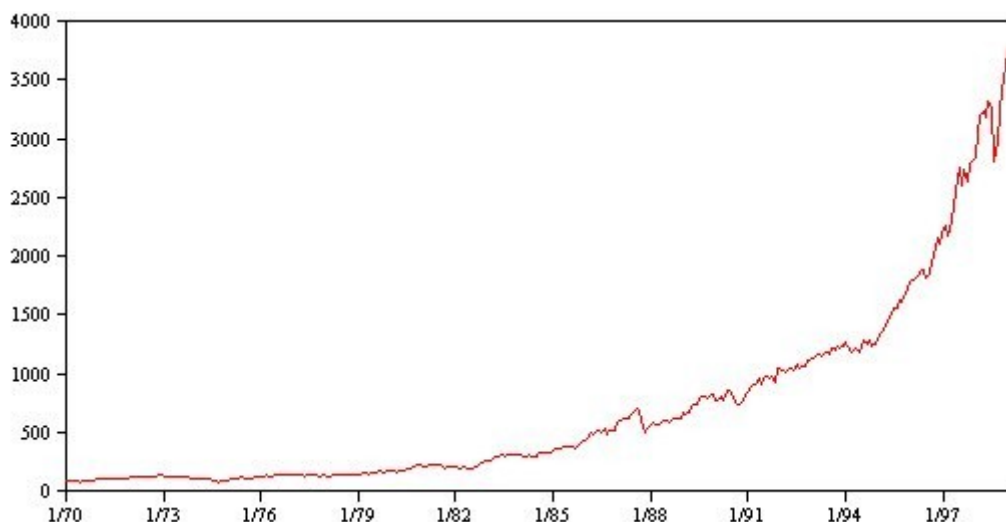


Figura 3.8: Ejemplo de una serie temporal con tendencia

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. Las tendencias lineales globales son adecuadas y hacen previsiones correctas con el modelo ARIMA.

- **Ciclos estacionales y no estacionales:** Un **ciclo estacional** es un patrón repetitivo y predecible de los valores de las series temporales. Por ejemplo, los datos mensuales suelen mostrar un comportamiento cíclico a lo largo de trimestres y años. Una serie mensual puede mostrar un ciclo trimestral significativo con un mínimo en el primer trimestre o un ciclo anual con un pico en cada mes de diciembre. Se dice que las series con un ciclo estacional muestran **estacionalidad**; los patrones estacionales resultan útiles para obtener buenos ajustes y previsiones.

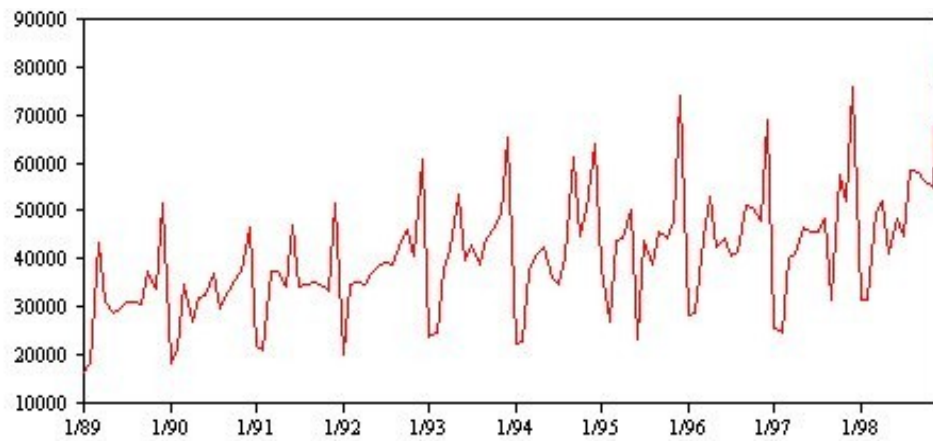


Figura 3.9: Ejemplo de una serie temporal con ciclos estacionales

Por otra parte, un **ciclo no estacional** es un patrón repetitivo y posiblemente impredecible de los valores de las series. Algunas series, como la tasa de desempleo, muestran un claro comportamiento cíclico; no obstante, la periodicidad del ciclo varía a lo largo del tiempo, por lo que resulta difícil predecir cuándo se van a producir máximos o mínimos. 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 *temporal*, o **pulso**, en el nivel de la serie, y un cambio repentino y *permanente*, o **paso**, en el nivel de la serie.

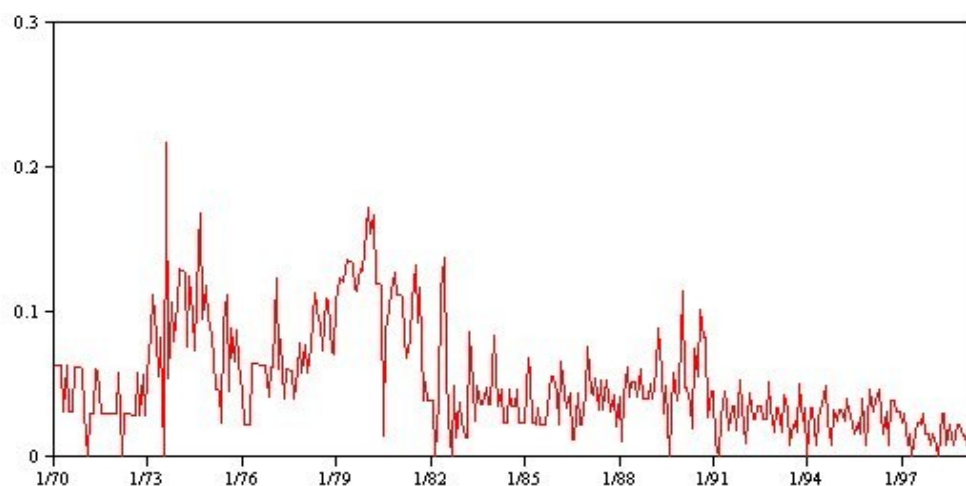


Figura 3.10: Serie temporal con pulsos

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**. Por ejemplo, puede que un comercio minorista descubra que sus ventas se incrementaron mucho más de lo normal un día que todos los artículos se rebajaron un 50%. Si se especifica una promoción de rebajas del 50% como **evento** recurrente, puede mejorar el ajuste del modelo y estimar la repercusión que tendría esa misma promoción en el futuro.

- **Valores atípicos:** Los desplazamientos en el nivel de una serie temporal que no se pueden explicar se denominan **valores atípicos**. Estas observaciones no coinciden con el resto de las series y pueden influir considerablemente en el análisis y, por lo tanto, afectar a la capacidad de previsión del modelo de serie temporal.

En las siguientes figuras se muestran los distintos tipos de valores atípicos que se producen normalmente en las series temporales. Las líneas azules representan una serie sin valores atípicos. Las líneas rojas sugieren un patrón que podría estar presente si la serie contuviera valores atípicos.

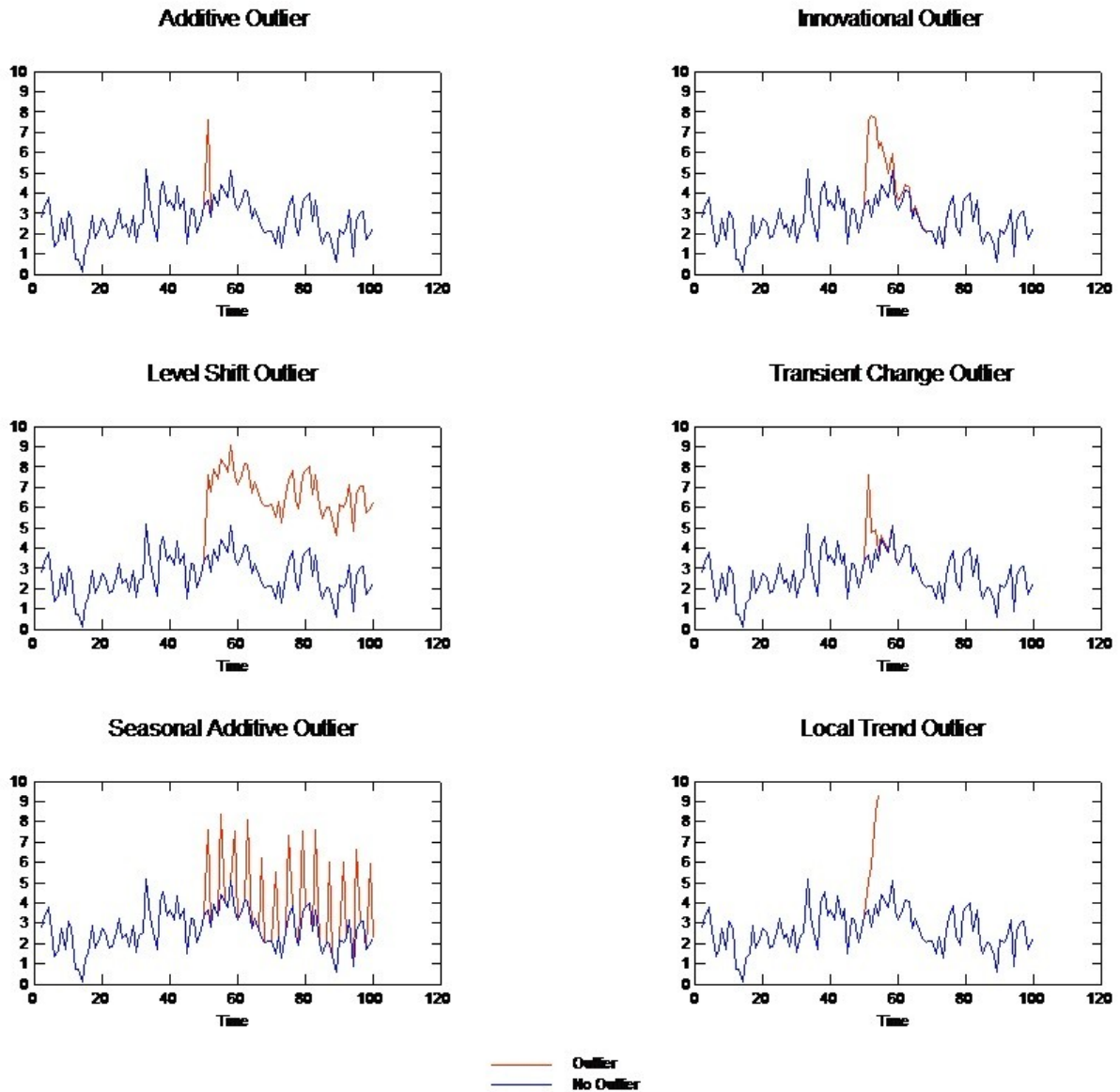


Figura 3.11: Distintos tipos de valores atípicos

- **Valor atípico aditivo.** Un valor atípico aditivo aparece como un valor inesperadamente alto o bajo que se produce para una única observación. Las siguientes observaciones no se ven afectadas por un valor atípico aditivo. Los valores atípicos aditivos consecutivos se denominan normalmente **parches de valores atípicos aditivos**.
- **Valor atípico innovador.** Un valor atípico innovador se caracteriza por un impacto inicial con efectos que se extienden sobre las siguientes observaciones. La influencia de los valores atípicos puede aumentar mientras avanza el tiempo.

- ***Valor atípico de cambio de nivel.*** En el cambio de nivel, todas las observaciones que aparecen después del valor atípico se desplazan a un nuevo nivel. A diferencia de los valores atípicos aditivos, un valor atípico de cambio de nivel afecta a diversas observaciones y tiene un efecto permanente.
- ***Valor atípico de cambio transitorio.*** Los valores atípicos de cambio transitorio son similares a los valores atípicos de cambio de nivel, pero su efecto se reduce exponencialmente en las siguientes observaciones. Finalmente, las series vuelven a su nivel normal.
- ***Valor atípico aditivo estacional.*** Un valor atípico aditivo estacional aparece como un valor inesperadamente alto o bajo que se produce repetidamente en intervalos regulares.
- ***Valor atípico de tendencia local.*** Un valor atípico de tendencia local produce un cambio general en la serie causado por un patrón en los valores atípicos después de la aparición del valor atípico inicial.

La detección de valores atípicos en una serie temporal implica determinar la ubicación, tipo y magnitud de todos los valores atípicos presentes. Tsay (1988) propuso un procedimiento iterativo para detectar el cambio del nivel de la media con el fin de identificar los valores atípicos deterministas. Este proceso implica la comparación de un modelo de serie temporal que supone que no hay presentes valores atípicos con otro modelo que incorpore valores atípicos. Las diferencias entre modelos permiten calcular el efecto de tratar cualquier punto como un valor atípico.

3.2.4.3 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 no siempre resulta ser el más adecuado, pero es interesante cuando en la serie se observa cierta tendencia o cierta periodicidad. Hay que resaltar que esta descomposición no es en general única.

Este enfoque descriptivo consiste en encontrar componentes que correspondan a una **tendencia a largo plazo**, un **comportamiento estacional** y una **parte aleatoria**. Las componentes o fuentes de variación que se consideran habitualmente son las siguientes:

- **Tendencia secular o regular:** Se puede definir como un *cambio a largo plazo* que se produce en relación al nivel medio, o el cambio a largo plazo de la media. La tendencia se identifica con un movimiento suave de la serie a largo plazo. La notaremos ***t***.
- **Efecto Estacional (Variación estacional):** Muchas series temporales presentan *cierta periodicidad* o, dicho de otro modo, variación de cierto periodo (anual, mensual ...). Por ejemplo, el paro laboral aumenta en general en invierno y disminuye en verano. Estos tipos de efectos son fáciles de entender y se pueden medir explícitamente o incluso se pueden eliminar del conjunto de los datos, desestacionalizando la serie original. La notaremos ***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. La notaremos ***r***.

Es necesario aislar de alguna manera la componente aleatoria y estudiar qué modelo probabilístico es el más adecuado. Conocido éste, podremos conocer el comportamiento de la serie a largo plazo.

3.2.4.4 Tipos de esquemas para series temporales

Las tres componentes enumeradas determinan conjuntamente los valores de la variable analizada en cada instante, sin que pueda valorarse con precisión el influjo individual de cada una de ellas. 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** supone que las observaciones se generan como suma de las tres componentes, es decir:

$$Y = t + e + r$$

En
este
caso
cada

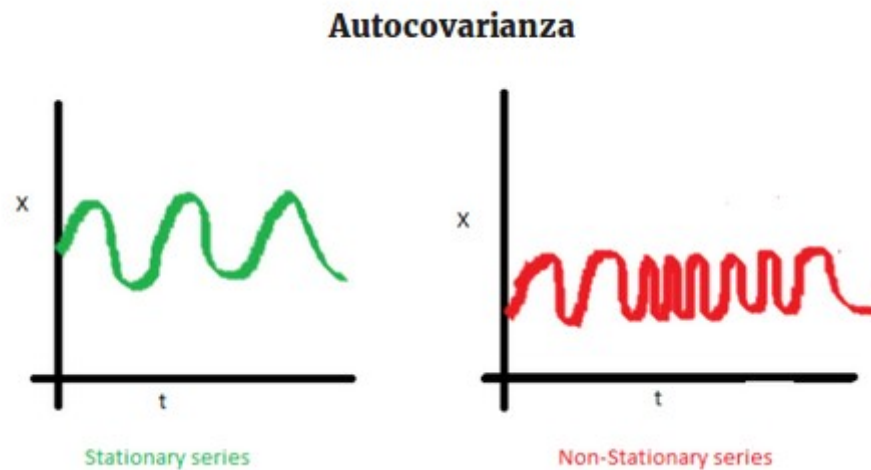


Figura 3.12: Comparación de una serie estacionaria con una no estacionaria con respecto a la autocovarianza

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). **¿Decir que éste es el modelo en el que nos vamos a basar?**

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

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

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:

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

Para diferenciar un esquema de otro, vamos a observar las siguientes imágenes:

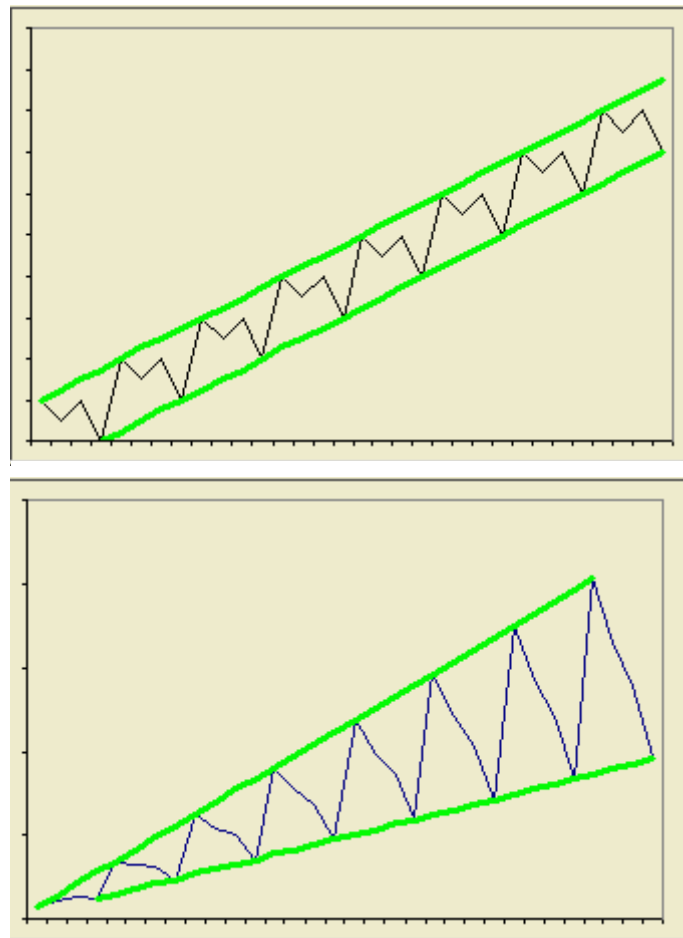


Figura 3.14: Serie temporal con tendencia
multiplicada por la estacionalidad

En la primera imagen, la *tendencia se ha agregado a la estacionalidad*. Por ello, la banda que ocupa la serie es siempre del mismo grosor. En la segunda imagen, la *tendencia se multiplica por la estacionalidad*. Por eso, el efecto de ésta es superior cuanto mayor sea la tendencia. Esta observación distingue el primer modelo de los dos últimos. La distinción entre el modelo multiplicativo y el mixto tiene que ver con el comportamiento de la componente irregular.

3.2.4.5 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*. Esto se refleja gráficamente en que los valores de la serie tienden a oscilar alrededor de una media constante y la variabilidad con respecto a esa media también permanece constante en el tiempo. Es una serie básicamente estable a lo largo del tiempo, sin que se aprecien aumentos o disminuciones sistemáticos de sus valores. En la serie temporal de la *Figura 3.14* se presenta una serie estacionaria discreta.

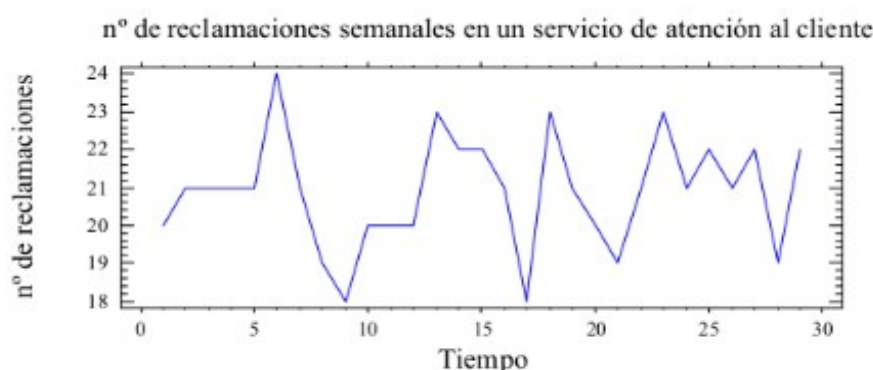


Figura 3.15: Ejemplo de serie estacionaria

- **No Estacionarias:** Son series en las cuales la 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. Por ejemplo, la serie de la *Figura 3.15* presenta una fuerte tendencia creciente aunque existen importantes oscilaciones con relación a esa tendencia de crecimiento lineal.

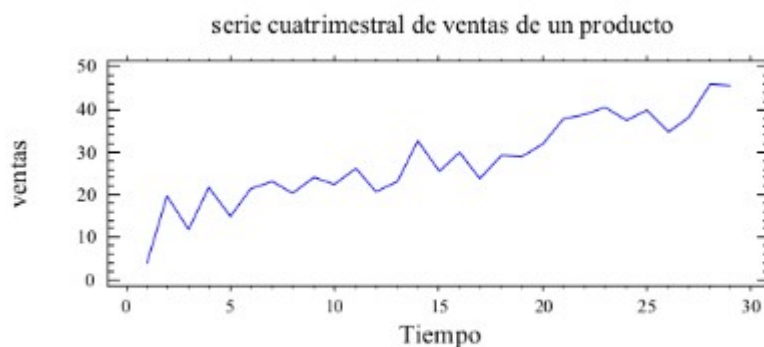


Figura 3.16: Ejemplo de serie no estacionaria

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

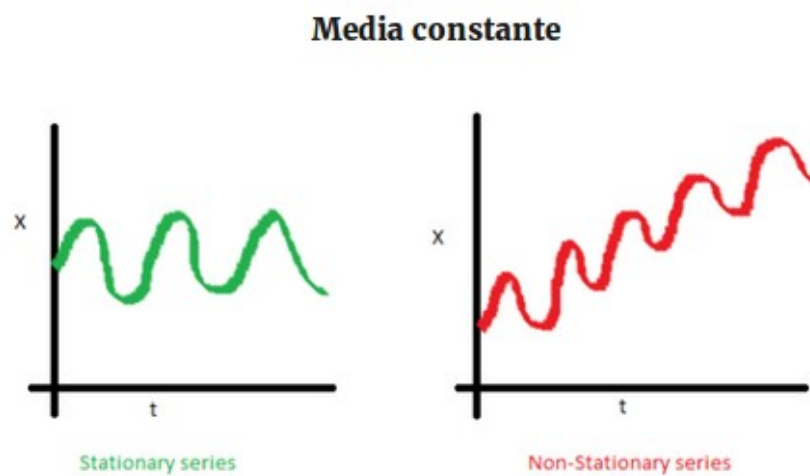


Figura 3.17: Comparación de una serie estacionaria con una no estacionaria con respecto a la media

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.

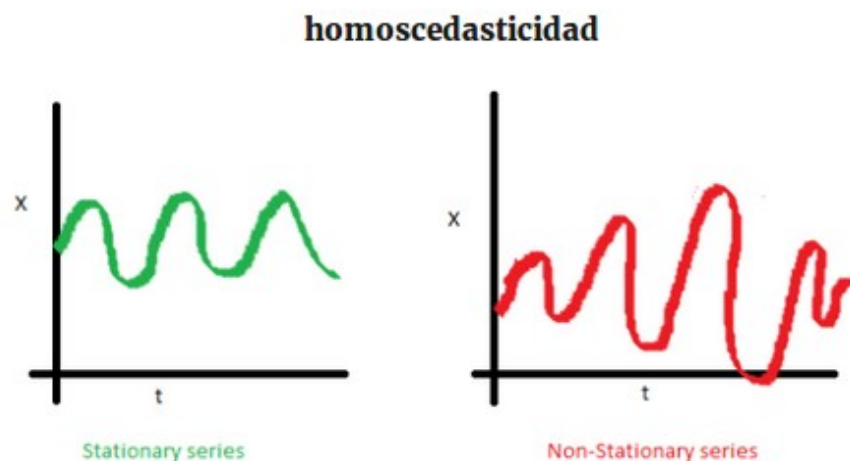


Figura 3.18: Comparación de una serie estacionaria con una no estacionaria con respecto a la varianza

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

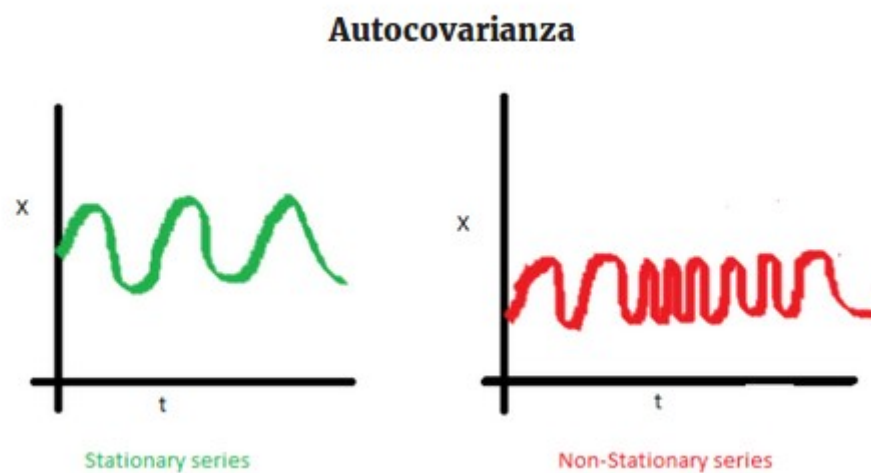


Figura 3.19: Comparación de una serie estacionaria con una no estacionaria con respecto a la autocovarianza

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

3.2.4.6 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 Y_t , distanciados en un lapso de tiempo 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.

Un ejemplo de gráfico de autocorrelación es el siguiente:

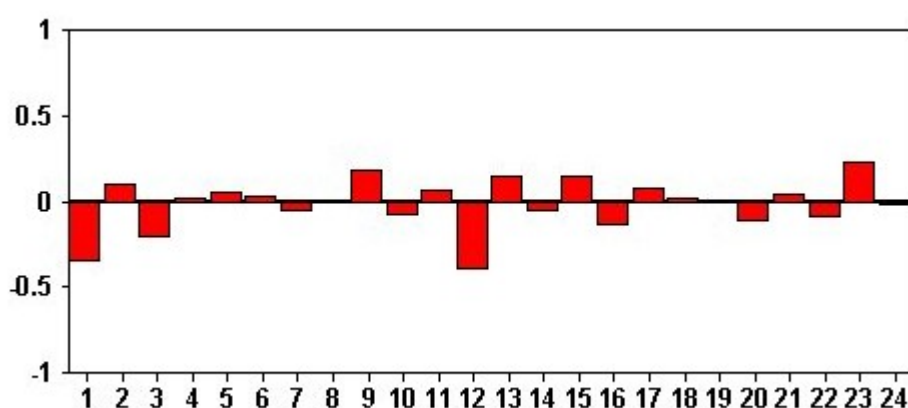


Figura 3.20: Ejemplo de un gráfico de autocorrelación

El eje x del gráfico indica el retardo en el que se calcula la autocorrelación; el eje y indica el valor de la correlación (entre -1 y 1). Una **correlación positiva** indica que los valores grandes actuales se corresponden con valores grandes en el retardo especificado; una **correlación negativa** indica que los valores grandes actuales se corresponden con valores pequeños en el retardo especificado. El *valor absoluto de una correlación* es una medida de la fuerza de la asociación, con valores absolutos mayores que indican relaciones más fuertes.

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

3.2.4.7 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.

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 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 $t = 1, 2, 3, \dots$. 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,

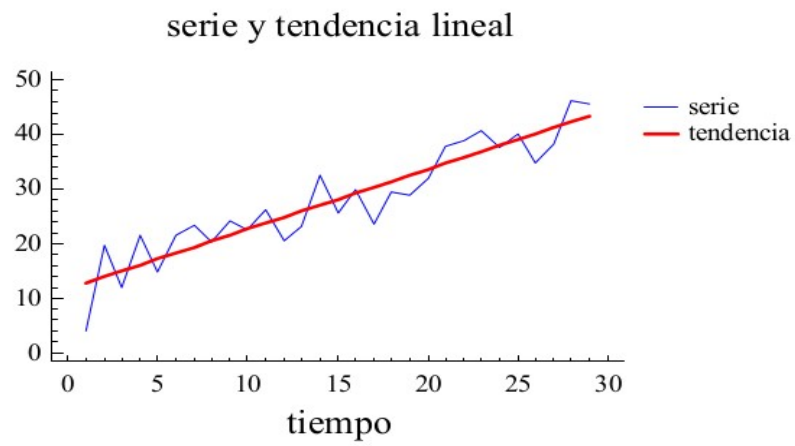


Figura 3.21: Serie temporal y su tendencia

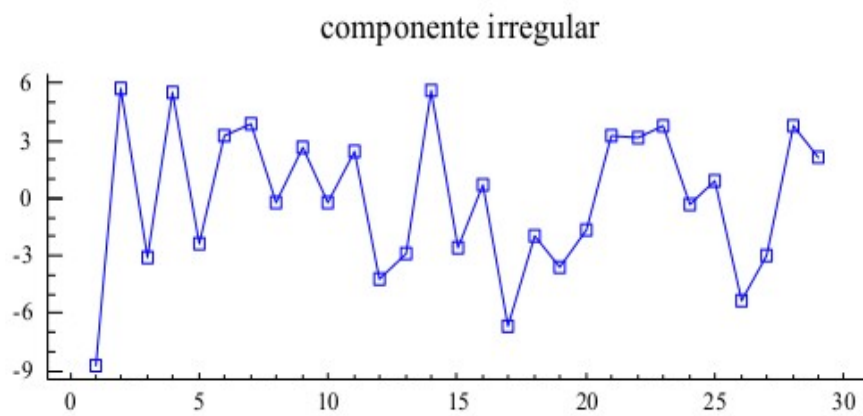


Figura 3.22: Componente regular de la serie temporal tras eliminar su tendencia

Capítulo 4

Fases del proyecto

Con el objetivo de desarrollar el proyecto de una forma ordenada, estructurada y adecuada, se ha procedido a dividir el mismo en una serie de fases:

- *Comprensión del problema planteado por la competición KDDCup 2017.* En primer lugar se ha llevado a cabo un estudio de las tareas encomendadas por la competición y de las reglas a aplicar en las predicciones que se realizaran.
- *Creación de una serie de bases de datos para almacenar los datos proporcionados por la competición.* Con el objetivo de poder acceder fácilmente y de forma flexible a la información suministrada por la competición KDDCup 2017, 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.
- *Modificación de las bases de datos creadas para adecuarlas a las tareas de predicción.* Una vez realizada la carga de las bases de datos, 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. En este aspecto, nos centramos en la modificación de los tipos de datos de los atributos de las distintas tablas de las bases de datos.
- *Creación de gráficas para visualizar 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 varias aproximaciones de predicciones del tiempo promedio de viaje.* 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 requeridos por la competición.
- *Realización de varias aproximaciones de predicciones del volumen de tráfico.* De igual manera que el tiempo promedio de viaje, también se procede a construir aproximaciones de predicciones del volumen de tráfico en las distintas barreras de peaje en las direcciones de entrada y salida y en los intervalos solicitados por la competición.
- *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 proporcionados por la competición 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* 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. Cada ventana de tiempo de 20 minutos se define como un intervalo semiabierto por la derecha; por ejemplo, [2016-09-18 23:40:00, 2016-09-19 00:00:00).

- **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*

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

*peaje-
dirección en
total.*

Figura 4.1: Cálculo del error de predicción del tiempo promedio de
viaje

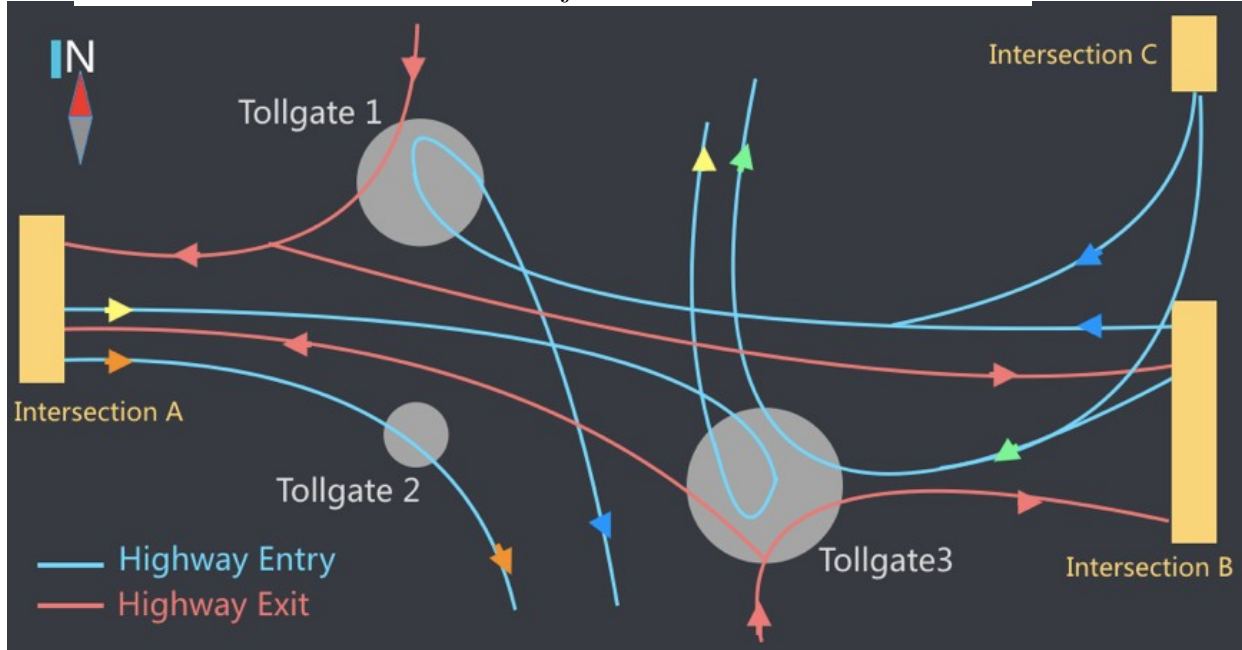


Figura 4.2: Topología de la red de carreteras de la zona objetivo

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*. 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**. **¿Meter lo de la segunda fase de predicción?**

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**. **¿Poner que más adelante se añada datos de entrenamiento del día 18 de Octubre hasta el 24 de Octubre o, en vez de esto, que se añaden estos datos**

para comprobar las predicciones (aunque falten datos de intervalos)?



Figura 4.3: Ventanas de tiempo para la predicción del tráfico

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. Por ejemplo, no se permite utilizar los datos de tráfico del día **20 de Octubre** para predecir el tráfico del día **19 de Octubre**.

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:

Esta fórmula se aplica a la primera tarea propuesta por la competición donde R , T , D , d_{rtd} y p_{rtd} son el *número de rutas*, el *número de ventanas de tiempo a predecir*, el *número de días en los que hay que predecir* y los *valores 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 C , T , D , f_{ctd} y p_{ctd} son el *número de pares barrera de peaje-dirección*, el *número de ventanas de tiempo a predecir*, el *número de días en los que hay que predecir* y los *valores 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 .

¿Mencionar el MAPE propuesto por la competición también sin los días?

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 denominado *PostgreSQL*.

4.2.1 Estructura de bases de datos propuesta

La información suministrada por la competición *KDDCup 2017* se ha almacenado en tres 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* y tiene el objetivo de ser un almacén de datos de referencia para las alteraciones pertinentes que se quieran llevar a cabo sobre la misma. 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 *tfgtest1*. Aparte, se ha generado otro almacén de datos denominado *tfgtraining2* para guardar los datos reales de los días a predecir; es decir, se almacenan los datos reales de los intervalos de tiempo a estimar para poder compararlos con los valores que se vayan a predecir.

¿Cambiar los nombres de las bases de datos *tfgtest1* y *tfgtraining2*?

¿Cambiar la finalidad de la base de datos *tfgtraining2* diciendo que en realidad eran los datos de la segunda fase de entrenamiento pero que en este caso los hemos utilizado para ... (lo que hemos puesto)?

4.2.2 Base de datos con los datos originales

4.2.2.1 Tabla *vehicle_routes* (“*routes_table4.csv*”)

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

Campo	Tipo	Descripción
<i>intersection_id</i>	string (char(1))	Identificador de la intersección
<i>tollgate_id</i>	string (char(1))	Identificador de la barrera de peaje
<i>link_seq</i>	string (varchar(47))	Secuencia de enlaces que conforman la ruta desde la intersección hasta la barrera de peaje

Tabla 4.1: Tabla *vehicle_routes*

¿Poner el nombre de la tabla en cursiva en el pie de tabla?

La red de carreteras utilizada en la competición es un grafo dirigido formado por *enlaces* o *tramos* de carreteras interconectados. Una ruta en la red está representada por una secuencia de tramos. 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*" (ver *Figura 4.3*)

4.2.2.2 Tabla *road_links* (“*links_table3.csv*”)

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

Campo	Tipo	Descripción
<i>link_id</i>	string (char(3))	Identificador del enlace
<i>length</i>	float	Longitud del enlace en metros
<i>width</i>	float	Anchura del enlace en metros
<i>lanes</i>	int	Número de carriles
<i>in_top</i>	string (varchar(7))	Este atributo contiene los enlaces entrantes al enlace actual, separados por comas
<i>out_top</i>	string (varchar(7))	Este atributo contiene los enlaces salientes del enlace actual, separados por comas
<i>lane_width</i>	float	Anchura de cada uno de los carriles del enlace en metros

Tabla 4.2: Tabla road_links

Esta tabla contiene la **descripción de cada uno de los tramos** que forman una ruta (ver *Figura 4.3*).

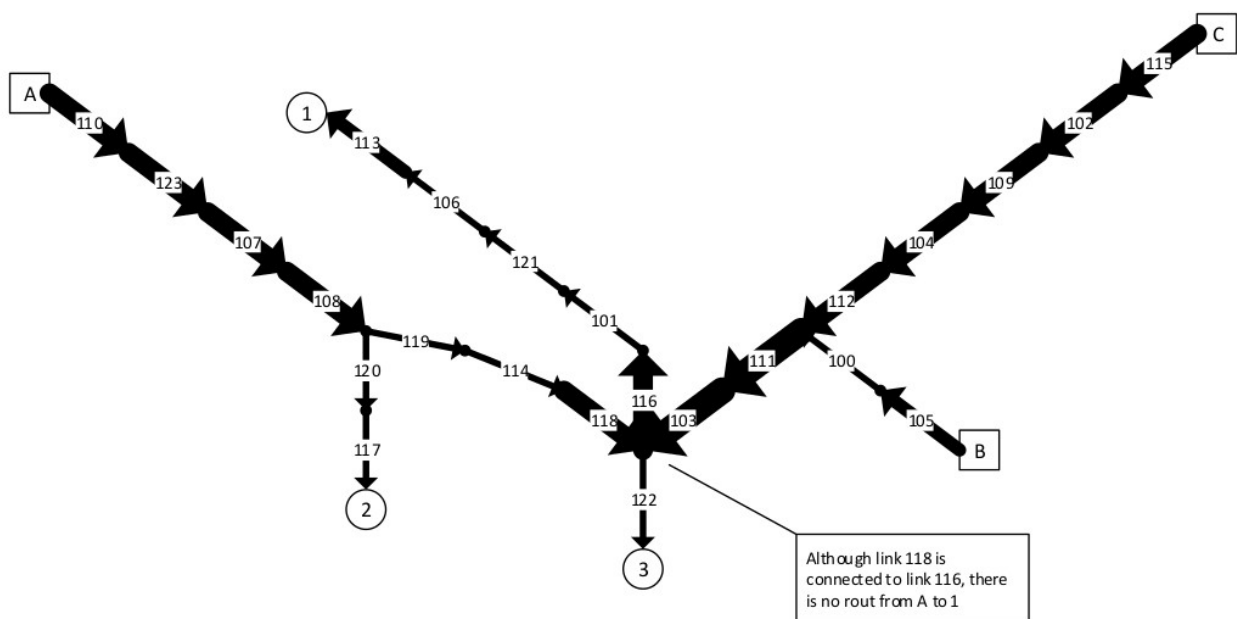


Figura 4.5: Rutas de la competición KDDCup2017 con los enlaces que las forman

¿Señalar el propietario de la imagen?

4.2.2.3 Tabla *vehicle_trajectories_training* (“*trajectories_table_5_training.csv*”)

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

Campo	Tipo	Descripción
<i>intersection_id</i>	string (char(1))	Identificador de la intersección
<i>tollgate_id</i>	string (char(1))	Identificador de la barrera de peaje
<i>vehicle_id</i>	string (varchar(30))	Identificador del vehículo
<i>starting_time</i>	datetime (timestamp)	Momento del tiempo en el que el vehículo entra en la ruta
<i>travel_seq</i>	string (varchar(400))	Trayectoria de la ruta formada por un conjunto de enlaces. Estos enlaces están separados por un “;” y, para cada enlace, se especifica, separados por “#”, su identificador, el momento del tiempo en el que el vehículo entra en ese enlace y el tiempo que pasa el vehículo atravesando dicho enlace en segundos.
<i>travel_time</i>	float	Tiempo total que tarda el vehículo en viajar desde la intersección hasta la barrera de peaje.

Tabla 4.3: Tabla vehicle_trajectories_training

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*
(“*volume_table6_training.csv*”)

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

Campo	Tipo	Descripción
<i>time</i>	datetime (timestamp)	Momento en el que un vehículo atraviesa la barrera de peaje
<i>tollgate_id</i>	string (char(1))	Identificador de la barrera de peaje
<i>direction</i>	string (char(1))	Dirección en la que el vehículo atraviesa la barrera de peaje. Si es 0, la dirección es de salida; si es 1, la dirección es de entrada.
<i>vehicle_model</i>	int	Modelo del vehículo. Este número (compendido entre los valores 0 y 7), cuanto mayor sea, mayor es su capacidad
<i>has_etc</i>	string (char(1))	Indica si el vehículo utiliza un ETC (Electronic Toll Collection)
<i>vehicle_type</i>	string(char(1))	Tipo de vehículo. Indica si el vehículo es de pasajeros o de carga

Tabla 4.4: Tabla *traffic_volume_tollgates_training*

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 considera importante.**

4.2.2.5 Tabla `weather_data` (“`weather (table 7)_training.csv`”)

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

Campo	Tipo	Descripción
<i>date</i>	date	Fecha
<i>hour</i>	int	Hora
<i>pressure</i>	float	Presión del aire (hPa)
<i>sea_pressure</i>	float	Presión del nivel del mar (hPa)
<i>wind_direction</i>	float	Dirección del viento (°)
<i>wind_speed</i>	float	Velocidad del viento (m/s)
<i>temperature</i>	float	Temperatura(°C)
<i>rel_humidity</i>	float	Humedad relativa
<i>precipitation</i>	float	Precipitaciones (mm)

Tabla 4.5: Tabla weather_data

Esta tabla contiene los datos meteorológicos de cada una de las fechas contenidas en intervalos de 3 horas dentro del conjunto de entrenamiento.

Nota: En meteorología, es importante tener en cuenta que la dirección nos indica de dónde viene el viento, no hacia dónde va. Se mide en grados, desde 0° (excluido) hasta 360° (incluido), girando en el sentido de las agujas del reloj en el plano horizontal visto desde arriba. Valores cercanos a 1° y 360° indican viento del norte, cercanos a 90° viento del este, 180° del sur y 270° del oeste. Entre estos valores tendremos el resto de direcciones: nordeste, sureste, suroeste y noroeste.

4.2.2.6 Tabla *travel_time_intersection_to_tollgate*

(“trajectories_table5_training_20min_avg_travel_time.csv”)

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

Campo	Tipo	Descripción
<i>intersection_id</i>	string (char(1))	Identificador de la intersección
<i>tollgate_id</i>	string (char(1))	Identificador de la barrera de peaje
<i>time_window</i>	string(varchar(43))	Ventana de tiempo de 20 minutos
<i>avg_travel_time</i>	float	Tiempo medio de viaje (segundos)

Tabla 4.6: Tabla *travel_time_intersection_to_tollgate*

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* con el objetivo de agrupar datos. Para construir esta tabla, el script crea un diccionario para guardar el tiempo de viaje para cada una de las rutas por cada ventana de tiempo y calcula la media del tiempo de viaje para cada una de esas rutas por cada ventana de tiempo. Por lo tanto, a partir de la tabla *vehicle_trajectories_training* se almacena en la tabla *travel_time_intersection_to_tollgate* el tiempo medio de viaje para cada una de las rutas y ventana de tiempo de 20 minutos en el conjunto de entrenamiento.

4.2.2.7 Tabla *traffic_volume_tollgates* (“volume_table6_training_20min_avg_volume.csv”)

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

siguiente:

Campo	Tipo	Descripción
<i>tollgate_id</i>	string (char(1))	Identificador de la intersección
<i>time_window</i>	string (varchar(45))	Ventana de tiempo de 20 minutos
<i>direction</i>	string(char(1))	Dirección en la que se atraviesa la barrera de peaje
<i>volume</i>	int	Número de vehículos que atraviesan la barrera de peaje en la ventana de tiempo de 20 minutos

Tabla 4.7: Tabla traffic_volume_tollgates

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* con el objetivo de agrupar datos. Para construir esta tabla, el script crea un diccionario para guardar el volumen de vehículos y la dirección en la que atraviesan cada una de las barreras de peaje cada ventana de 20 minutos. Por lo tanto, a partir de la tabla *traffic_volume_tollgates_training* se almacena en la tabla *traffic_volume_tollgates* el volumen de tráfico para cada una de los pares **barrera de peaje-dirección** en cada ventana de tiempo de 20 minutos en el conjunto de entrenamiento.

4.2.3 Base de datos con los datos modificados

4.2.3.1 Tabla *road_links_modified* (“*links_table3.csv*”)

El nuevo esquema de la tabla es el siguiente:

Campo	Tipo	Descripción
<i>link_id</i>	smallint	Identificador del enlace
<i>length</i>	float	Longitud del enlace en metros
<i>width</i>	float	Anchura del enlace en metros
<i>lanes</i>	int	Número de carriles
<i>in_top</i>	smallint[]	Este atributo contiene los enlaces entrantes al enlace actual
<i>out_top</i>	smallint[]	Este atributo contiene los enlaces salientes del enlace actual
<i>lane_width</i>	float	Anchura de cada uno de los carriles del enlace en metros

Tabla 4.8: Tabla *road_links_modified*

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. Para poder cargar desde el archivo *.csv* la tabla y que no surgiera conflicto de tipos, inicialmente se crearon las columnas con el tipo **varchar**, se generaron arrays a partir de los valores de esas columnas y se alteró la tabla para realizar una conversión explícita a arrays de tipo **smallint**.

4.2.3.2 Tabla *vehicle_routes_modified* (“*routes_table4.csv*”)

El nuevo esquema de la tabla es el siguiente:

Campo	Tipo	Descripción
<i>intersection_id</i>	char(1)	Identificador de la intersección
<i>tollgate_id</i>	smallint	Identificador de la barrera de peaje
<i>link_seq</i>	smallint[]	Secuencia de enlaces que conforman la ruta desde la intersección hasta la barrera de peaje

Tabla 4.9: Tabla *vehicle_routes_modified*

En esta tabla se cambió el tipo de la columna *tollgate_id* a **smallint** y la columna *link_seq* a **smallint[]**. Este último atributo se modificó de la misma forma que las columnas *in_top* y *out_top* de la tabla anterior.

4.2.3.3 Tabla *vehicle_trajectories_training_modified* (“*trajectories_table 5_training.csv*”)

El nuevo esquema de la tabla es el siguiente:

Campo	Tipo	Descripción
<i>intersection_id</i>	char(1)	Identificador de la intersección
<i>tollgate_id</i>	smallint	Identificador de la barrera de peaje
<i>vehicle_id</i>	int	Identificador del vehículo
<i>starting_time</i>	timestamp	Momento del tiempo en el que el vehículo entra en la ruta
<i>travel_seq</i>	link_object[]	Trayectoria de la ruta formada por un conjunto de enlaces. Estos enlaces están representados con un objeto link_object, que contiene su identificador, el momento del tiempo en el que el vehículo entra en ese enlace y el tiempo que pasa el vehículo atravesando dicho enlace en segundos.
<i>travel_time</i>	float	Tiempo total que tarda el vehículo en viajar desde la intersección hasta la barrera de peaje.

Tabla 4.10: Tabla vehicle_trajectories_training_modified

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**).

Para convertir los valores de la columna *travel_seq* a un conjunto de objetos de tipo **link_object**[], primero se convirtieron en arrays de **varchar** y se cambió el tipo de la columna a este tipo de arrays. A continuación, se recorrió cada una de las filas de la tabla, de tal forma que, por cada fila, se cogió el **array de varchar**, se creó un objeto **link_object** por cada uno de los elementos del array y se guardaron estos objetos en un **link_object**{}. Así, se actualizó la tabla con este nuevo tipo y, para que la columna tenga el tipo correcto, se realizó la conversión explícita a **link_object**{}.

4.2.3.4 Tabla *traffic_volume_tollgates_training_modified* (“*volume_table6_training.csv*”)

El nuevo esquema de la tabla es el siguiente:

Campo	Tipo	Descripción
<i>time</i>	timestamp	Momento en el que un vehículo atraviesa la barrera de peaje
<i>tollgate_id</i>	smallint	Identificador de la barrera de peaje
<i>direction</i>	smallint	Dirección en la que el vehículo atraviesa la barrera de peaje. Si es 0, la dirección es de salida; si es 1, la dirección es de entrada.

Tabla 4.11: Tabla *traffic_volume_tollgates_training_modified*

<i>has_etc</i>	boolean	Indica si el vehículo utiliza un ETC (Electronic Toll Collection)
----------------	---------	---

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* (“*weather (table 7)_training.csv*”)

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 día 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* (“*trajectories_table5_training_20min_avg_travel_time.csv*”)

El nuevo esquema de la tabla es el siguiente:

Campo	Tipo	Descripción
<i>intersection_id</i>	char(1)	Identificador de la intersección
<i>tollgate_id</i>	smallint	Identificador de la barrera de peaje
<i>time_window</i>	Timestamp ARRAY[2]	Ventana de tiempo de 20 minutos
<i>avg_travel_time</i>	float	Tiempo medio de viaje (segundos)

Tabla 4.12: Tabla *travel_time_intersection_to_tollgate_modified*

En esta tabla se modificaron los tipos de las columnas *tollgate_id* (a **smallint**) y *time_window* (a **timestamp ARRAY[2]**). El proceso que se llevó a cabo para convertir esta última columna al nuevo tipo es similar al que se llevó a cabo para el tipo **link_object[]** en la tabla *vehicle_trajectories_training_modified*. Para representar el intervalo en la columna *time_window* se ha utilizado un array de dos posiciones por conveniencia.

4.2.3.7 Tabla *traffic_volume_tollgates_modified* (“*volume_table_6_training_20min_avg_volume.csv*”)

El nuevo esquema de la tabla es el siguiente:

Campo	Tipo	Descripción
<i>tollgate_id</i>	smallint	Identificador de la intersección
<i>time_window</i>	timestamp ARRAY[2]	Ventana de tiempo de 20 minutos
<i>direction</i>	smallint	Dirección en la que se atraviesa la barrera de peaje
<i>volume</i>	int	Número de vehículos que atraviesan la barrera de peaje en la ventana de tiempo de 20 minutos
<i>proportion_h asetc_vehicles</i>	Int	Proporción de coches que han pasado en la ventana de tiempo de 20 minutos que y que tienen ETC (<i>Electronic Toll Collection</i>)

Tabla 4.13: Tabla traffic_volume_tollgates_modified

En esta tabla se modificaron los tipos de las variables *tollgate_id* (a **smallint**), *time_window* (a **timestamp ARRAY[2]**) y *direction* (a **smallint**). Para realizar el cambio del tipo del atributo **time_window** se realizó el mismo proceso que la columna *time_window* en la tabla anterior. Además, en el script proporcionado por la competición que construye esta tabla (**aggregate_volume.py**) se alteró para que en esta tabla apareciera la proporción de coches que pasan por la ventana de 20 minutos y que utilizan el dispositivo ETC. Este atributo es importante tenerlo en cuenta puesto que los coches pasan más rápido por la barrera de peaje si no tienen que pararse a pagar en la barrera de peaje puesto que con este dispositivo se cobra al vehículo automáticamente al atravesar la misma.

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 primera fase de pruebas.

4.2.4.1 Tabla *travel_time_intersection_to_tollgate_test1* (“test1_20min_avg_travel_time.csv”)

Campo	Tipo	Descripción
<i>intersection_id</i>	char(1)	Identificador de la intersección
<i>tollgate_id</i>	smallint	Identificador de la barrera de peaje
<i>time_window</i>	timestamp ARRAY[2]	Ventana de tiempo de 20 minutos
<i>avg_travel_time</i>	float	Tiempo medio de viaje (segundos)

Tabla 4.14: Tabla *travel_time_intersection_to_tollgate_test1*

Para comprender el sentido de esta relación, es necesario observar la **Figura 4.3**. En esta tabla se proporciona el *tiempo promedio de viaje* de cada una de las rutas en los intervalos de tiempo previos (color verde) a los intervalos de tiempo a predecir. Particularmente, 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 de Octubre de 2016** hasta el día **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_test1* (“test1_20min_avg_volume.csv”)

El esquema de la tabla es el siguiente:

Campo	Tipo	Descripción
<i>tollgate_id</i>	smallint	Identificador de la intersección
<i>time_window</i>	timestamp ARRAY[2]	Ventana de tiempo de 20 minutos
<i>direction</i>	smallint	Dirección en la que se atraviesa la barrera de peaje
<i>volume</i>	int	Número de vehículos que atraviesan la barrera de peaje en la ventana de tiempo de 20 minutos
<i>proportion_hasetc _vehicles</i>	int	Proporción de coches que han pasado en la ventana de tiempo de 20 minutos que tienen ETC (Electronic Toll Collection)

Tabla 4.15: Tabla *traffic_volume_tollgates_test1*

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 los intervalos de tiempo previos (color verde) a los intervalos de tiempo a predecir. Particularmente, se especifica el volumen de tráfico 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 el siguiente:

Campo	Tipo	Descripción
<i>intersection_id</i>	char(1)	Identificador de la intersección
<i>tollgate_id</i>	smallint	Identificador de la barrera de peaje
<i>time_window</i>	timestamp ARRAY[2]	Ventana de tiempo de 20 minutos
<i>avg_travel_time</i>	float	Tiempo medio de viaje (segundos)

Tabla 4.16: Tabla *tabla_resultado_average_travel_time*

Esta tabla contiene los intervalos que nos insta la competición a predecir en la *fase de testeo*. Específicamente, en esta tabla se guardan las predicciones realizadas con respecto al tiempo promedio de viaje en los intervalos a predecir (**Figura 4.3**).

4.2.4.4 Tabla *tabla_resultado_traffic_volume*

El esquema de la tabla es el siguiente:

Campo	Tipo	Descripción
<i>tollgate_id</i>	smallint	Identificador de la intersección
<i>time_window</i>	timestamp ARRAY[2]	Ventana de tiempo de 20 minutos
<i>direction</i>	smallint	Dirección en la que se atraviesa la barrera de peaje
<i>volume</i>	int	Número de vehículos que atraviesan la barrera de peaje en la ventana de tiempo de 20 minutos

Tabla 4.17: Tabla *tabla_resultado_traffic_volume*

Esta tabla contiene los intervalos que nos insta la competición a predecir en la fase de testeo. Específicamente, en esta tabla se guardan las predicciones realizadas con respecto al volumen de tráfico en los intervalos a predecir (**Figura 4.3**).

4.3 Creación de gráficas para visualizar los datos almacenados

¿Poner las gráficas en los anexos?

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

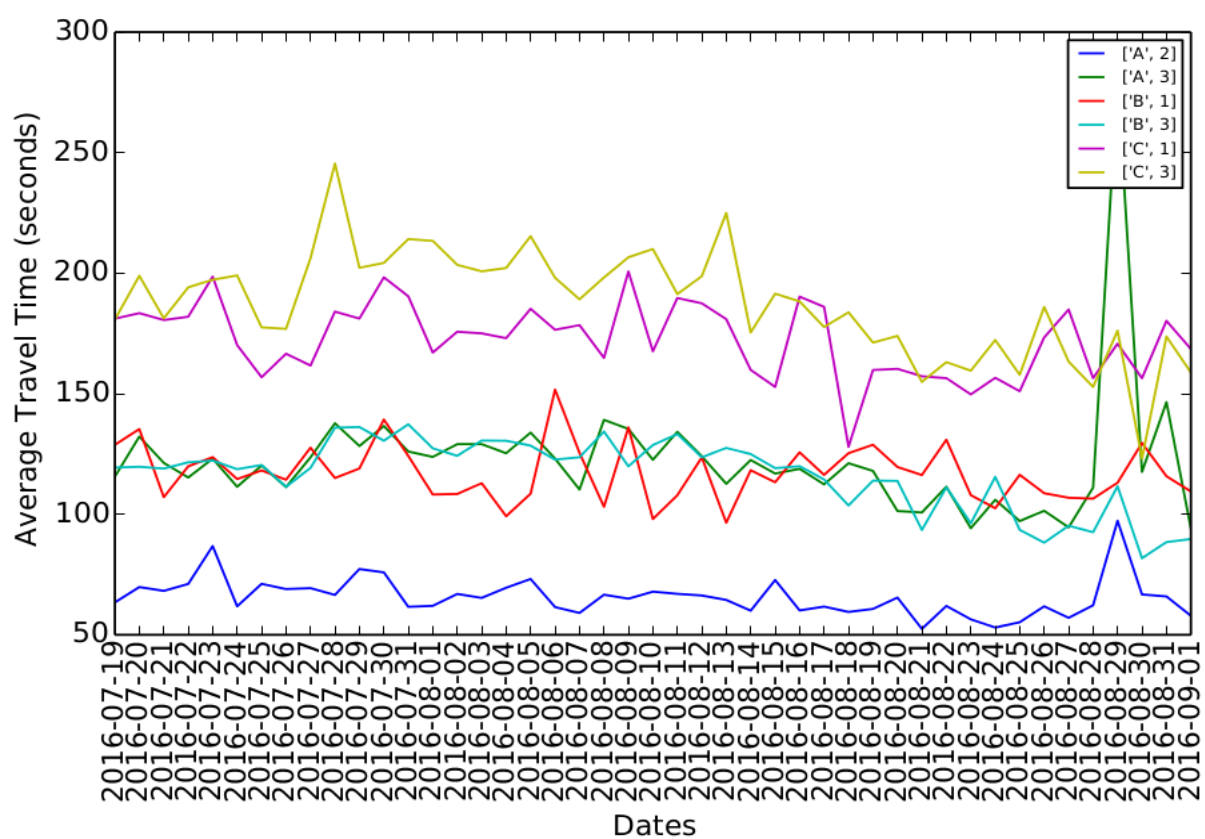
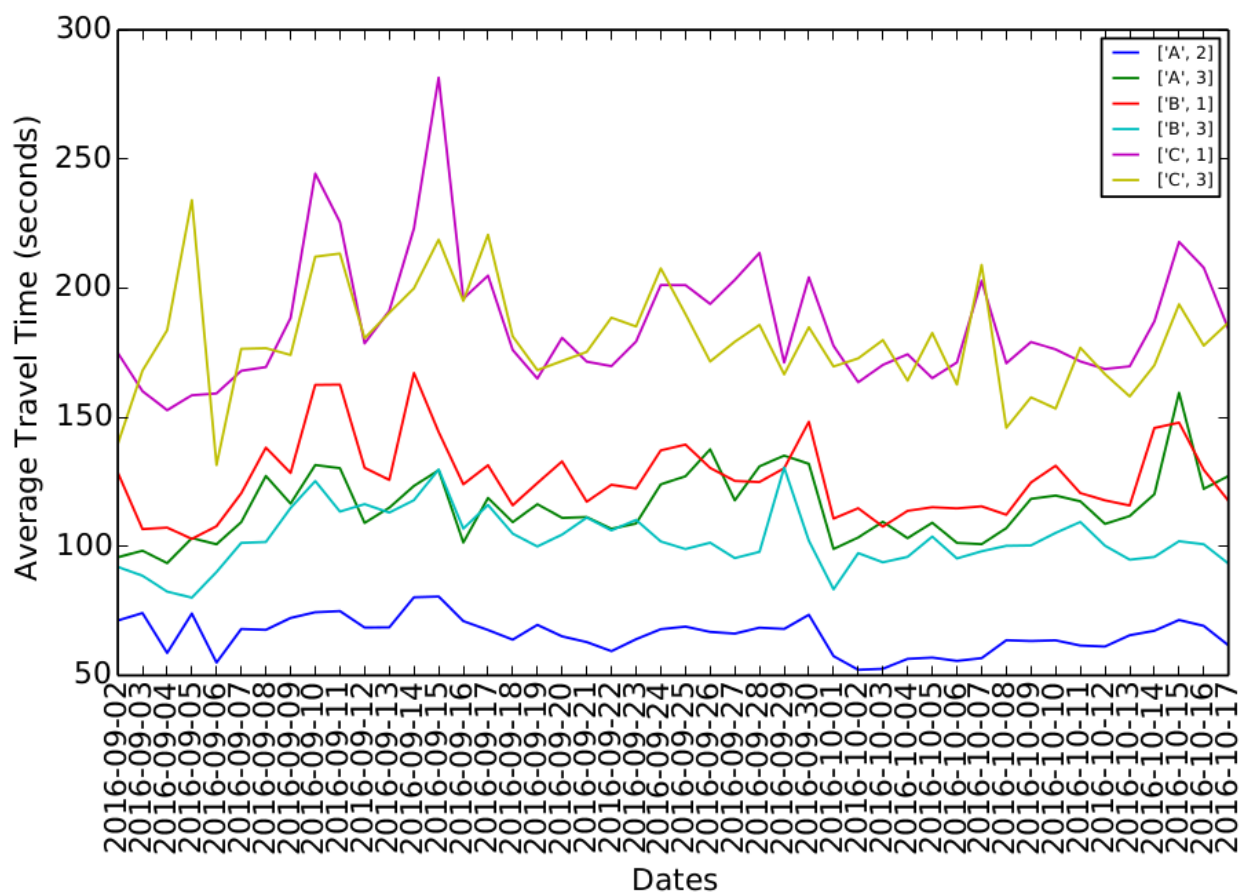


Figura 4.6: Tiempo promedio de viaje medio en cada uno de los días en las diferentes rutas



En estas gráficas se representa el valor medio de los tiempos promedios de viaje de todos los intervalos de tiempo de 20 minutos 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 y anchura de las carreteras que las forman (Figura 4.5).

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

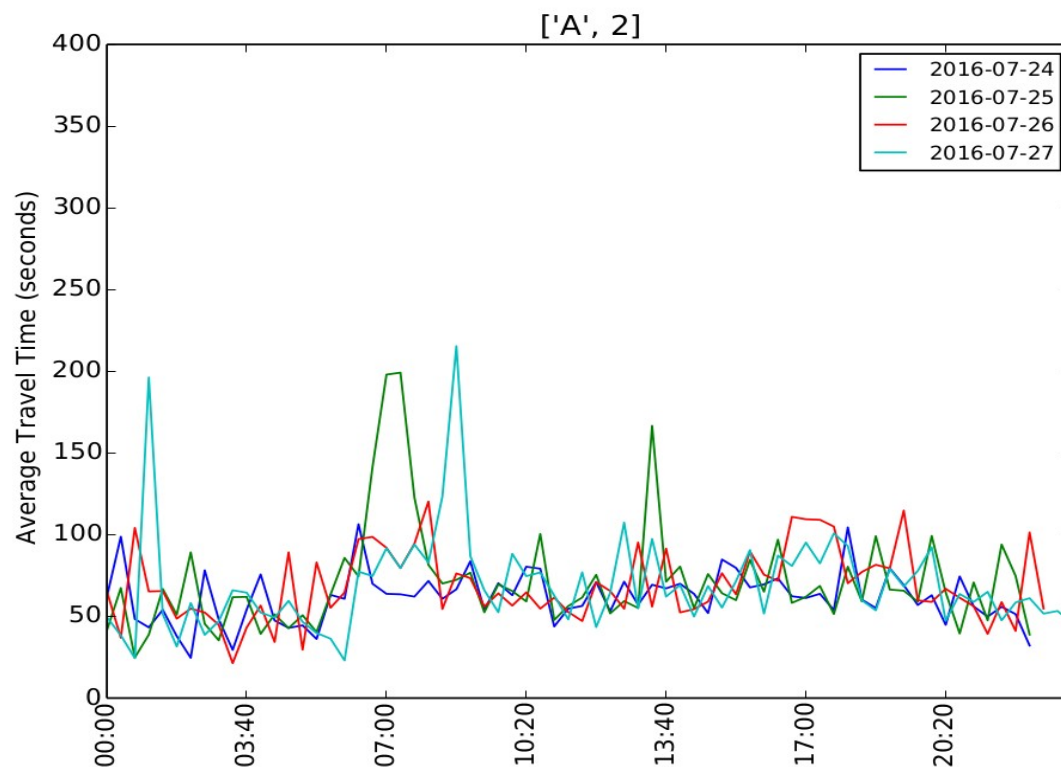


Figura 4.8: Tiempo promedio de viaje por horas en algunos días en la ruta A-2

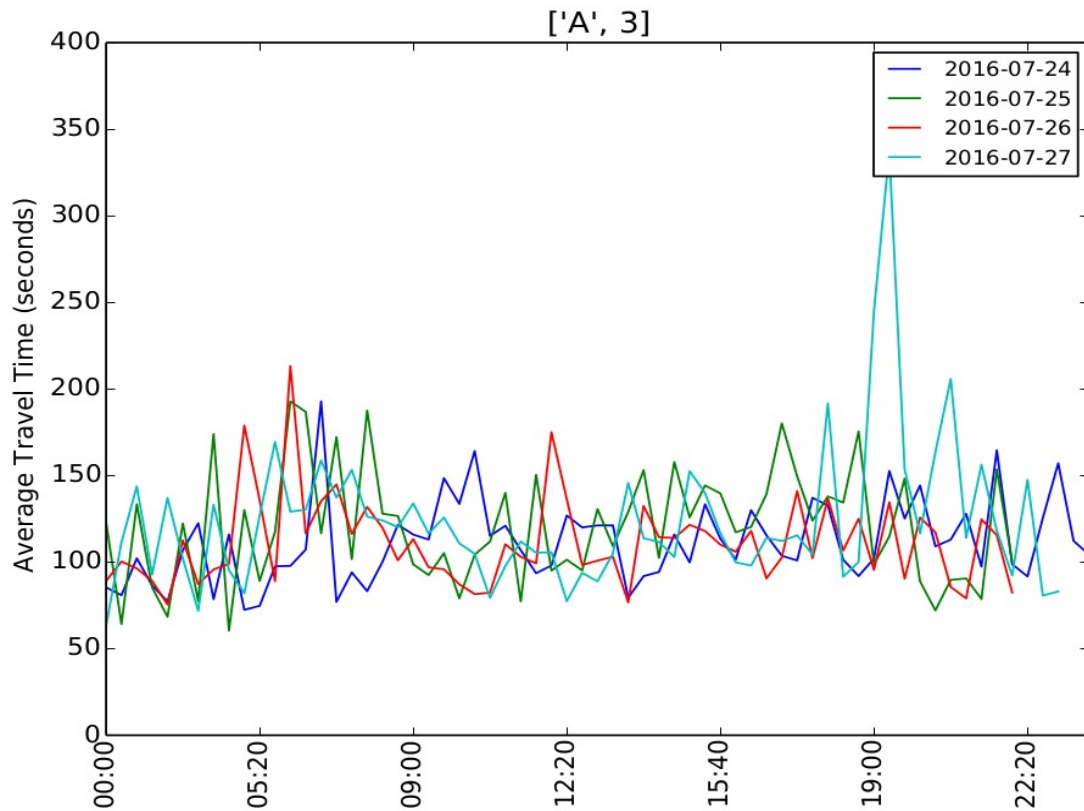


Figura 4.10: Tiempo promedio de viaje por horas en algunos días en la ruta A-3

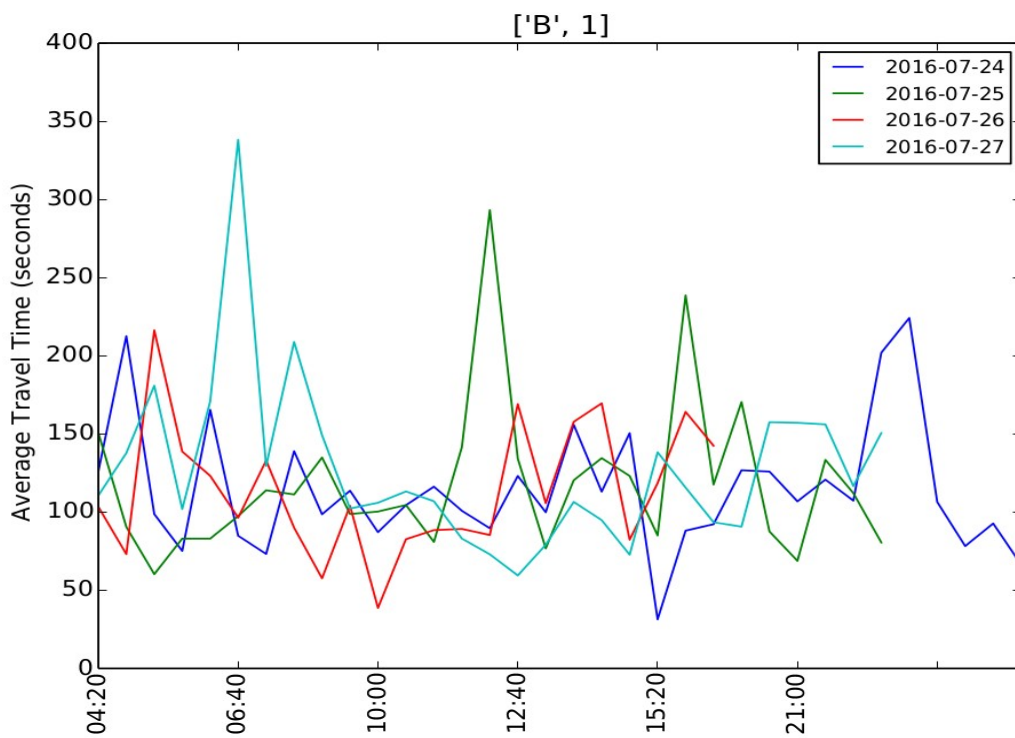


Figura 4.9 Tiempo promedio de viaje por horas en algunos días en la ruta B-1

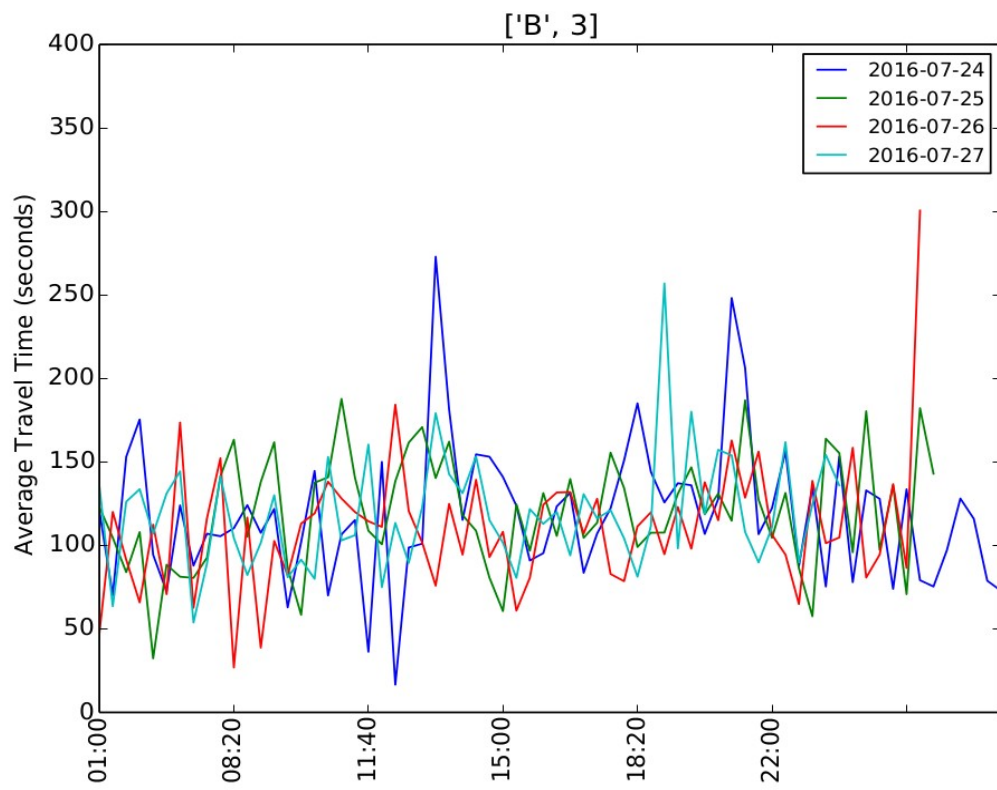


Figura 4.11: Tiempo promedio de viaje por horas en algunos días en la ruta

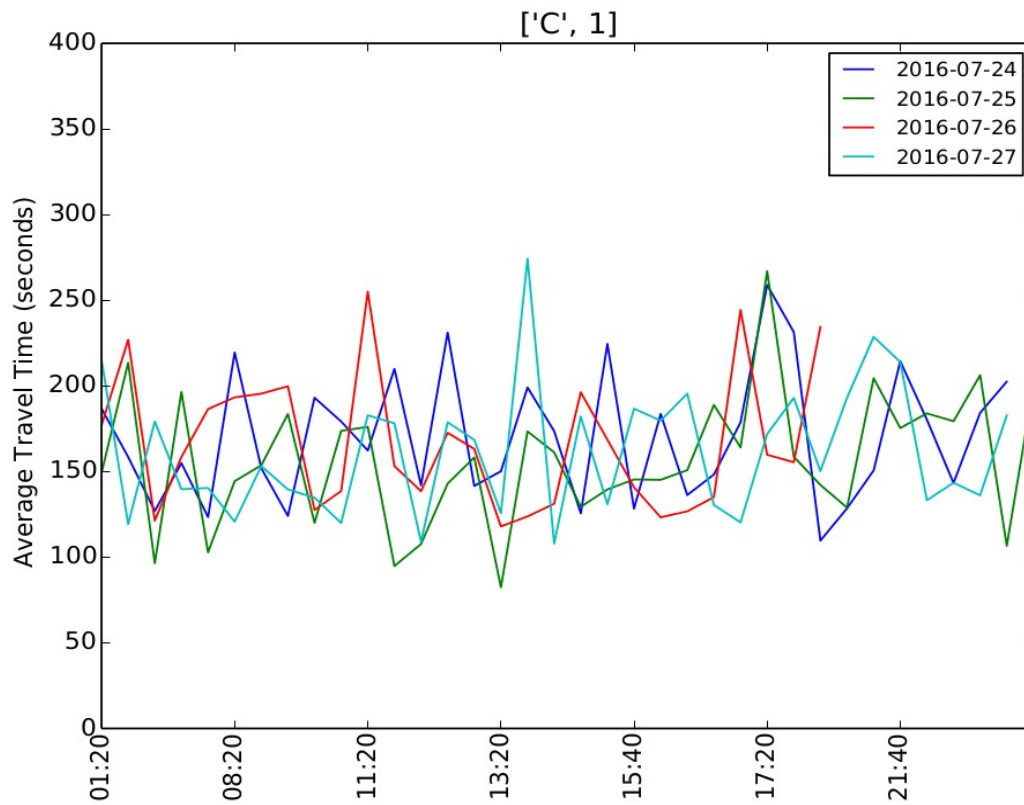


Figura 4.12: Tiempo promedio de viaje por horas en algunos días en la ruta

C-1

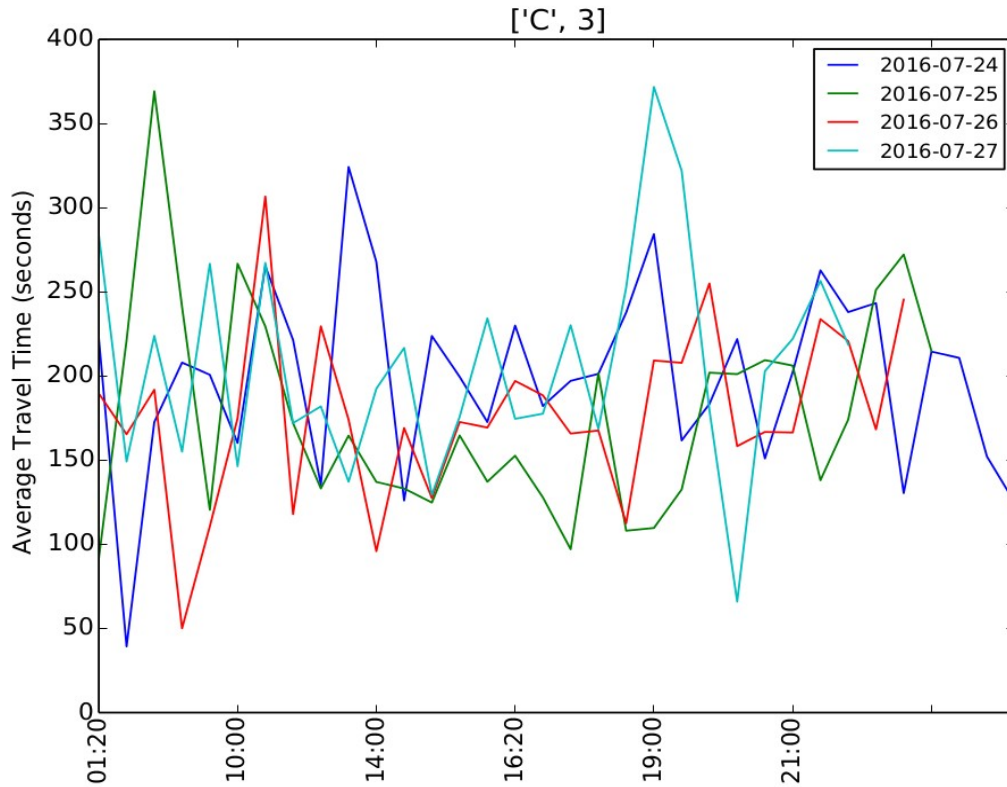


Figura 4.13: Tiempo promedio de viaje por horas en algunos días en la ruta

C-3

En estas imágenes se grafica el tiempo promedio de viaje por horas de algunos días en todas las 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 intervalos de tiempo de 20 minutos, las series tienen una serie de diferencias que dificultan el descubrimiento de un comportamiento totalmente estacional en las mismas.

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 (según la **Figura 4.3**). 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

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 ha creado una nueva estructura de los datos para pasarla como entrada a los algoritmos a utilizar para estimar valores. Dicha estructura se representa en la siguiente tabla:

Campo	Tipo	Descripción
<i>type_day</i>	date	Tipo de día de la semana (laborable (1) o fin de semana (0))
<i>twenty_min_previous</i>	float	Tiempo medio de viaje 20 minutos antes de la ventana de tiempo (segundos)
<i>forty_min_previous</i>	float	Tiempo medio de viaje 40 minutos antes de la ventana de tiempo (segundos)
<i>sixty_min_previous</i>	float	Tiempo medio de viaje 60 minutos antes de la ventana de tiempo (segundos)
<i>eighty_min_previous</i>	float	Tiempo medio de viaje 80 minutos antes de la ventana de tiempo (segundos)
<i>onehundred_min_previous</i>	float	Tiempo medio de viaje 100 minutos antes de la ventana de tiempo (segundos)
<i>onehundredtwenty_min_previous</i>	float	Tiempo medio de viaje 120 minutos antes de la ventana de tiempo (segundos)
<i>pressure</i>	float	Presión del aire (hPa)

<i>sea_pressure</i>	float	Presión del nivel del mar (hPa)
<i>wind_direction</i>	float	Dirección del viento (°)
<i>wind_speed</i>	float	Velocidad del viento (m/s)
<i>temperature</i>	float	Temperatura(°C)
<i>rel_humidity</i>	float	Humedad relativa
<i>precipitation</i>	float	Precipitaciones (mm)
<i>avg_travel_time</i>	float	Tiempo medio de viaje (segundos)

Tabla 4.18: Vista creada sobre los datos a utilizar para entrenamiento y testeo de la primera aproximación de predicciones

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 4.14: 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 4.15: 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, ya que, como los pares *ruta-ventana de tiempo* están ordenados por **intersección**, **barrera de peaje** y **ventana de tiempo** tras realizar la consulta previa y las ventanas de tiempo se diferencian 20 minutos (ya que la competición proporciona los datos en intervalos de 20 minutos).

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 4.16: 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 4.17: 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 adquiriría 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 tenga 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 4.18: 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, 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 4.19: 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.

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 4.20: 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 creada en la **Figura 4.20**. 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 la primera aproximación de predicciones del tiempo promedio de viaje 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 **Tabla 4.18** para los datos de prueba puesto que son imprescindibles para poder proporcionar los mismos atributos en la fase de prueba con el objetivo de que prediga los valores del *tiempo promedio de viaje* de las ventanas de tiempo a predecir.

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

- A continuación se accede a la base de datos *tfgtest1* para obtener la vista que contiene los datos de prueba correspondientes a la ruta y al intervalo que estamos considerando. Es decir, obtenemos 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 4.22: 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 4.23: 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 4.1**. 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 4.24: 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 4.25: 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 4.26: 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 que se acumula el error correspondiente a los intervalos de una ruta a predecir, se acumula, a su vez, el *error relacionado con cada una de las rutas de la competición* (primer sumatorio). Esta acumulación va sumando los errores del *segundo sumatorio* :

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

Figura 4.27: Acumulación de los errores del segundo sumatorio de la fórmula del error MAPE

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

Figura 4.28: 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 (con el script *aggregate_travel_time.py*) 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.

¿Meter la tabla con las predicciones de cada algoritmo para cada ruta e intervalo junto con los valores reales y los resultados?

Comentar el por qué de los resultados.

4.4.2 Segunda aproximación

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 y días 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 modelos *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 4.29: 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. Por ejemplo, si de la ruta *A-2* en el intervalo *13:00-13:20* en un determinado día no tenemos el valor del *tiempo promedio de viaje*, entonces hallamos el valor medio de los días restantes en esta misma ruta e intervalo y lo establecemos como valor del *tiempo promedio de viaje* de esa ruta e intervalo.

```

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

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 de 20 minutos **17:00-19:00**. Para llevar a cabo esto, se llevan a cabo 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='tfctest1' user='javisunani' 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 4.31: 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 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 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(dispatch=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%$ MSE=%.3f' % (best_cfg, best_score))

```

Figura 4.32: 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 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. Las mejores predicciones obtenidas fueron las siguientes:

¿Introducir la tabla con las distintas predicciones?

4.5 Predicciones del volumen de tráfico

Inconvenientes del proyecto

- Se han proporcionado pocos datos de entrenamiento.
- Al agrupar datos, faltan datos de intervalos.

Capítulo 5

Conclusiones y líneas futuras

Este capítulo es obligatorio. Toda memoria de Trabajo de Fin de Grado debe incluir unas conclusiones y unas líneas de trabajo futuro

Capítulo 6

Summary and Conclusions

This chapter is compulsory. The memory should include an extended summary and conclusions in english.

Capítulo 7

Presupuesto

Este capítulo es obligatorio. Toda memoria de Trabajo de Fin de Grado debe incluir un presupuesto.

7.1 Sección Uno

Capítulo 8

Título del Apéndice 1

8.1 Algoritmo XXX

```
/******  
*  
* Fichero .h  
*  
*****  
*  
* AUTORES  
*  
*  
* FECHA  
*  
*  
* DESCRIPCION  
*  
*  
*****/
```

8.2 Algoritmo YYY

```
/******  
*  
* Fichero .h  
*  
*****  
*  
* AUTORES
```

*
* FECHA
*
* DESCRIPCION
*
*
*****/

Capítulo 9

Título del Apéndice 2

9.1 Otro apéndice: Sección 1

texto

9.2 Otro apéndice: Sección 2

texto

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. 419-431, 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=1590qIrBgZvJjANFsTvHYZILkWRz9vfFN>
- [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] 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
- [7] Y. Huang, “Highway Tollgates Traffic Flow Prediction Task 1. Travel Time Prediction”, presented at the KDD 2017, Halifax, Nova Scotia – Canada, 2017. [Online]. Disponible en: https://drive.google.com/open?id=1GD1ZIpWDq7qMM7bvpTSTnIWqSnpDlS_t
- [8] H. Cai, R. Zhong, C.Wang et al., “KDD CUP 2017 Travel Time Prediction”, presented at the KDD 2017, Halifax, Nova Scotia – Canada, 2017. [Online]. Disponible en: <https://drive.google.com/open?id=1ew6oLOPHGoz8IMIt5g6XkZn67ADDtxJJ>
- [9] 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=1uK8IwRTe061NVbWQn4FQFxFH7BUx6b6mq>
- [10] J. Zhou, Y. Guo, Y. Chen, J. Lin and H. Lin, “Learning and Prediction over Light-Weight Spatio-Temporal Data”, presented at the KDD 2017, Halifax, Nova Scotia – Canada, 2017. [Online]. Disponible en: https://drive.google.com/open?di=1Jbz0GNxYcjCghp0cfJia0omYKu1y5_aR
- [11] S. Luo, “KDD CUP 2017: Volume Prediction Task Solution”, presented at the KDD 2017, Halifax, Nova Scotia – Canada, 2017. [Online]. Disponible en: <https://drive.google.com/open?id=1-krIjoSatfoPNnzYRSOZKfxIKK0RHT59>

<http://www.it.uc3m.es/jvillena/irc/practicas/06-07/22.pdf>

<https://blog.es.logicalis.com/analytics/mineria-de-datos-aplicaciones-que-ya-son-una-realidad>

<http://www.it.uc3m.es/jvillena/irc/practicas/10-11/15mem.pdf>

http://sedici.unlp.edu.ar/bitstream/handle/10915/35555/Documento_completo.pdf?sequence=1

<https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>

<https://blog.statsbot.co/time-series-prediction-using-recurrent-neural-networks-lstms-807fa6ca7f>

<http://www.uokufa.edu.iq/staff/ehsanali/Tan.pdf> → Introducción a la minería de datos

<https://books.google.es/books?>

[id=lpjcDgAAQBAJ&pg=PA228&lpg=PA228&dq=sparse+aware+missing+data&source=bl&ots=lZe_3Bk0x9&sig=zDrvLwe2SL_CjMbskFtWFfITYTs&hl=es&sa=X&ved=0ahUKEwiZhYn-](https://books.google.es/books?id=lpjcDgAAQBAJ&pg=PA228&lpg=PA228&dq=sparse+aware+missing+data&source=bl&ots=lZe_3Bk0x9&sig=zDrvLwe2SL_CjMbskFtWFfITYTs&hl=es&sa=X&ved=0ahUKEwiZhYn-1K3aAhWKPBQKHytICr0Q6AEIaTAI#v=onepage&q=sparse%20aware%20missing%20data&f=false)

[1K3aAhWKPBQKHytICr0Q6AEIaTAI#v=onepage&q=sparse%20aware%20missing%20data&f=false](https://books.google.es/books?id=lpjcDgAAQBAJ&pg=PA228&lpg=PA228&dq=sparse+aware+missing+data&source=bl&ots=lZe_3Bk0x9&sig=zDrvLwe2SL_CjMbskFtWFfITYTs&hl=es&sa=X&ved=0ahUKEwiZhYn-1K3aAhWKPBQKHytICr0Q6AEIaTAI#v=onepage&q=sparse%20aware%20missing%20data&f=false)

<https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>

- Libro de introducción a la minería de datos

<http://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html>

