



Universidad
de La Laguna

Escuela Superior de
Ingeniería y Tecnología

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, 14 de mayo de 2018

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

C E R T I F I C A

Que la presente memoria titulada:

“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 14 de mayo de 2018

Agradecimientos

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

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

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

Al resto de mi familia 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.....	3
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.....	14
2.2.2.1 RapidMiner.....	14
2.2.2.2 Lenguaje R.....	15
2.2.2.3 Weka.....	15
2.2.3 Otros tipos de software empleados.....	16

2.2.3.1 Github.....	16
2.2.3.2 Gedit.....	17
Capítulo 3 La minería de datos.....	18
3.1 Introducción.....	18
3.2 Técnicas de minería de datos.....	21
3.2.1 XGBOOST.....	21
3.2.1.1 Definición.....	21
3.2.1.2 Gradient boosting.....	22
3.2.1.3 Implementación del algoritmo.....	25
3.2.2 LightGBM.....	26
3.2.2.1 Definición.....	26
3.2.2.2 Ventajas.....	27
3.2.3 Perceptrón multicapa (Redes neuronales).....	28
3.2.3.1 Definición.....	28
3.2.3.2 Neuronas.....	29
3.2.3.3 Redes de neuronas.....	30
3.2.3.4 Entrenamiento de una red neuronal.....	31
3.2.4 Modelo ARIMA.....	33
3.2.4.1 Definición de una serie temporal.....	33
3.2.4.2 Características de las series temporales.....	34
3.2.4.3 Componentes de las series temporales.....	39
3.2.4.4 Tipos de esquemas para series temporales.....	40
3.2.4.5 Clasificación descriptiva de las series temporales.....	42
3.2.4.6 Funciones de autocorrelación y autocorrelación parcial.....	47
3.2.4.7 Estimación de las componentes de una serie temporal.....	48
3.2.4.8 Definición del modelo ARIMA.....	62
3.2.4.9 Modelos autorregresivos AR(p).....	63
3.2.4.10 Modelo de medias móviles MA(q).....	64

3.2.4.11 Modelos ARMA(p,q).....	64
3.2.4.12 Fases del modelo ARIMA.....	65
3.2.4.13 Identificación práctica del modelo ARIMA.....	68
3.2.5 k-Vecinos Más Cercanos.....	69
3.2.5.1 Definición.....	69
3.2.5.2 El algoritmo K-NN básico.....	69
3.2.5.3 Método de regresión basado en KNN.....	72
Capítulo 4 Fases del proyecto.....	73
4.1 Problema planteado por la competición KDDCup 2017.....	74
4.1.1 Contexto.....	74
4.1.2 Tareas.....	75
4.1.3 Etapas y reglas de la competición.....	76
4.1.4 Métricas de evaluación.....	80
4.2 Creación de bases de datos para almacenar los datos de la competición y modificaciones.....	81
4.2.1 Estructura de bases de datos propuesta.....	82
4.2.2 Base de datos con los datos originales.....	82
4.2.2.1 Tabla vehicle_routes.....	82
4.2.2.2 Tabla road_links.....	82
4.2.2.3 Tabla vehicle_trajectories_training.....	83
4.2.2.4 Tabla traffic_volume_tollgates_training.....	84
4.2.2.5 Tabla weather_data.....	84
4.2.2.6 Tabla travel_time_intersection_to_tollgate.....	84
4.2.2.7 Tabla traffic_volume_tollgates.....	85
4.2.3 Base de datos con los datos modificados.....	85
4.2.3.1 Tabla road_links_modified.....	85
4.2.3.2 Tabla vehicle_routes_modified.....	85
4.2.3.3 Tabla vehicle_trajectories_training_modified.....	86
4.2.3.4 Tabla traffic_volume_tollgates_training_modified.....	86

4.2.3.5 Tabla weather_data_modified.....	86
4.2.3.6 Tabla travel_time_intersection_to_tollgate_modified.....	87
4.2.3.7 Tabla traffic_volume_tollgates_modified.....	87
4.2.4 Base de datos con los datos relacionados con la fase de pruebas....	87
4.2.4.1 Tabla travel_time_intersection_to_tollgate_test.....	88
4.2.4.2 Tabla traffic_volume_tollgates_test.....	88
4.2.4.3 Tabla tabla_resultado_average_travel_time.....	88
4.2.4.4 Tabla tabla_resultado_traffic_volume.....	88
4.2.5 Base de datos con los datos reales de los valores a predecirse.....	89
4.2.5.1 Tabla travel_time_intersection_to_tollgate_real.....	89
4.2.5.2 Tabla traffic_volume_tollgates_real.....	89
4.3 Creación de gráficas para visualizar los datos almacenados.....	89
4.3.1 Gráficas del tiempo promedio de viaje de todos los días por rutas.	90
4.3.2 Gráficas del tiempo promedio de viaje de algunos días en todas las horas en cada una de las rutas.....	92
4.3.3 Gráficas del volumen de tráfico de todos los días en todas las horas en cada una de los pares barrera de peaje-dirección.....	102
4.4 Predicciones del tiempo promedio de viaje.....	106
4.4.1 Primera aproximación.....	106
4.4.1.1 Creación de las vistas minables.....	109
4.4.1.2 Realización de predicciones a partir de las vistas.....	118
4.4.2 Segunda aproximación.....	126
4.4.2.1 Preparación de los datos.....	126
4.4.2.2 Realización de estimaciones.....	128
4.4.3 Tercera aproximación.....	130
4.4.3.1 Preparación de los datos.....	130
4.5 Predicciones del volumen de tráfico.....	132
4.5.1 Primera aproximación.....	133
4.5.2 Segunda aproximación.....	136

4.5.3 Tercera aproximación.....	139
Capítulo 5 Conclusiones y líneas futuras.....	141
5.1.1 Conclusiones.....	141
5.1.2 Líneas futuras.....	141
Capítulo 6 Summary and Conclusions.....	143
6.1.1 Conclusions.....	143
6.1.2 Future lines.....	143
Capítulo 7 Presupuesto.....	145
Capítulo 8 Anexo.....	147
8.1.1 Tablas con los datos de la competición.....	147

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 2.7: Interfaz gráfica de Github.....	17
Figura 3.1: El proceso de descubrimiento de conocimiento en bases de datos (KDD).....	18
Figura 3.2: Cuatro de las tareas principales de minería de datos.....	20
Figura 3.3: Crecimiento level-wise del árbol en XGBoost.....	27
Figura 3.4: Crecimiento leaf-wise del árbol en LightGBM.....	27
Figura 3.5: Estructura de una neurona artificial.....	29
Figura 3.6: Estructura de una red neuronal.....	30
Figura 3.7: Ejemplo de serie temporal.....	33
Figura 3.8: Ejemplo de una serie temporal con tendencia.....	34
Figura 3.9: Ejemplo de una serie temporal con ciclos estacionales.....	35
Figura 3.10: Serie temporal con pulsos.....	36
Figura 3.11: Comparación de una serie estacionaria con una no estacionaria con respecto a la autocovarianza.....	35
Figura 3.12: Distintos tipos de valores atípicos.....	37

Figura 3.13: Serie temporal con tendencia agregada a la estacionalidad.....	40
Figura 3.14: Serie temporal con tendencia multiplicada por la estacionalidad.....	41
Figura 3.15: Ejemplo de serie estacionaria.....	42
Figura 3.16: Ejemplo de serie no estacionaria.....	42
Figura 3.17: Comparación de una serie estacionaria con una no estacionaria con respecto a la media.....	43
Figura 3.18: Comparación de una serie estacionaria con una no estacionaria con respecto a la varianza.....	43
Figura 3.19: Comparación de una serie estacionaria con una no estacionaria con respecto a la autocovarianza.....	44
Figura 3.20: Ejemplo de un gráfico de autocorrelación.....	45
Figura 3.21: Serie temporal y su tendencia.....	46
Figura 3.22: Componente regular de la serie temporal tras eliminar su tendencia.....	47
Figura 3.23: Número de accidentes de tráfico durante 10 meses.....	48
Figura 3.24: Representación de los datos de accidentes de tráfico.....	49
Figura 3.25: Primera media de la serie.....	49
Figura 3.26: Segunda media de la serie.....	49
Figura 3.27: Tercer media de la serie.....	50
Figura 3.28: Medias móviles de orden 3.....	50
Figura 3.29: Nueva serie de medias móviles.....	50
Figura 3.30: Ejemplo de estimación de la tendencia mediante medias móviles.....	51
Figura 3.31: Componente irregular tras eliminar la tendencia.....	51
Figura 3.32: Serie temporal antes de la diferenciación.....	52
Figura 3.33: Serie temporal después de la diferenciación.....	52
Figura 3.34: Serie temporales antes de la desestacionalización.....	53
Figura 3.35: Los últimos 6 años de la serie temporal de la Figura 3.34.....	54

Figura 3.36: Serie desestacionalizada.....	55
Figura 3.37: Fases del modelo ARIMA.....	60
Figura 3.38: Algoritmo básico KNN.....	65
Figura 3.39: Ejemplo de aplicación del algoritmo K-NN básico.....	65
Figura 4.1: Topología de la red de carreteras de la zona objetivo.....	71
Figura 4.2: Ventanas de tiempo para la predicción del tráfico.....	72
Figura 4.3: Cálculo del error de predicción del tiempo promedio de viaje ..	72
Figura 4.4: Cálculo del error de predicción del volumen de tráfico.....	73
Figura 4.5: Rutas de la competición KDDCup2017 con los enlaces que las forman.....	75
Figura 4.6: Tiempo promedio de viaje medio en cada uno de los días en las diferentes rutas.....	82
Figura 4.7: Continuación de la gráfica de la Figura 4.4.....	83
Figura 4.8: Tiempo promedio de viaje por horas en algunos días en la ruta A-2.....	84
Figura 4.9: Tiempo promedio de viaje por horas en algunos días en la ruta A-3.....	85
Figura 4.10 Tiempo promedio de viaje por horas en algunos días en la ruta B-1.....	85
Figura 4.11: Tiempo promedio de viaje por horas en algunos días en la ruta B-3.....	86
Figura 4.12: Tiempo promedio de viaje por horas en algunos días en la ruta C-1.....	86
Figura 4.13: Tiempo promedio de viaje por horas en algunos días en la ruta C-3.....	87
Figura 4.14: Barrera de peaje 1 en la dirección de entrada.....	88
Figura 4.15: Barrera de peaje 1 en la dirección de salida.....	89
Figura 4.16: Barrera de peaje 2 en la dirección de entrada.....	89
Figura 4.17: Barrera de peaje 3 en la dirección de entrada.....	90

Figura 4.18: Barrera de peaje 3 en la dirección de salida.....	90
Figura 4.19: Bloque principal de código que crea la evolución de las 2 horas previas a la ventana de tiempo correspondiente en los datos de entrenamiento.....	94
Figura 4.20: Obtención de las rutas de tráfico junto con las ventanas de tiempo comprendidas en los intervalos de tiempo a predecir.....	95
Figura 4.21: Estructura iterativa para crear las columnas relacionadas con la evolución del tráfico en las dos horas previas a la ventana de tiempo considerada.....	96
Figura 4.22: Consulta SQL que rellena la columna del tiempo promedio de viaje 20 minutos antes de la ventana de tiempo considerada.....	96
Figura 4.23: 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.....	97
Figura 4.24: Combinación de la tabla con los datos de entrenamiento del tiempo promedio de viaje junto con los datos meteorológicos.....	98
Figura 4.25: Código SQL que crea una vista para cada ruta e intervalo a partir de la vista con los datos combinados.....	98
Figura 4.26: Obtención de los datos de entrenamiento para la ruta e intervalo en consideración.....	100
Figura 4.27: Obtención de los datos de prueba para la ruta e intervalo en consideración.....	100
Figura 4.28: Entrenamiento y predicción del tiempo promedio de viaje en los distintos días de predicción con el modelo XGBoost.....	100
Figura 4.29: Cálculo del error medio de los días de predicción para una ruta e intervalo determinado.....	101
Figura 4.30: Acumulación de los errores correspondiente a los días a predecir de cada uno de los intervalos a estimar.....	101
Figura 4.31: Cálculo del segundo sumatorio de la fórmula del error MAPE para un algoritmo.....	101

Figura 4.32: Acumulación de los errores del segundo sumatorio de la fórmula del error MAPE.....	101
Figura 4.33: Cálculo del primer sumatorio de la fórmula del error MAPE para un algoritmo.....	102
Figura 4.34: Número de instancias de entrenamiento por intervalo de tiempo a predecir para la ruta A-2.....	103
Figura 4.35: Número de instancias de entrenamiento por intervalo de tiempo a predecir para la ruta A-2.....	104
Figura 4.36: Número de instancias de entrenamiento por intervalo de tiempo a predecir para la ruta B-1.....	104
Figura 4.37: Número de instancias de entrenamiento por intervalo de tiempo a predecir para la ruta B-3.....	105
Figura 4.38: Número de instancias de entrenamiento por intervalo de tiempo a predecir para la ruta C-1.....	105
Figura 4.39: Número de instancias de entrenamiento por intervalo de tiempo a predecir para la ruta C-3.....	106
Figura 4.40: Obtención de los valores reales del tiempo promedio de viaje para las rutas e intervalos de tiempo a predecir.....	107
Figura 4.41: Cálculo del valor del tiempo promedio de viaje de una parte de aquellas rutas e intervalos de las que no disponemos datos.....	108
Figura 4.42: 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.....	108
Figura 4.43: Cálculo del mejor modelo ARIMA para una ruta, día y ventana de tiempo a estimar.....	109
Figura 4.44: Concatenación de las dos horas previas a los intervalos de tiempo a predecir un día determinado con la serie temporal de una ruta.	110
Figura 4.45: Serie temporal a problema de aprendizaje supervisado.....	111
Figura 4.46: Estructura generada con el método de la ventana deslizante	111
Figura 4.47: Ejemplo de gráfica del volumen de tráfico en una ruta	

determinada en los días de entrenamiento.....	117
Figura 4.48: Eliminación del ruido de los datos proporcionados para la predicción del volumen de tráfico.....	118
Figura 4.49: Gráfica resultante de la eliminación de ruido.....	118

Índice de tablas

Tabla 3.1: Obtención de los parámetros (p, d, q) en los distintos modelos.	63
Tabla 3.2: Estructura de los datos de entrada para el algoritmo KNN.....	64
Tabla 4.1: Vista minable creada sobre los datos a utilizar para entrenamiento y testeo de la primera aproximación de predicciones del tiempo promedio de viaje.....	93
Tabla 4.2: Errores MAPE de la primera aproximación de predicciones del tiempo promedio de viaje.....	102
Tabla 4.3: Vista minable creada sobre los datos a utilizar para entrenamiento y testeo de la primera aproximación de predicciones del volumen de tráfico.....	115
Tabla 7.1: Presupuesto del material utilizado en el proyecto.....	125
Tabla 8.1: Tabla vehicle_routes.....	127
Tabla 8.2: Tabla road_links.....	128
Tabla 8.3: Tabla vehicle_trajectories_training.....	129
Tabla 8.4: Tabla traffic_volume_tollgates_training.....	130
Tabla 8.5: Tabla weather_data.....	131
Tabla 8.6: Tabla travel_time_intersection_to_tollgate.....	132
Tabla 8.7: Tabla traffic_volume_tollgates.....	132
Tabla 8.8: Tabla road_links_modified.....	133
Tabla 8.9: Tabla vehicle_routes_modified.....	134
Tabla 8.10: Tabla vehicle_trajectories_training_modified.....	135

Tabla 8.11: Tabla traffic_volume_tollgates_training_modified.....	136
Tabla 8.12: Tabla travel_time_intersection_to_tollgate_modified.....	137
Tabla 8.13: Tabla traffic_volume_tollgates_modified.....	138
Tabla 8.14: Tabla travel_time_intersection_to_tollgate_test.....	139
Tabla 8.15: Tabla traffic_volume_tollgates_test.....	140
Tabla 8.16: Tabla tabla_resultado_average_travel_time.....	141
Tabla 8.17: Tabla tabla_resultado_traffic_volume.....	141
Tabla 8.18: Tabla travel_time_intersection_to_tollgate_real.....	142
Tabla 8.19: Tabla traffic_volume_tollgates_real.....	142
Tabla 8.20: Tabla de predicciones de la primera aproximación de predicciones del tiempo promedio de viaje (La fila con un '-' corresponde a una ruta, día e intervalo de tiempo de la que no se dispone dato real para comparar).....	181

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 sumarización 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 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.
- En el capítulo 5 se exponen las conclusiones y las líneas futuras planteadas para seguir desarrollando el proyecto llevado a cabo.
- En el capítulo 7 se detalla el presupuesto utilizado para llevar a cabo el proyecto.
- En el capítulo 8 (anexo) se visualizan las estructuras de las distintas tablas de las bases de datos creadas para las tareas de predicción.

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

```
javisunami@javiramos:~/Escritorio/TFG/machineLearningProyecto/Aproximaciones/Tiempo promedio de viaje/2º 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.

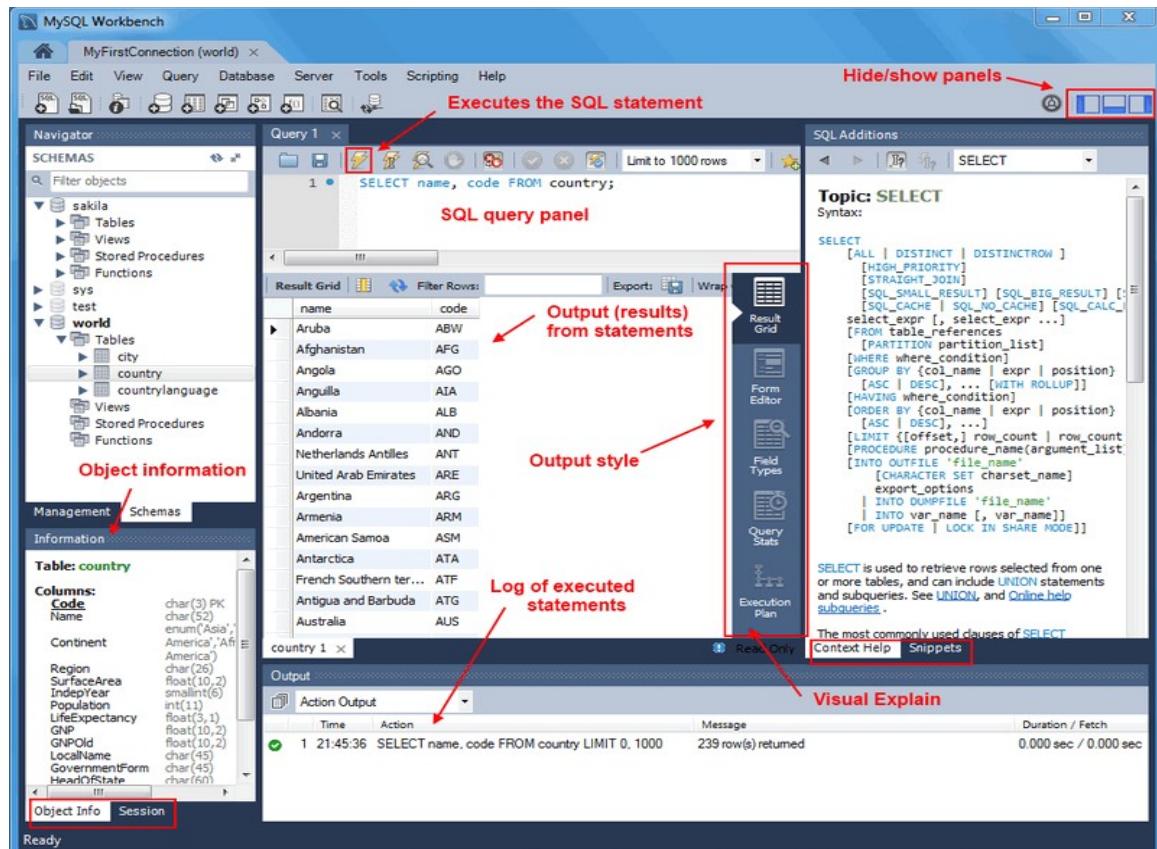


Figura 2.2: Entorno gráfico de MySQL Workbench

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.

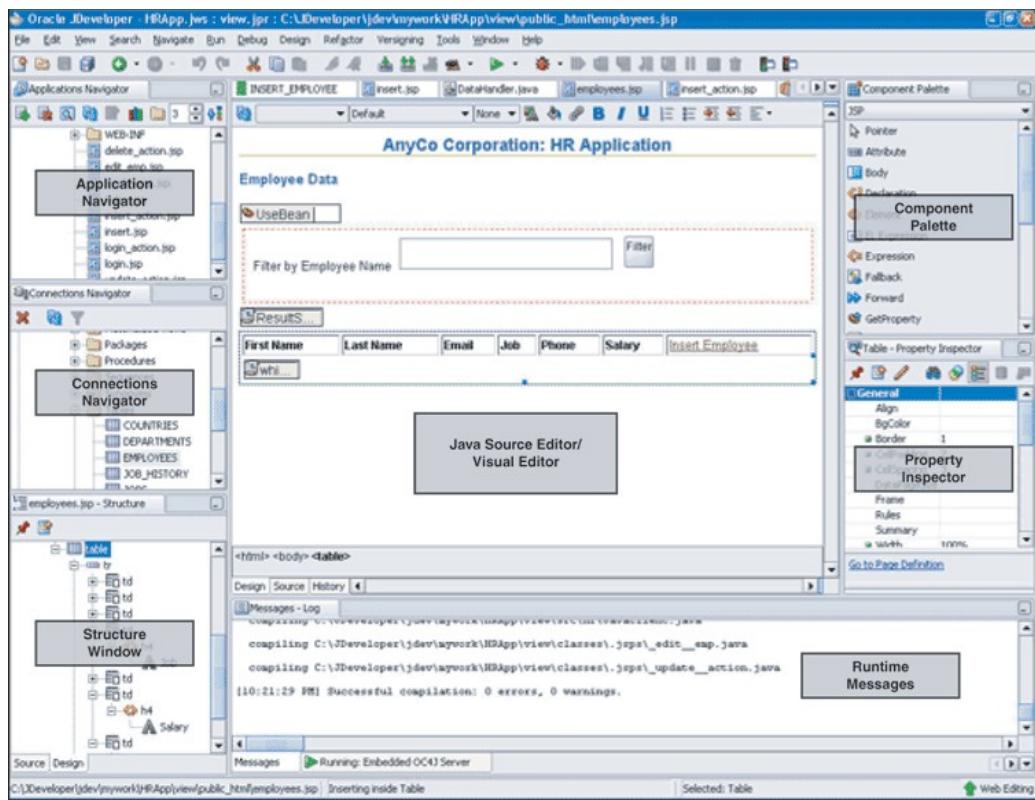


Figura 2.3: Entorno gráfico del entorno de desarrollo Oracle JDeveloper

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*.

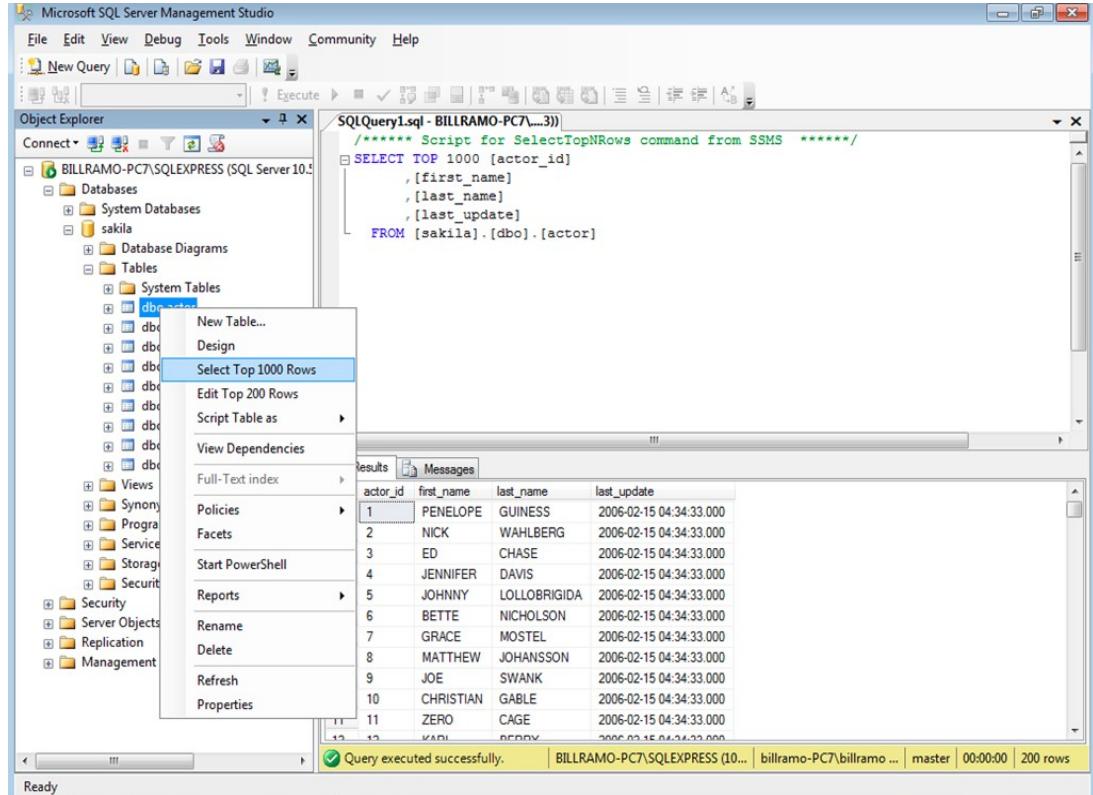


Figura 2.4: Entorno gráfico de Microsoft SQL Server Management Studio

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*.

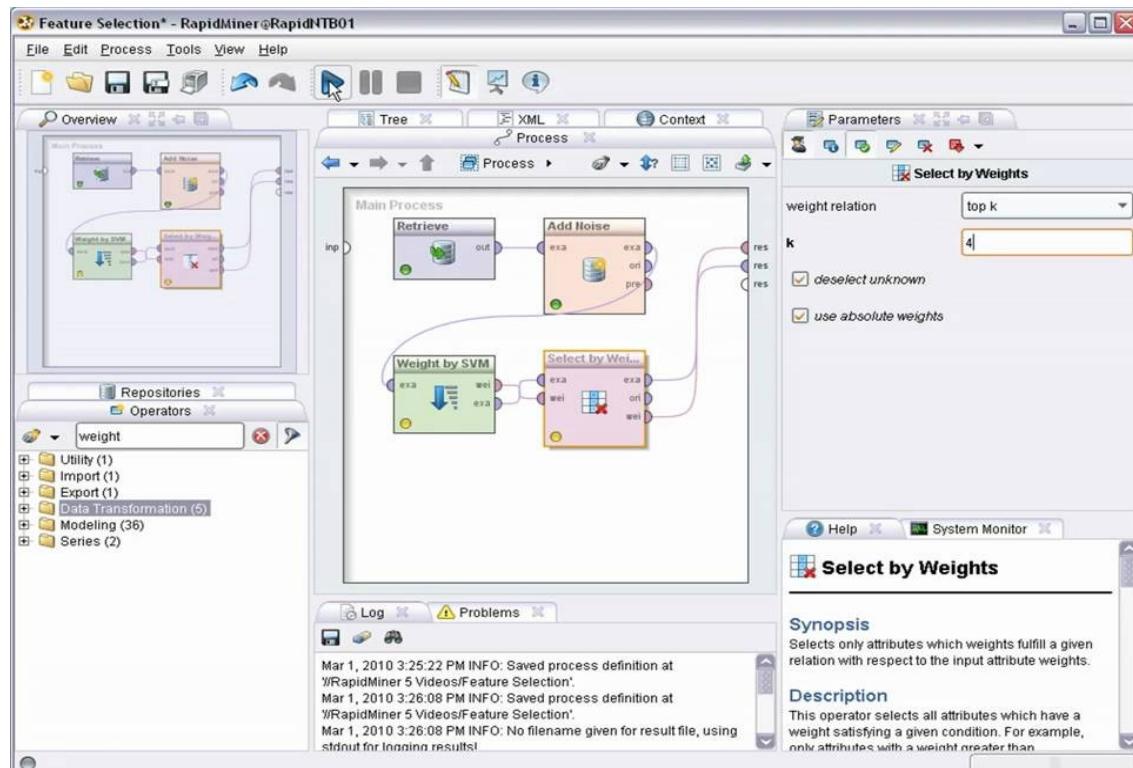


Figura 2.5: Entorno gráfico de RapidMiner

2.2.2.2 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 Appletiser* para su uso en aplicaciones y puede conectarse a bases de datos utilizando *CJD*.

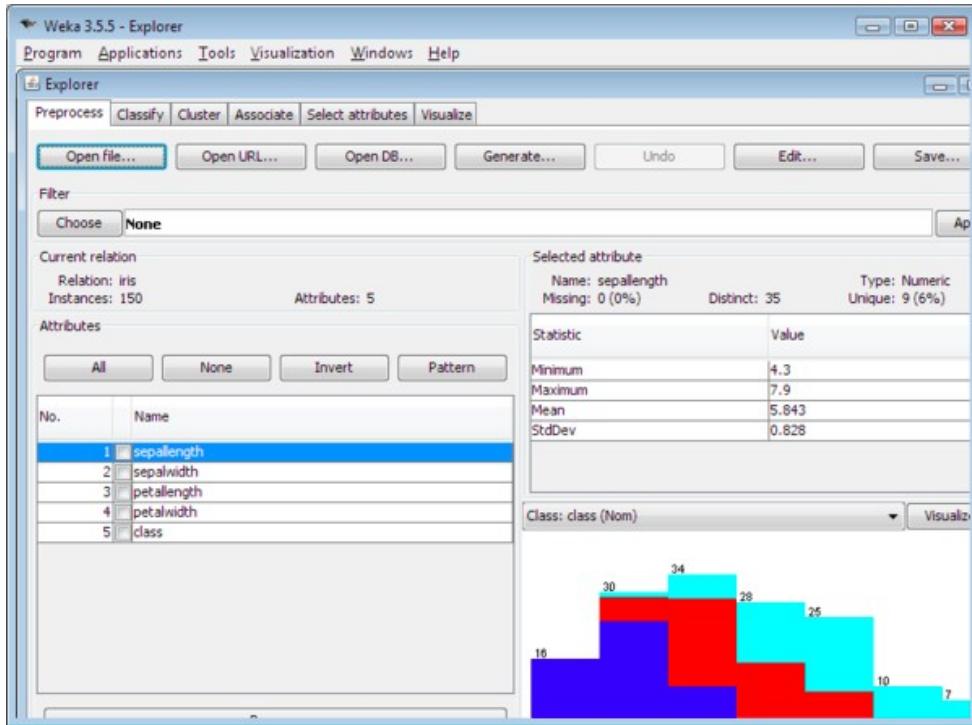


Figura 2.6: Entorno gráfico de Weka

2.2.3 Otros tipos de software empleados

2.2.3.1 Github

GitHub es una plataforma de **desarrollo colaborativo de software** para alojar proyectos utilizando el sistema de control de versiones *Git*. GitHub, aparte de ser un servicio de alojamiento de código, ofrece varias herramientas útiles para el **trabajo en equipo**. Entre ellas, caben destacar:

- Una *wiki* para el mantenimiento de las distintas versiones de las páginas.
- Un *sistema de seguimiento de problemas* que permiten a los miembros de un equipo detallar un problema con el software desarrollado o una sugerencia que se desee hacer.
- Una *herramienta de revisión de código*, donde se pueden añadir anotaciones en cualquier punto de un fichero y debatir sobre determinados cambios realizados en una confirmación específica.
- Un *visor de ramas* donde se pueden comparar los progresos realizados en las distintas ramas de nuestro repositorio.

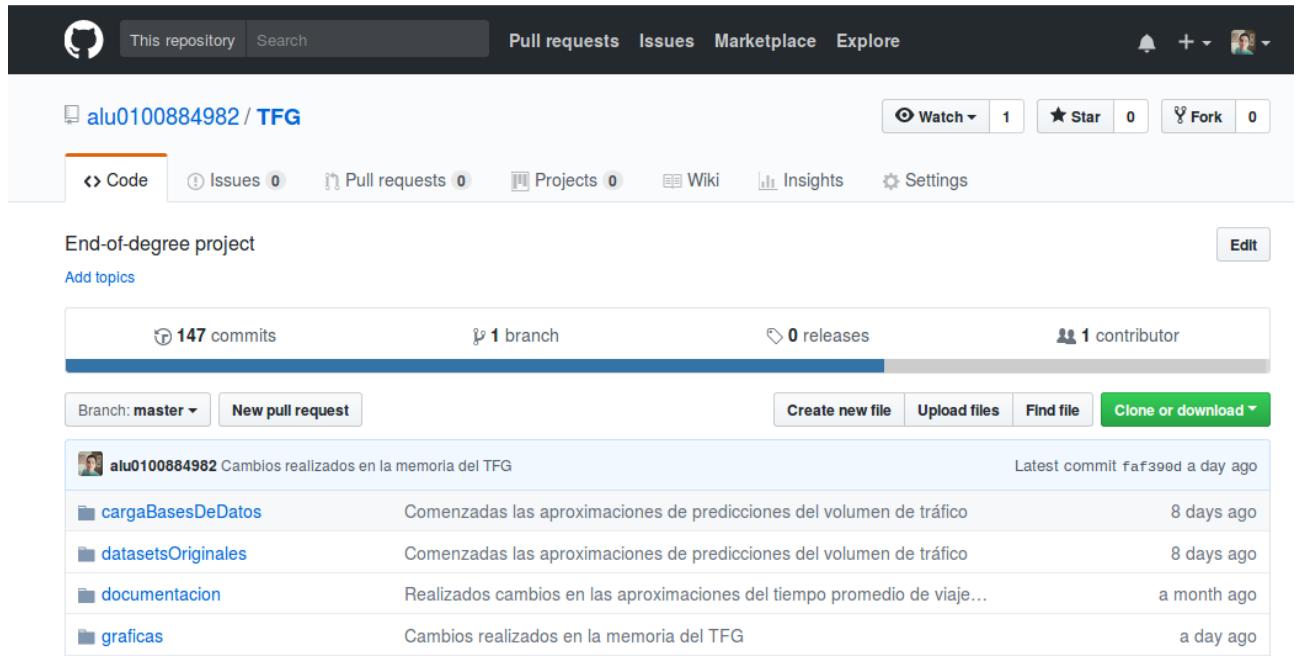


Figura 2.7: Interfaz gráfica de Github

2.2.3.2 Gedit

Gedit es un editor de textos de propósito general que enfatiza la simplicidad y facilidad de uso. Incluye herramientas para la edición de código fuente y textos estructurados, como lenguajes de marcado . Es el editor predeterminado de *GNOME*.

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.

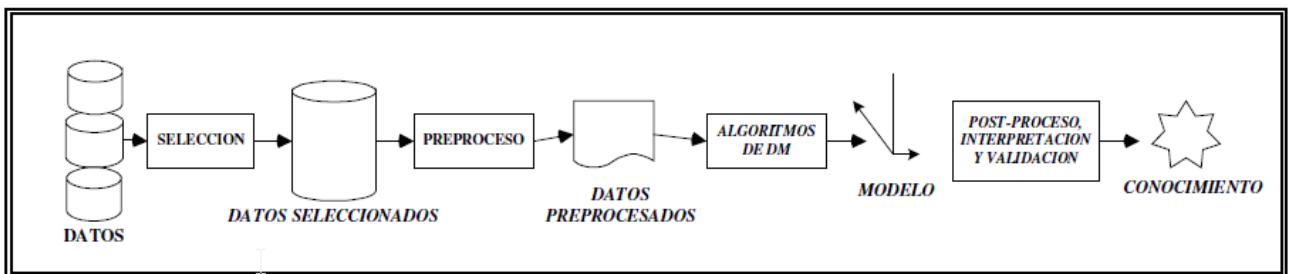


Figura 3.1: El proceso de descubrimiento de conocimiento en bases de datos (KDD)

- 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

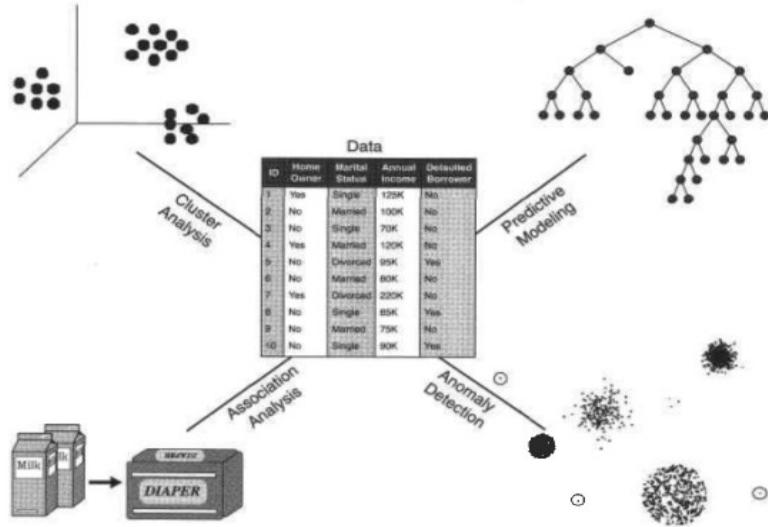


Figura 3.2: Cuatro de las tareas principales de minería de datos

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, después 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 llevan 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:

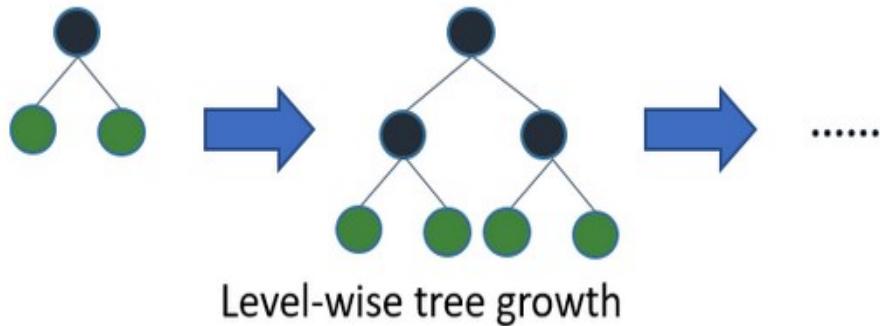


Figura 3.3: Crecimiento level-wise del árbol en XGBoost

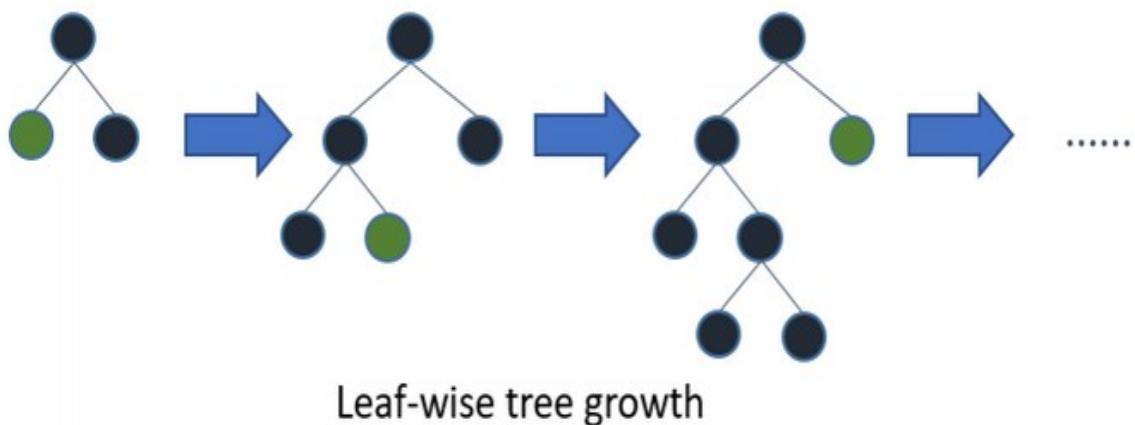


Figura 3.4: Crecimiento leaf-wise del árbol en LightGBM

3.2.2.2 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.

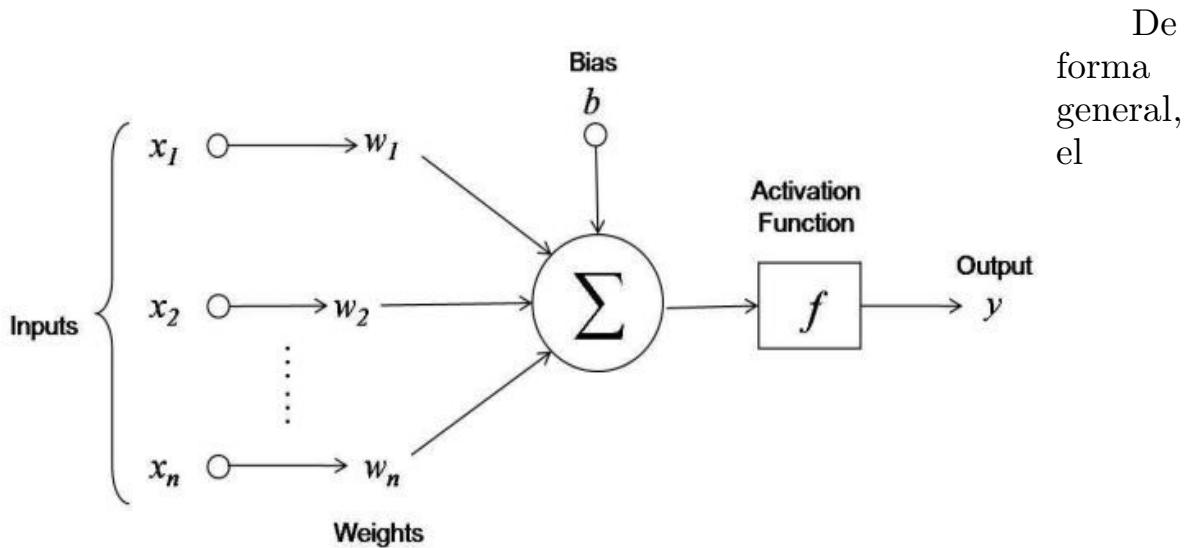


Figura 3.5: Estructura de una neurona artificial
funcionamiento de una neurona artificial es el siguiente:

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**.

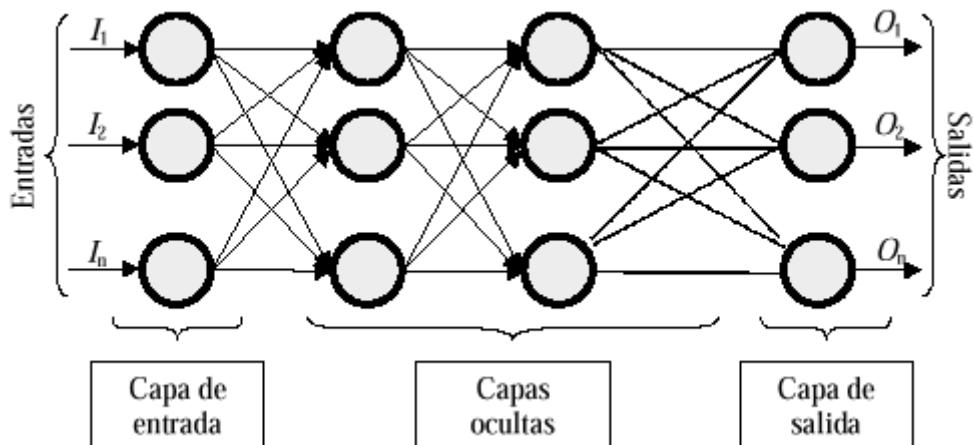


Figura 3.6: Estructura de una red neuronal

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 *categóricos*, 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:

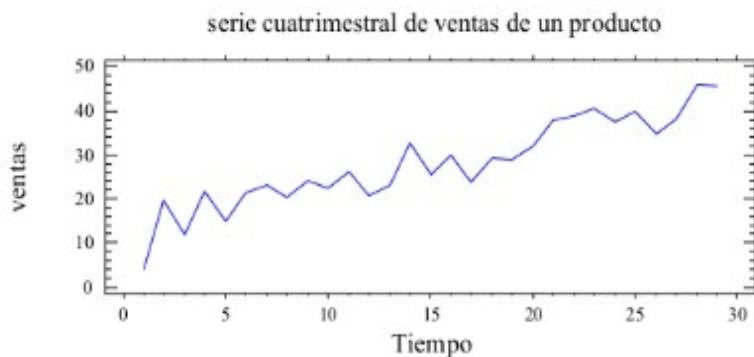


Figura 3.7: Ejemplo de serie temporal

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.

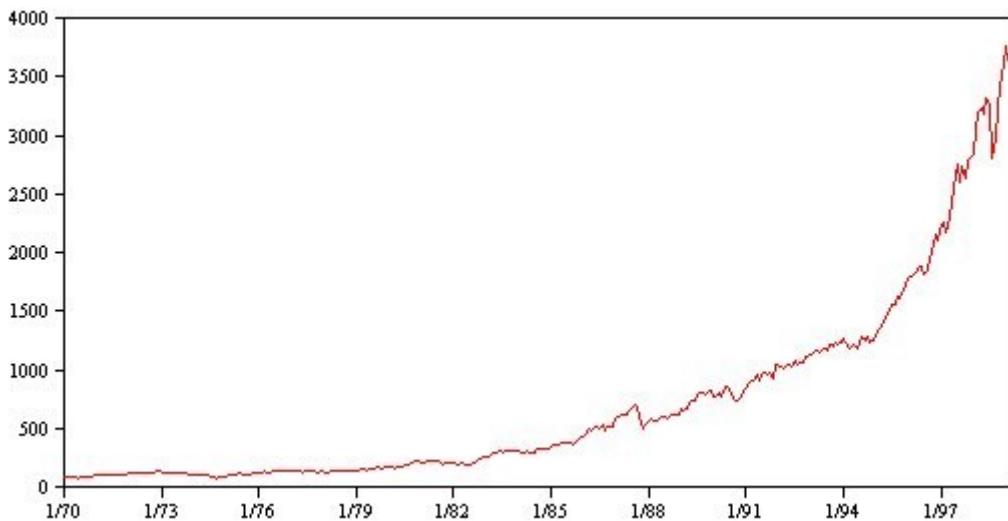


Figura 3.8: Ejemplo de una serie temporal con tendencia

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

Autocovarianza

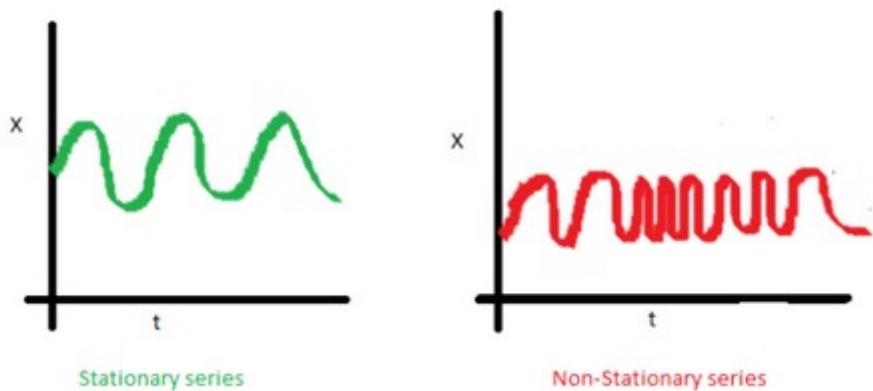


Figura 3.9: Comparación de una serie estacionaria con una no estacionaria con respecto a la autocovarianza estacionales resultan útiles para obtener buenos ajustes y previsiones.

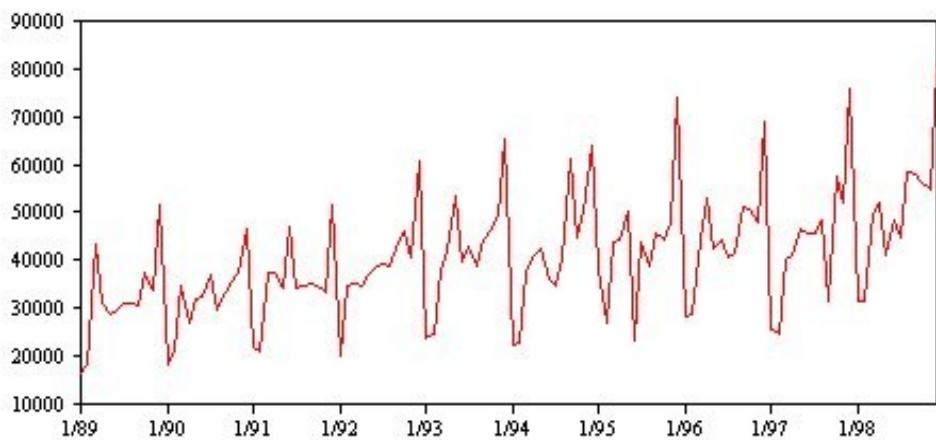


Figura 3.10: Ejemplo de una serie temporal con ciclos estacionales

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.

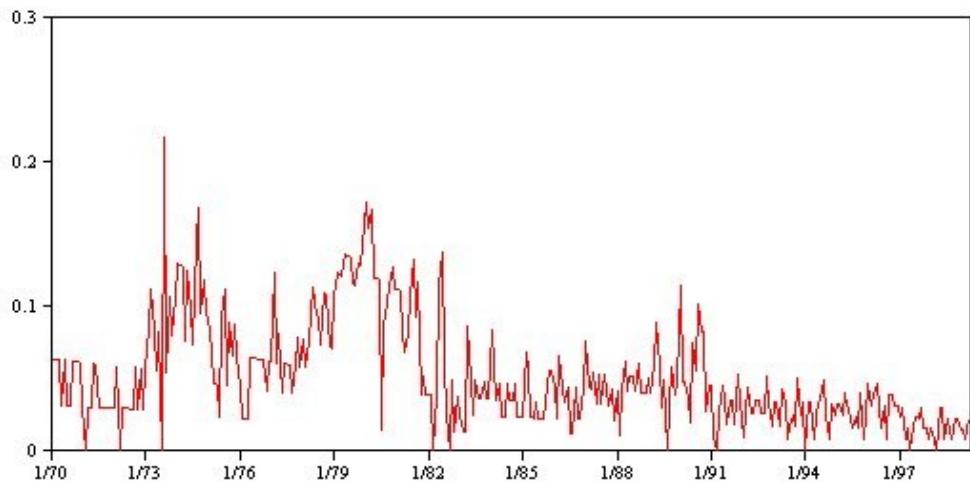


Figura 3.11: Serie temporal con pulsos

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.

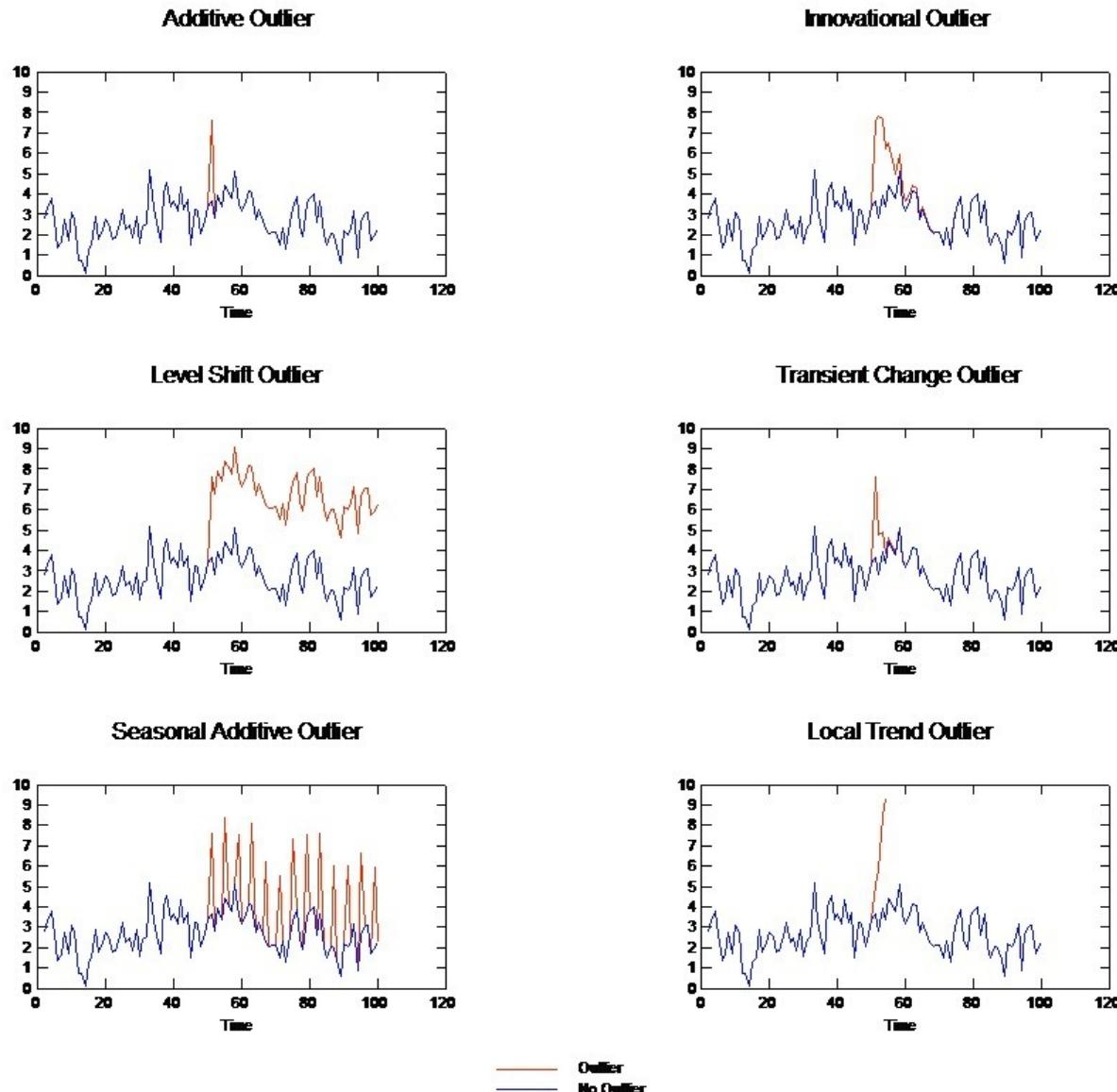


Figura 3.12: Distintos tipos de valores atípicos

- **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 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:

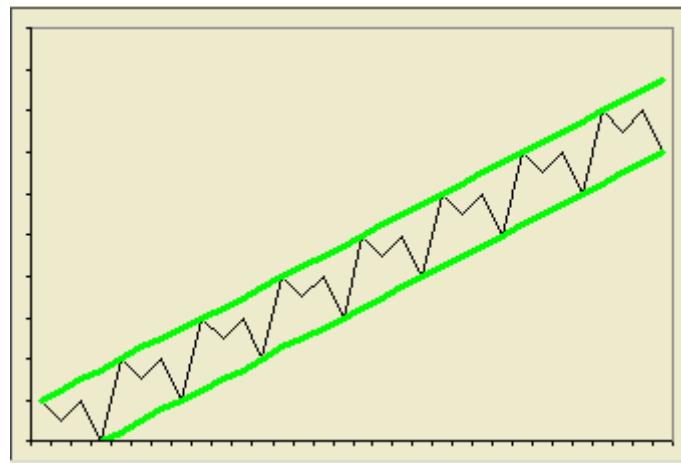


Figura 3.13: Serie temporal con tendencia
agregada a la estacionalidad

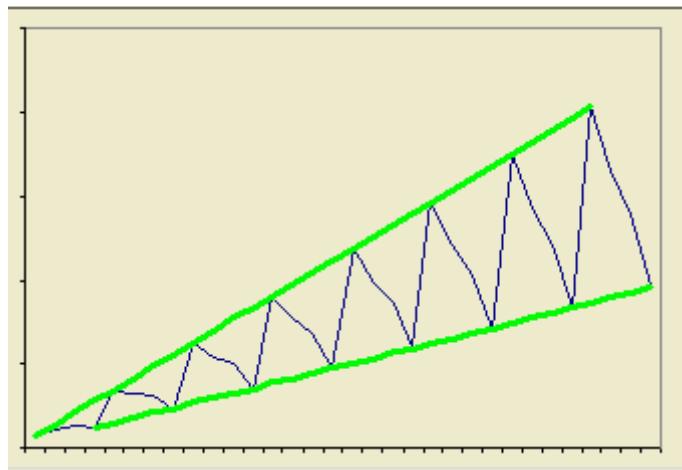


Figura 3.14: Serie temporal con tendencia multiplicada por la estacionalidad

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

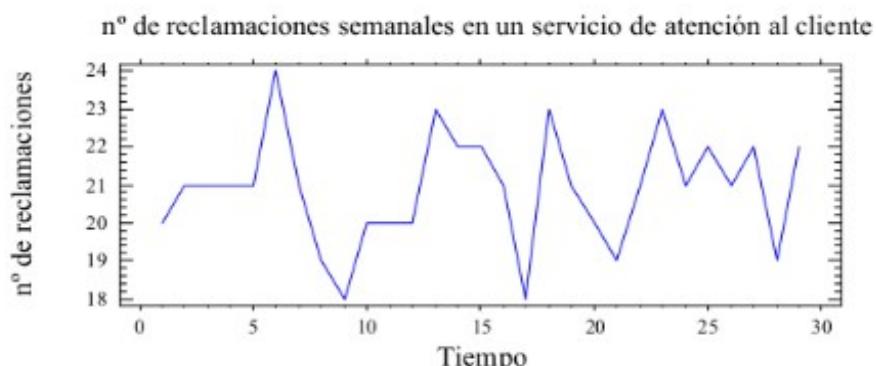


Figura 3.15: Ejemplo de serie estacionaria

- **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.

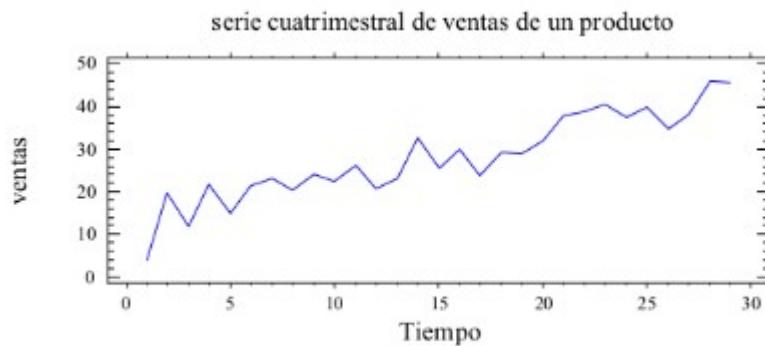


Figura 3.16: Ejemplo de serie no estacionaria

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

Media constante

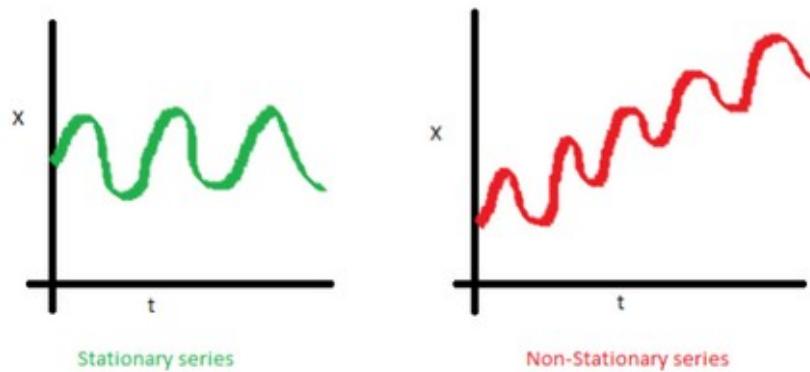


Figura 3.17: Comparación de una serie estacionaria con una no estacionaria con respecto a la media

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.

homoscedasticidad

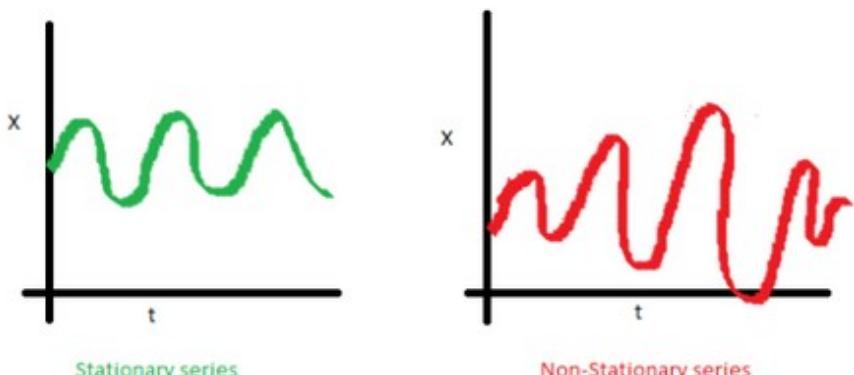


Figura 3.18: Comparación de una serie estacionaria con una no estacionaria con respecto a la varianza

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

Autocovarianza

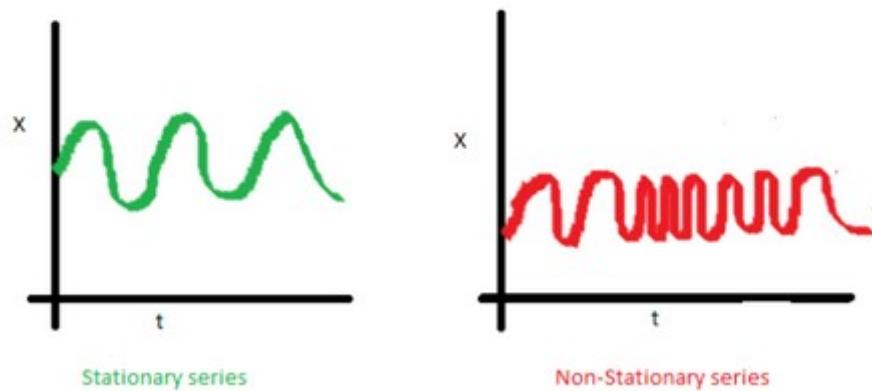


Figura 3.19: Comparación de una serie estacionaria con una no estacionaria con respecto a la autocovarianza

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:

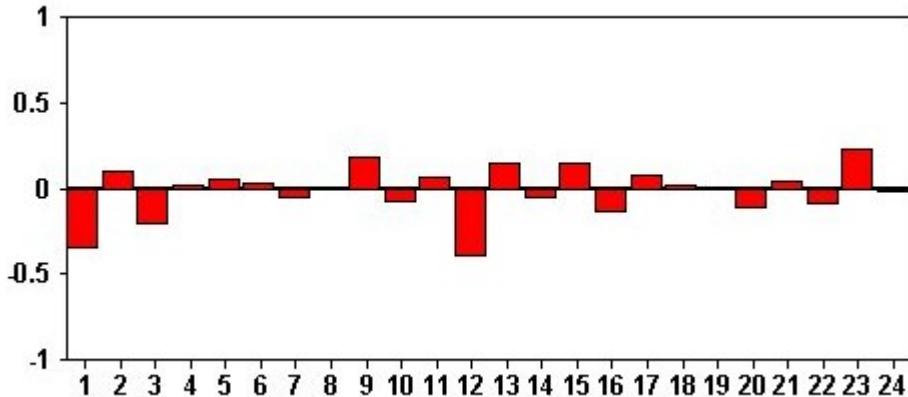


Figura 3.20: Ejemplo de un gráfico de autocorrelación

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,

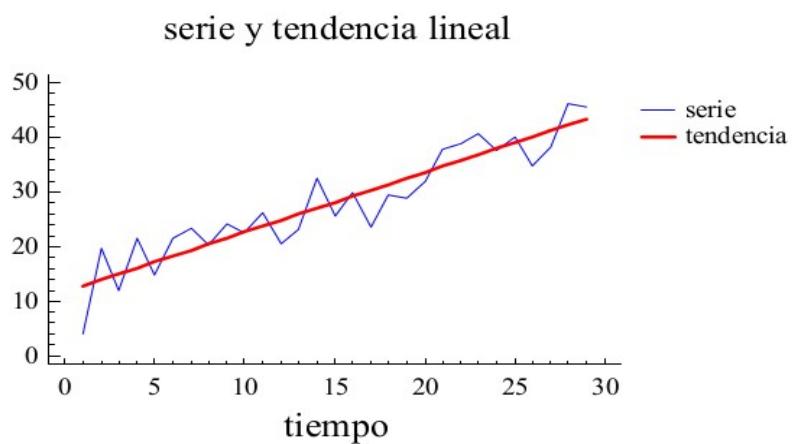


Figura 3.21: Serie temporal y su tendencia

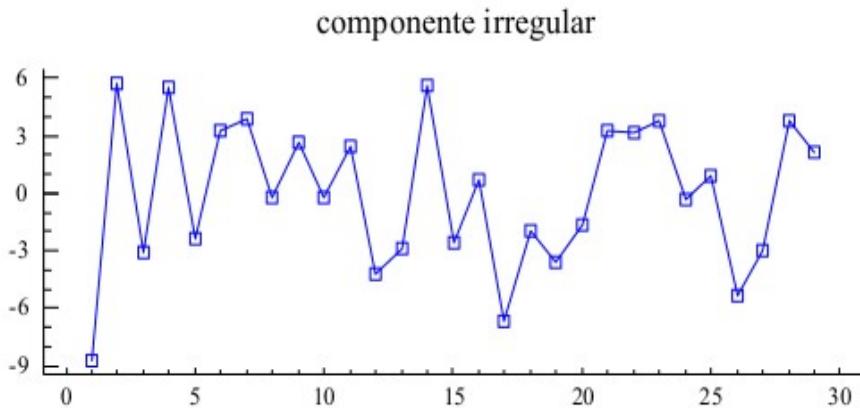


Figura 3.22: Componente regular de la serie temporal tras eliminar su tendencia

- **Tendencia evolutiva:** A menudo, la tendencia de la serie no sigue una recta y evoluciona a lo largo del tiempo. En ese caso, un método general de estimar la tendencia es suponer que evoluciona lentamente en el tiempo, y que se puede aproximar con una función sencilla para intervalos cortos del tiempo. Para ello, suponemos que la representación de la tendencia por una recta es *válida para tres períodos consecutivos de tiempo $t-1$, t y $t+1$* , y representamos las tendencias en los tres periodos consecutivos de la siguiente manera:

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

$$T_t = T_t$$

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

Si hacemos la media de tres observaciones consecutivas

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

entonces

$$m_t = T_t + \frac{I_{t-1} + I_t + I_{t+1}}{3}$$

y como la componente irregular tiene **media cero**, la media de los tres valores del componente irregular se puede suponer que es *despreciable frente a la tendencia*, y m_t representa la tendencia en ese instante. Esta operación se denomina *media móvil de orden tres*.

Este método, denominado *medias móviles*, consiste fundamentalmente en agrupar sistemáticamente un número fijo k de valores de la serie y determinar para cada grupo su media. Para entender este concepto, se expone el siguiente ejemplo:

mes	1	2	3	4	5	6	7	8	9	10
nº accidentes	20	25	15	22	30	42	48	51	45	49

Figura 3.23: Número de accidentes de tráfico durante 10 meses



Figura 3.24: Representación de los datos de accidentes de tráfico

Los siguientes datos corresponden al número de accidentes de tráfico durante 10 meses registrados en una determinada zona considerada como conflictiva (**Figura 3.23**). Si nos fijamos en la representación de estos datos (**Figura 3.24**), vemos que su tendencia es ascendente. Para obtenerla, vamos a obtenerla mediante **medias móviles de orden 3**. Tomemos por tanto, los tres primeros valores de la serie y calculemos su media:

$$20,25,15 \rightarrow y_1 = \frac{20+25+15}{3} = 20$$

Figura 3.25: Primera media de la serie

Quitemos ahora el primer valor y añadamos el cuarto:

$$25,15,22 \rightarrow y_2 = \frac{25+15+22}{3} = 20,67$$

Figura 3.26: Segunda media de la serie

Quitando el segundo valor y añadiendo el quinto:

$$15,22,30 \rightarrow y_3 = \frac{15+22+30}{3} = 22,33$$

Figura 3.27: Tercer media de la serie

Repitiendo esta operación aparecen todas las medias móviles de orden 3. Asignaremos cada una de ellas al periodo mediano correspondiente:

mes	1	2	3	4	5	6	7	8	9	10
y'_i		20.00	20.67	22.33	31.33	40.00	47.00	48.00	48.33	

Figura 3.28: Medias móviles de orden 3

tendencia (por medias móviles de orden 3)

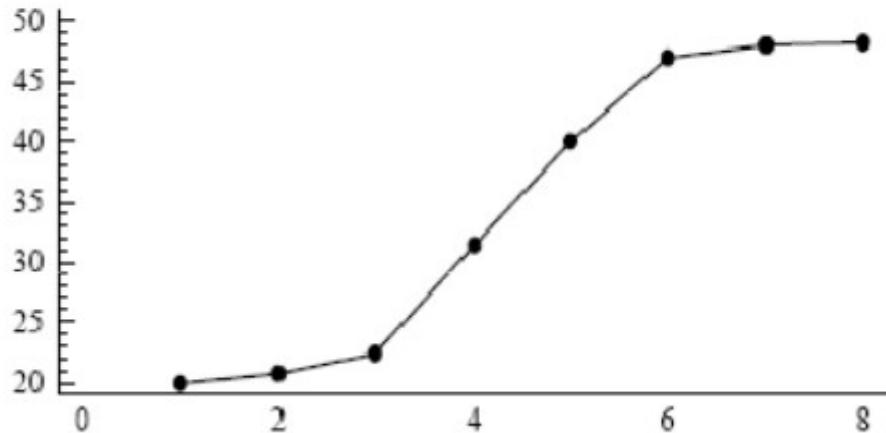


Figura 3.29: Nueva serie de medias móviles

Un ejemplo más visual de cómo quedaría la estimación de la tendencia y la componente regular tras eliminar dicha tendencia es la siguiente:

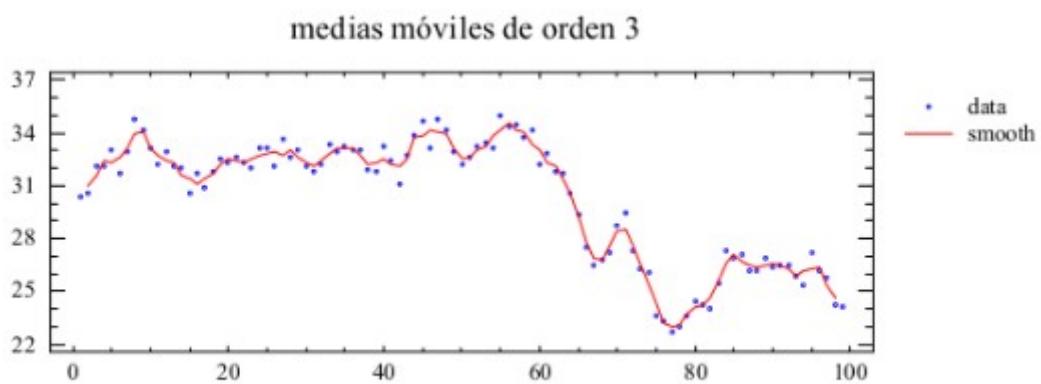


Figura 3.30: Ejemplo de estimación de la tendencia mediante medias móviles

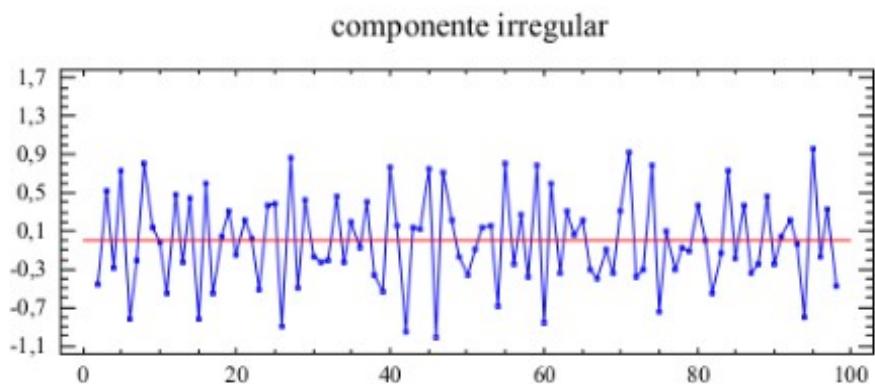


Figura 3.31: Componente irregular tras eliminar la tendencia

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

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

Un ejemplo de este método es el siguiente:

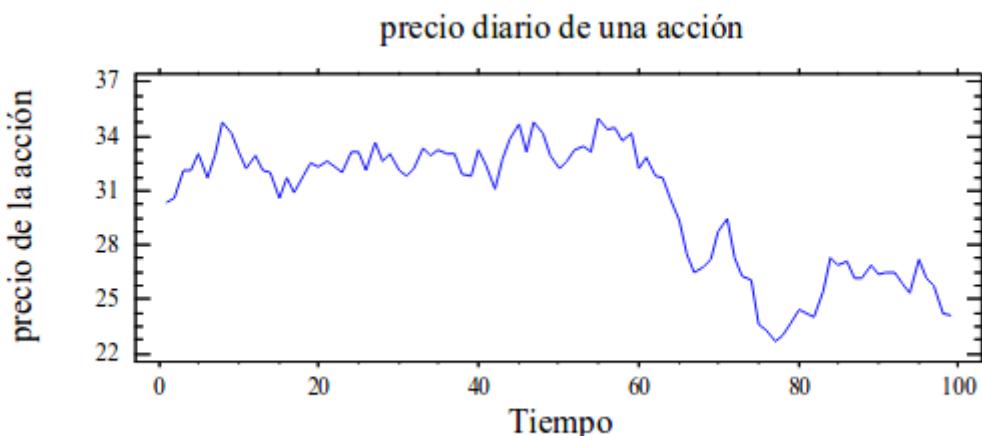


Figura 3.32: Serie temporal antes de la diferenciación

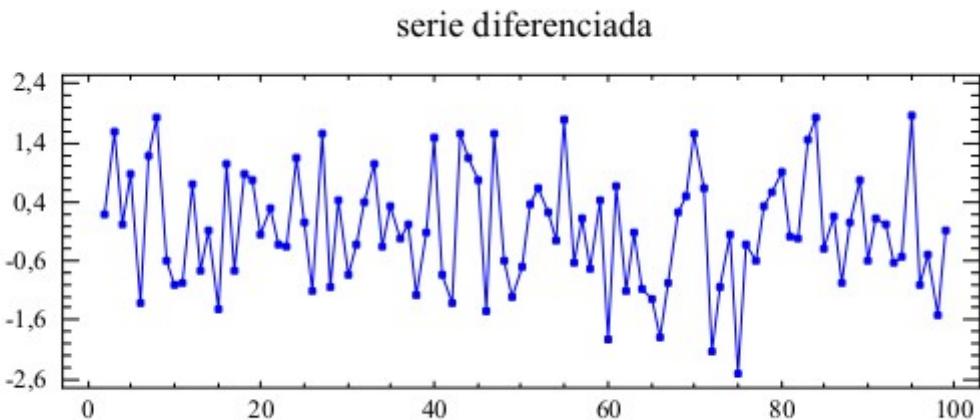


Figura 3.33: Serie temporal después de la diferenciación

Para estimar la variable de *estacionalidad* se pueden tener en cuenta estos métodos:

- **Variación de la media del período respecto de la media global:** Un método de estimar el efecto estacional (por ejemplo mensual) es considerar cómo varía la media del período (mes) respecto de la media global. Para entender este método, vamos a eliminar de la serie la *componente estacional*, es decir, se **desestacionaliza la serie** mediante los últimos 6 años de la serie de la . Esta serie presenta una estacionalidad mensual, de modo que se colocan los datos en una tabla de doble entrada (**Figura 3.35**):

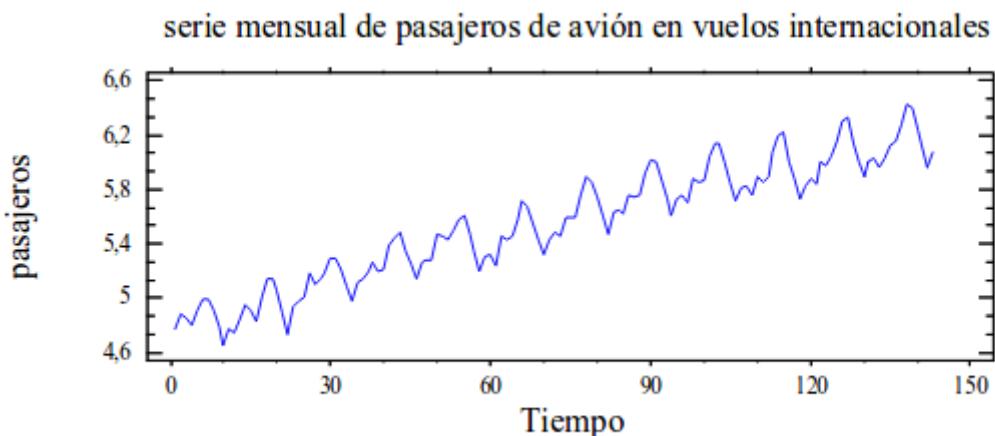


Figura 3.34: Serie temporales antes de la desestacionalización

	90	91	92	93	94	95	medias	coef. est.
Enero	5.49	5.65	5.75	5.83	5.89	6.03	5.77	-0.14
Febrero	5.45	5.62	5.71	5.76	5.83	5.97	5.72	-0.19
Marzo	5.59	5.76	5.87	5.89	6.01	6.04	5.86	-0.05
Abril	5.59	5.75	5.85	5.85	5.98	6.13	5.86	-0.05
Mayo	5.60	5.76	5.87	5.89	6.04	6.16	5.89	-0.02
Junio	5.75	5.92	6.05	6.08	6.16	6.28	6.04	0.13
Julio	5.90	6.02	6.14	6.20	6.31	6.43	6.17	0.26
Agosto	5.85	6.00	6.15	6.22	6.33	6.41	6.16	0.25
Septiembre	5.74	5.87	6.00	6.00	6.14	6.23	6.00	0.09
Octubre	5.61	5.72	5.85	5.88	6.01	6.13	5.87	-0.04
Noviembre	5.47	5.60	5.72	5.74	5.89	5.97	5.73	-0.18
Diciembre	5.63	5.72	5.82	5.82	6.00	6.07	5.84	-0.07

Figura 3.35: Los últimos 6 años de la serie temporal de la Figura 3.34

En este ejemplo hay efecto estacional mensual y, por tanto, existen 12

coeficientes estacionales, uno para cada mes del año. Para estimarlos, se calcula primero la media de las observaciones para cada mes, M_1, \dots, M_{12} , y el coeficiente estacional resulta ser:

$$S_i = M_i - M$$

para $i = 1, \dots, 12$

donde M es la media total de las observaciones. Así se puede observar que los meses con observaciones más pequeñas (por debajo de la media general) tienen *coeficientes estacionales negativos*, mientras que los meses con observaciones mayores tienen *coeficientes estacionales positivos*. Un ejemplo de cálculo de este coficiente estacional es el siguiente:

- Media total de las observaciones es $M=5,91$,
- Por lo tanto, $S_1=M_1 - M = 5,77 - 5,91 = -0,14$

La suma de los coeficientes estacionales tiene que ser cero. Las temporadas más bajas son las correspondientes a los meses de febrero y noviembre, y las más altas las de los meses de julio y agosto.

Tras aplicar este método sobre la serie temporal se obtiene una *serie desestacionalizada*, es decir, una serie donde se ha eliminado el efecto de cada mes y que se obtiene restando al valor de cada mes el coeficiente estacional de dicho mes. Teniendo una serie desestacionalizada, podemos observar el *comportamiento general de una serie cronológica sin tener en cuenta la componente estacional*. En nuestro ejemplo, con todos los datos, la serie desestacionalizada es:

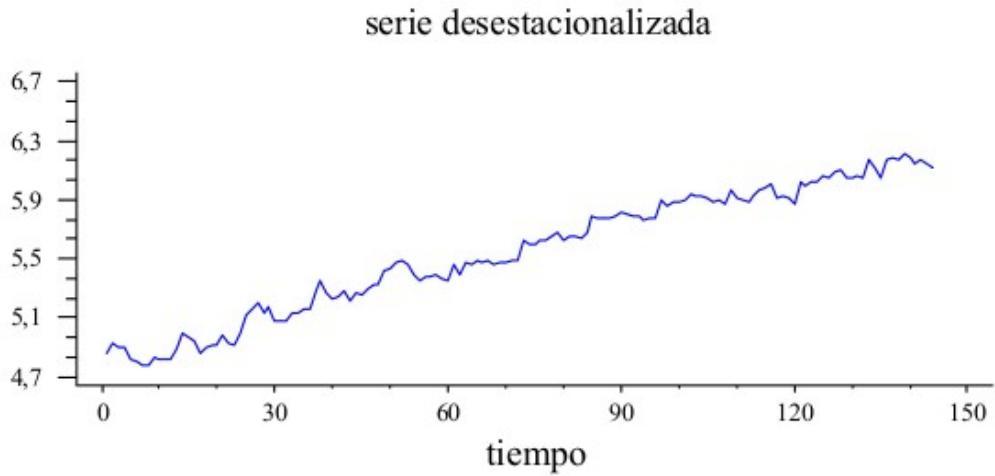


Figura 3.36: Serie desestacionalizada

- **Diferenciación estacional de la serie:** Es un método más general que consiste en no hacer ninguna hipótesis sobre la forma general de la estacionalidad a corto plazo y suponer simplemente que evoluciona lentamente en el tiempo. Para ello, construimos una nueva serie diferenciando cada valor en un instante determinado con aquel valor anterior que se encuentra s instantes de tiempo hacia detrás:

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

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

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

Esta serie se denomina *serie diferenciada estacionalmente*.

A la hora de eliminar tanto la *tendencia* como la *estacionalidad* de una serie temporal para aplicar un modelo probabilístico sobre la componente irregular de la misma, es aconsejable eliminar primero la *tendencia* y en segundo lugar *desestacionalizar la serie* puesto que, si se realiza esto al revés, puede que siga habiendo *estacionalidad* tras eliminar previamente la

estacionalidad y luego la *tendencia* y se tenga que volver a repetir pasos.

Las transformaciones suelen ser útiles para estabilizar una serie antes de estimar modelos y se aplican sobre las componentes de las series temporales. Esto es especialmente importante para **modelos ARIMA**, que necesitan que *las series sean estacionarias antes de estimar los modelos*.

3.2.4.8 Definición del modelo ARIMA

Un **modelo ARIMA** es una clase de modelos estadísticos para *analizar y pronosticar datos de series temporales*. Abarca explícitamente un conjunto de estructuras estándar en los datos de series temporales y, como tal, proporciona un método simple pero potente para hacer pronósticos hábiles de series temporales.

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

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

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

- **Autorregresivo (p).** Es el número de órdenes autorregresivos del modelo. Los órdenes autorregresivos especifican los valores previos de la serie utilizados para predecir los valores actuales. Por ejemplo, un orden autorregresivo igual a 2 especifica que se van a utilizar los valores de la serie correspondientes a dos períodos de tiempo del pasado para predecir el valor actual.
- **Diferencia (d).** Especifica el orden de diferenciación aplicado a la

serie antes de estimar los modelos. La diferenciación es necesaria si hay tendencias (las series con tendencias suelen ser no estacionarias y el modelado de ARIMA asume la estacionariedad) y se utiliza para eliminar su efecto. El orden de diferenciación se corresponde con el grado de la tendencia de la serie (la diferenciación de primer orden representa las tendencias lineales, la diferenciación de segundo orden representa las tendencias cuadráticas, etc.).

- **Media móvil (q).** Es el número de órdenes de media móvil presentes en el modelo. Los órdenes de media móvil especifican el modo en que se utilizan las desviaciones de la media de la serie para los valores previos con el fin de predecir los valores actuales. Por ejemplo, los órdenes de media móvil de 1 y 2 especifican que las desviaciones del valor medio de la serie de cada uno de los dos últimos períodos de tiempo se tienen en cuenta al predecir los valores actuales de la serie.

Se puede utilizar el valor 0 para cualquier parámetro del modelo, lo que indica que no se debe utilizar ese elemento del modelo. De esta manera, el modelo ARIMA puede configurarse para realizar la función de un modelo ARMA, e incluso de un simple modelo AR, I o MA.

3.2.4.9 Modelos autorregresivos AR(p)

Un *modelo autorregresivo AR* describe una clase particular de proceso en que las observaciones en un momento dado son predecibles a partir de las observaciones previas del proceso más un término de error. El caso más simple es el *ARIMA(1,0,0)* o *AR(1)* o de primer orden, cuya expresión matemática es:

$$AR(1) \equiv X_t = \Phi_1 X_{t-1} + a_t$$

El proceso autorregresivo de orden **p**, representado por *ARIMA(p,0,0)* o simplemente por *AR(p)*, tiene la siguiente expresión matemática:

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

3.2.4.10 Modelo de medias móviles MA(q)

Un **modelo de medias móviles MA** describe una serie temporal en la que el valor actual puede predecirse a partir de la componente aleatoria de este momento y, en menor medida, de los impulsos aleatorios anteriores. El modelo $ARIMA(0,0,1)$, también denotado por $MA(1)$, viene dado por la expresión:

$$MA(1) \equiv X_t = a_t - v_1 a_{t-1}$$

El proceso de medias móviles de orden q , representado por $ARIMA(0,0,q)$ o también por $MA(q)$, viene dado por la expresión:

$$MA(q) \equiv X_t = a_t - v_1 a_{t-1} - v_2 a_{t-2} - \dots - v_q a_{t-q}$$

3.2.4.11 Modelos ARMA(p,q)

Una extensión natural de los modelos $AR(p)$ y $MA(q)$ es un tipo de modelos que incluyen tanto términos *autorregresivos* como de *medias móviles* y se definen como **ARIMA(p, 0, q)**. Se representan por la ecuación:

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

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

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

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

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

Un modelo **ARIMA(p, d, q)** permite describir una serie de observaciones después de que hayan sido diferenciadas **d** veces, a fin de extraer las posibles fuentes de no estacionariedad. Esta fórmula se puede aplicar a cualquier modelo.

3.2.4.12 Fases del modelo ARIMA

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

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

- La cuarta fase es la *predicción*, una vez que se ha obtenido un modelo adecuado se realizan predicciones con el mismo.

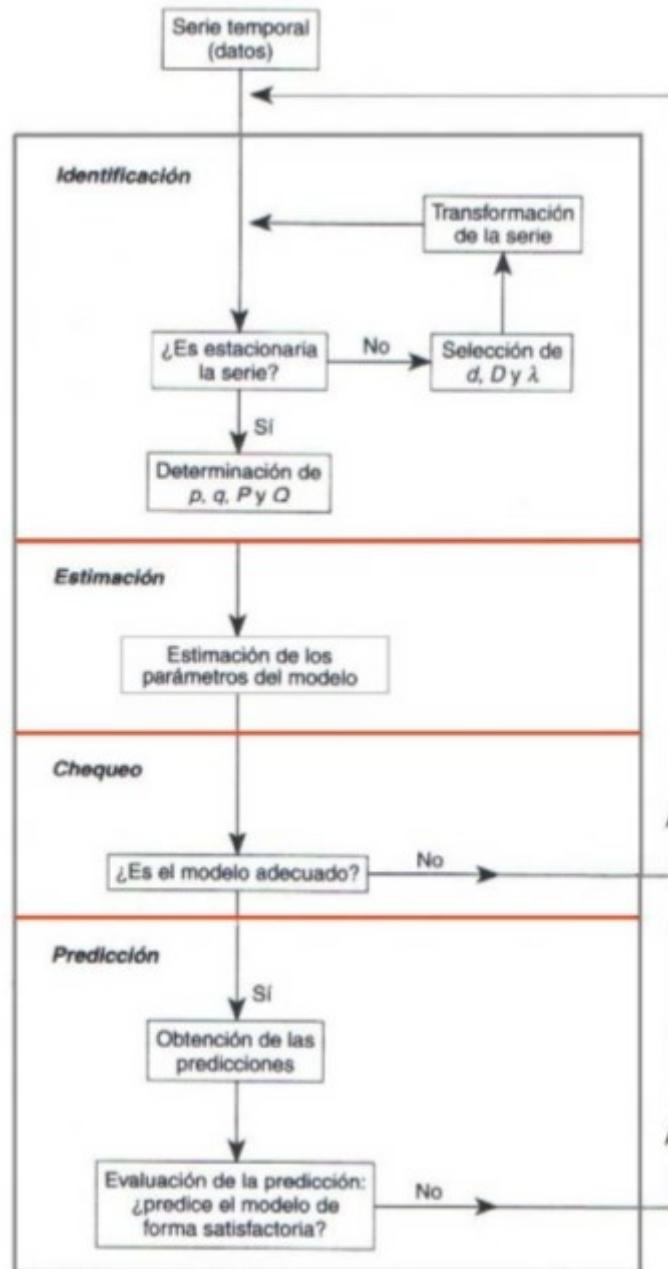


Figura 3.37: Fases del modelo ARIMA

Los pasos a seguir para llevar a cabo el análisis de datos con el objetivo de construir el modelo ARIMA son los siguientes:

1. **Recogida de datos:** Es conveniente disponer de 50 o más datos y, en el caso de series mensuales, trabajar entre seis y diez años completos.
2. **Representación gráfica:** Es de gran utilidad disponer de un gráfico

de la serie para decidir sobre la *estacionariedad*. En ocasiones, se utilizan medias y desviaciones típicas por subperiodo para juzgar sobre la estacionariedad de la serie.

3. **Transformación previa de la serie:** Cuando la serie no es estacionaria en varianza se requiere una transformación logarítmica. No obstante, la transformación logarítmica es muy frecuente incluso en series con dispersión relativamente constante en el tiempo. Una práctica habitual es ensayar con la serie original y en logaritmos y comprobar resultados.
4. **Eliminación de la tendencia:** La observación del gráfico de la serie indica la existencia o no de tendencia. Una tendencia lineal será corregida tomando primeras diferencias, que será el caso más frecuente. Una tendencia no lineal suele llevar en la práctica al uso de dos diferencias como mucho.
5. **Identificación del modelo:** Consiste en determinar el tipo de modelo más adecuado, esto es, el orden de los procesos autorregresivos y de medias móviles de las componentes regular y estacional. Técnicamente esta decisión se toma en base a las funciones de *autocorrelación* (FAC) y *autocorrelación parcial* (FAC parcial), tanto en la parte regular como estacional. Es habitual terminar eligiendo entre los procesos más simples AR(1), AR(2), MA(1), MA(2) y ARMA(1,1), tanto en la parte regular como estacional. En caso de duda pueden seleccionarse varios modelos alternativos que serán estimados y contrastados posteriormente, para definir finalmente el modelo adoptado.
6. **Estimación de los coeficientes del modelo:** Decidido el modelo, se procede a la estimación de sus parámetros, dado que se trata de un procedimiento iterativo de cálculo, pueden sugerirse valores iniciales.
7. **Contraste de validez del modelo:** Se utilizan distintos procedimientos para valorar el modelo o modelos inicialmente seleccionados: contraste de significación de parámetros, covarianzas entre estimadores, coeficiente de correlación, suma de cuadrados de errores, etc.

8. **Análisis detallado de los errores:** Se tendrán en cuenta las diferencias históricas entre valores reales y estimados por el modelo para su valoración final. Hay que verificar un comportamiento no sistemático de los mismos, así como analizar la posible existencia de errores especialmente significativos.
9. **Selección del modelo:** En base a los resultados de pasos anteriores, se decide sobre el modelo adoptado.
10. **Predicción:** El modelo seleccionado se utilizará como fórmula inicial de predicción.

3.2.4.13 Identificación práctica del modelo ARIMA

Identificar un modelo significa utilizar los datos recogidos, así como cualquier información de cómo se genera la serie temporal objeto de estudio para sugerir un conjunto reducido de posibles modelos que tengan muchas posibilidades de ajustarse a los datos. Ante una serie temporal empírica, se deben encontrar los valores (p, d, q) más apropiados.

- Si la serie temporal presenta una tendencia, lo primero que se debe hacer es convertirla en estacionaria mediante *una diferenciación de orden d*. Una vez diferenciada la serie, una buena estrategia consiste en comparar los correlogramas de la función *de autocorrelación* (ACF) y la función *de autocorrelación parcial* (ACFP), proceso que suele ofrecer una orientación para la formulación del modelo orientativo.
- Los *procesos autorregresivos* presentan función de autocorrelación parcial (ACFP) con un número finito de valores distintos de cero. Un proceso **AR(p)** tiene los primeros p términos de la función de autocorrelación parcial distintos de cero y los demás son nulos. En la práctica se considera que una muestra dada proviene de un proceso autorregresivo de orden p si los términos de la función de autocorrelación parcial son casi cero a partir del que ocupa el lugar p .
 - Los *procesos de medias móviles* presentan función de autocorrelación con un número finito de valores distintos de cero. Un proceso **MA(q)** tiene los primeros q términos de la función de autocorrelación distintos de cero y los demás son nulos.

Las dos propiedades descritas son muy importantes con vistas a la identificación de un proceso mediante el análisis de las funciones de autocorrelación y autocorrelación parcial (**Tabla 3.1**).

Proceso	Función de autocorrelación (ACF)	Función de autocorrelación parcial (ACFP)
MA(q)	Solo los q primeros coeficientes son significativos. El resto se anulan bruscamente (coef. 0 para retardo >q)	Decrecimiento rápido exponencial atenuado u ondas sinusoidales.
AR(p)	Decrecimiento rápido exponencial atenuado u ondas sinusoidales.	Solo los p primeros coeficientes son significativos. El resto se anulan bruscamente (coef. 0 para retardo >q)
ARIMA(p, d, q)	Comportamiento irregular en los retardos $(1, \dots, q)$ con q picos. Decrecimiento para retardos posteriores a q .	Decrece (aproximadamente con exponentiales atenuados y ondas sinusoidales). No cero pronto.

Tabla 3.1: Obtención de los parámetros (p, d, q) en los distintos modelos

3.2.5 k-Vecinos Más Cercanos

3.2.5.1 Definición

El algoritmo de los *k vecinos más cercanos* (k-NN Nearest Neighbour) es un sistema de clasificación supervisado basado en criterios de vecindad (los *sistemas de clasificación supervisados* son aquellos en los que, a partir de un conjunto de ejemplos clasificados (conjunto de entrenamiento), intentamos asignar una clasificación a un segundo conjunto de ejemplos). En particular, la idea básica sobre la que se fundamenta este paradigma es que un nuevo caso se va a clasificar en la *clase más frecuente a la que pertenezcan sus k vecinos más cercanos*. El paradigma se fundamenta, por tanto, en una idea muy simple e intuitiva, lo que unido a su fácil implementación hace que sea un paradigma clasificadorio muy extendido.

3.2.5.2 El algoritmo K-NN básico

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

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

Tabla 3.2: Estructura de los datos de entrada para el algoritmo KNN

La notación a utilizar es la siguiente:

- La **Tabla 3.2** corresponde a un fichero D de N casos, cada uno de los cuales está caracterizado por n variables predictoras, X_1, \dots, X_n , y una variable a predecir, la clase C .
- Los N casos se denotan por:

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

para todo $i=1, \dots, N$ donde

$$x_i = (x_{i,1} \dots x_{i,n})$$

para todo $i=1, \dots, N$ y

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

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

c^1, \dots, c^m denotan los m posibles valores de la variable clase C .

- El nuevo caso que se pretende clasificar se denota por

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

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

COMIENZO

Entrada: $D = \{(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_N, c_N)\}$

$\mathbf{x} = (x_1, \dots, x_n)$ nuevo caso a clasificar

PARA todo objeto ya clasificado (x_i, c_i)

calcular $d_i = d(\mathbf{x}_i, \mathbf{x})$

Ordenar $d_i (i = 1, \dots, N)$ en orden ascendente

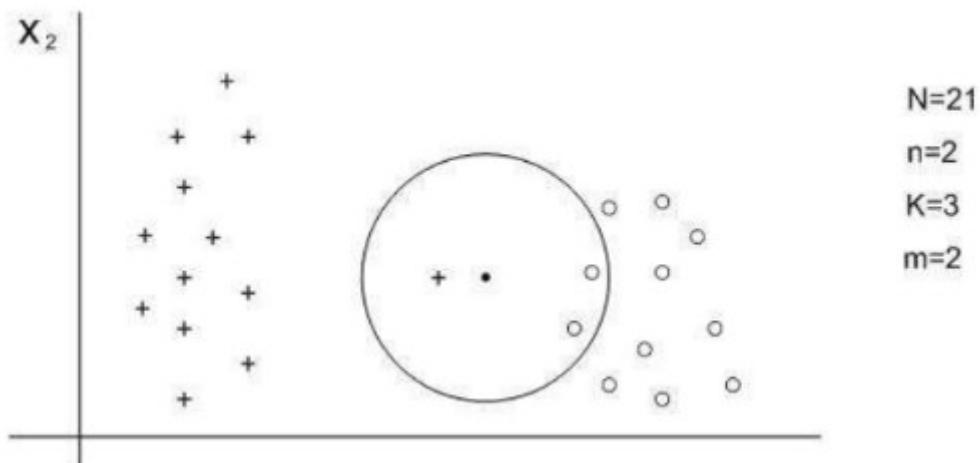
Quedarnos con los K casos $D_{\mathbf{x}}^K$ ya clasificados más cercanos a \mathbf{x}

Asignar a \mathbf{x} la clase más frecuente en $D_{\mathbf{x}}^K$

FIN

Figura 3.38: Algoritmo básico KNN

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



Tal y como se puede apreciar en la figura, tenemos 24 casos ya clasificados en dos posibles valores ($m=2$). Las variables predictoras son X_1 y X_2 , y se ha seleccionado $K=3$. De los 3 casos ya clasificados que se encuentran más cercanos al nuevo caso a clasificar, \mathbf{x} (representado por \bullet),

dos de ellos pertenecen a la clase \circ , por lo que el clasificador 3-NN predice la clase \circ para el nuevo caso. Nótese que el caso más cercano a x pertenece a la clase $+$. Es decir, que si hubiésemos utilizado un clasificador 1-NN, x se hubiese asignado a $+$.

En caso de que se produzca un *empate entre dos o más clases*, conviene tener una *regla heurística* para su ruptura. Un ejemplo de regla heurística para la ruptura de empates pueden ser seleccionar la clase con distancia menor, etc.

¿Meter regresión lineal y máquinas de soporte vectorial de forma muy resumida?

¿Quitar KNN?

3.2.5.3 Método de regresión basado en KNN

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

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

Todos los datos deben estar normalizados, para evitar que las características en el conjunto de entrada con valores más altos dominen el cálculo de la distancia.

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:

- *Comprendión 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*

peaje-dirección en total.

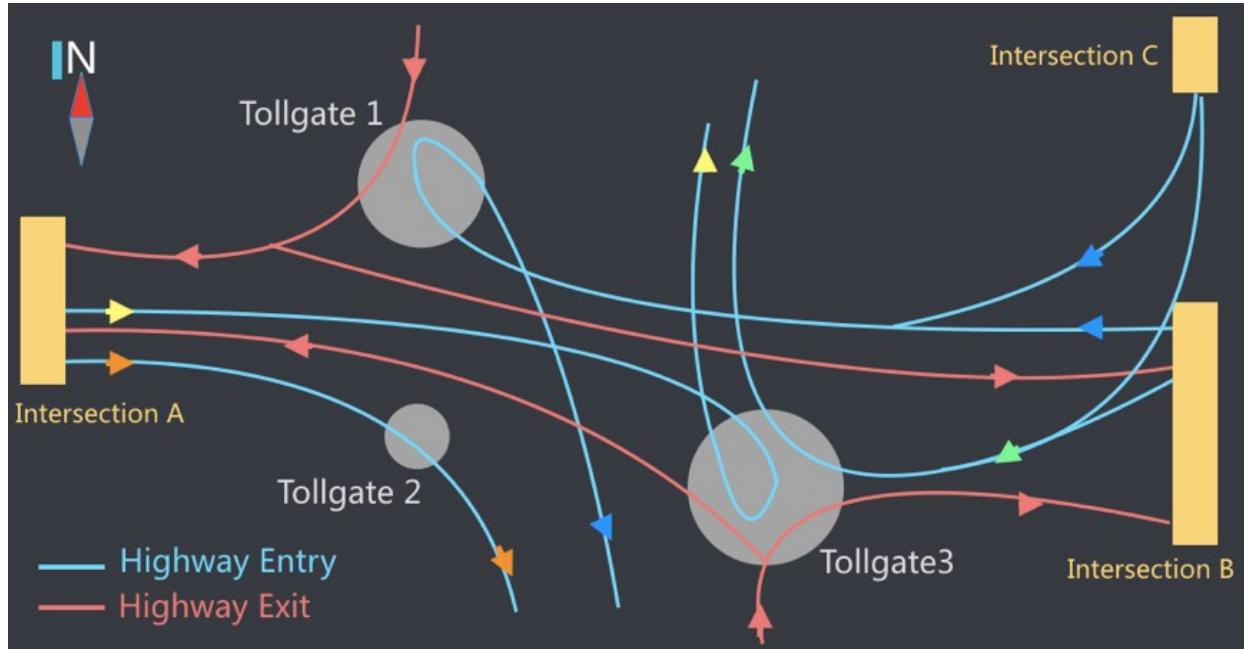


Figura 4.1: Topología de la red de carreteras de la zona objetivo

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**.

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



Figura 4.2: Ventanas de tiempo para la predicción del tráfico

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:

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

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

Esta fórmula se aplica a la primera tarea propuesta por la competición donde 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 .

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 cuatro bases de datos distintas. Por un lado, se ha construido una base de datos con los datos de entrenamiento originales de la competición denominada ***tfgdatosoriginales*** 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 ***tfgtest***. Aparte, se ha generado otro almacén de datos denominado ***tfgrealvalues*** 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.

4.2.2 Base de datos con los datos originales

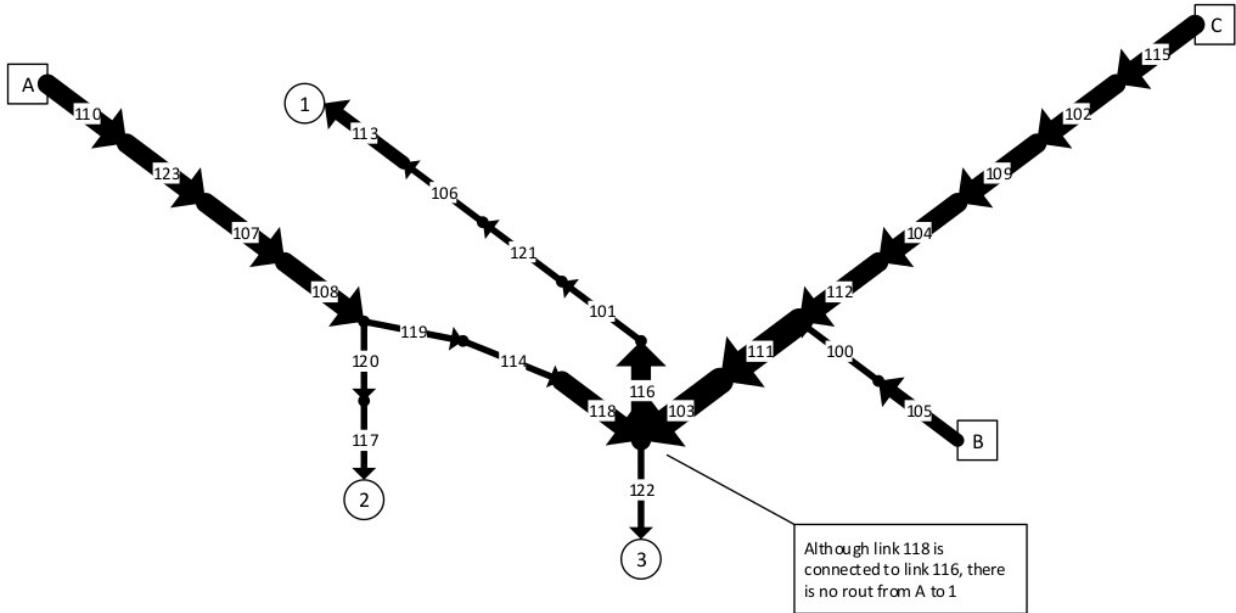
4.2.2.1 Tabla vehicle_routes

El esquema de la tabla original proporcionada por la competición es la que se expone en la **Tabla 8.1**. 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 4.5**)

4.2.2.2 Tabla road_links

El esquema de la tabla original proporcionada por la competición es la que se visualiza en la **Tabla 8.2**. Esta tabla contiene la **descripción de**

cada uno de los tramos que forman una ruta (ver **Figura 4.5**).



¿Señalar el propietario de la imagen?

4.2.2.3 Tabla vehicle_trajectories_training

El esquema de la tabla original proporcionada por la competición es la que se expone en la **Tabla 8.3**. Esta tabla contiene cada uno de los vehículos que han viajado en algún momento, entre el **19 de Julio** y el **17 de Octubre**, por alguna de las rutas establecidas en la tabla *vehicle_routes*. Para cada vehículo se establece el momento en el que entró en una ruta, el tiempo que estuvo ese vehículo en cada uno de los tramos que forman dicha ruta y el tiempo total de viaje que tardó en realizar esa ruta.

4.2.2.4 Tabla traffic_volume_tollgates_training

El esquema de la tabla original proporcionada por la competición es la que se visualiza en la **Tabla 8.4**. 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

El esquema de la tabla original proporcionada por la competición es la que se expone en la **Tabla 8.5**. 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

El esquema de la tabla original proporcionado por la competición es la que se visualiza en la **Tabla 8.6**. 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

El esquema de la tabla original proporcionado por la competición es la que se expone en la **Tabla 8.7**. 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

El nuevo esquema de la tabla es la que se visualiza en la **Tabla 8.8**. 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

El nuevo esquema de la tabla es la que se expone en la **Tabla 8.9**. 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

El nuevo esquema de la tabla es la que se visualiza en la **Tabla 8.10**. 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

El nuevo esquema de la tabla es la que se expone en la **Tabla 8.11**. En esta tabla se eliminaron las columnas *vehicle_model* y *vehicle_type* puesto que no son relevantes a la hora de realizar las predicciones pertinentes. Por otra parte, se modificaron los tipos de los atributos *tollgate_id* (a **smallint**), *direction* (a **smallint**) y *has_etc* (a **boolean**).

4.2.3.5 Tabla weather_data_modified

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

Por otro lado, se añadió una fila de una fecha que no existía en la tabla y que es necesaria para combinarse con otras tablas (día *10/10/16*).

4.2.3.6 Tabla travel_time_intersection_to_tollgate_modified

El nuevo esquema de la tabla es la que se visualiza en la **Tabla 8.12**. 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*

El nuevo esquema de la tabla es la que se expone en la **Tabla 8.13**. 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 fase de pruebas.

4.2.4.1 Tabla *travel_time_intersection_to_tollgate_test*

El esquema de la tabla es la que se visualiza en la **Tabla 8.14**. Para comprender el sentido de esta relación, es necesario observar la **Figura 4.2**. 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_test

El esquema de la tabla es la que se expone en la **Tabla 8.15**. 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 la que se visualiza en la **Tabla 8.16**. 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 4.2**).

4.2.4.4 Tabla tabla_resultado_traffic_volume

El esquema de la tabla es la que se expone en la **Tabla 8.17**. 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 4.2**).

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

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

4.2.5.1 Tabla *travel_time_intersection_to_tollgate_real*

El esquema de la tabla es la que se expone en la **Tabla 8.18**. Esta tabla contiene los datos reales del *tiempo promedio de viaje* de las ventanas de tiempo en las rutas y días que nos insta la competición a predecir en la fase de pruebas (**Figura 4.2**).

4.2.5.2 Tabla *traffic_volume_tollgates_real*

El esquema de la tabla es la que se visualiza en la **Tabla 8.19**. Esta tabla contiene los datos reales del *volumen de tráfico* de las ventanas de tiempo en las barreras de peaje y direcciones de conducción (entrada y salida) que nos insta la competición a predecir en la fase de pruebas (**Figura 4.2**).

4.3 Creación de gráficas para visualizar los datos almacenados

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

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

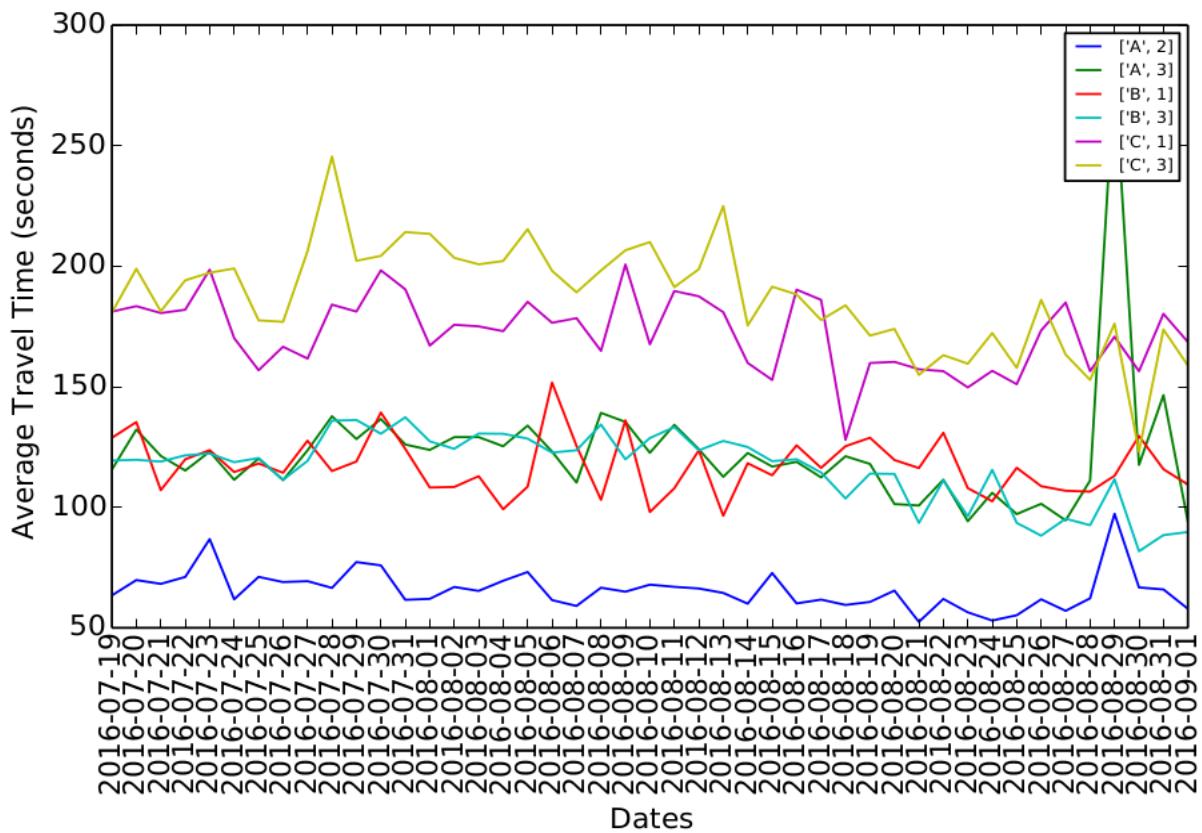


Figura 4.6: Tiempo promedio de viaje medio en cada uno de los días en las diferentes rutas

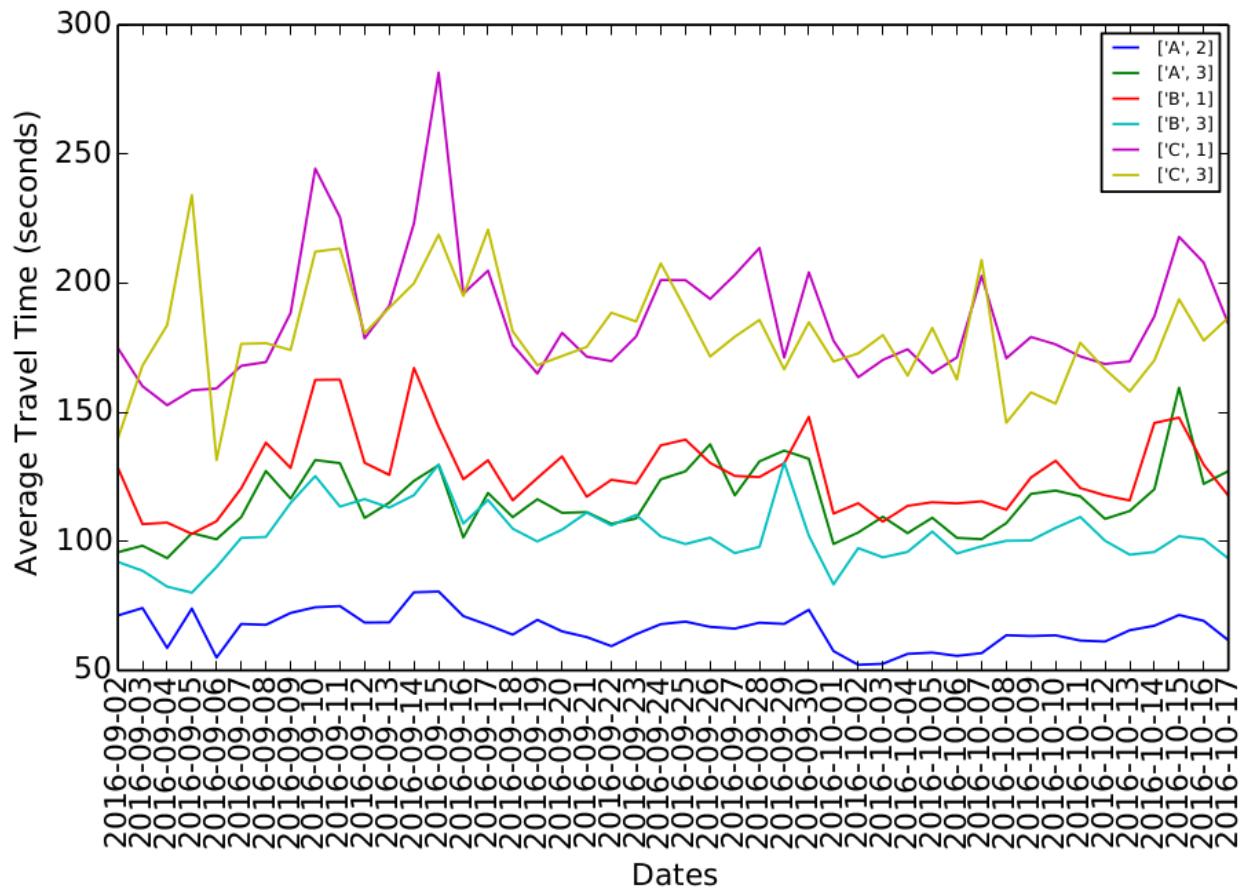
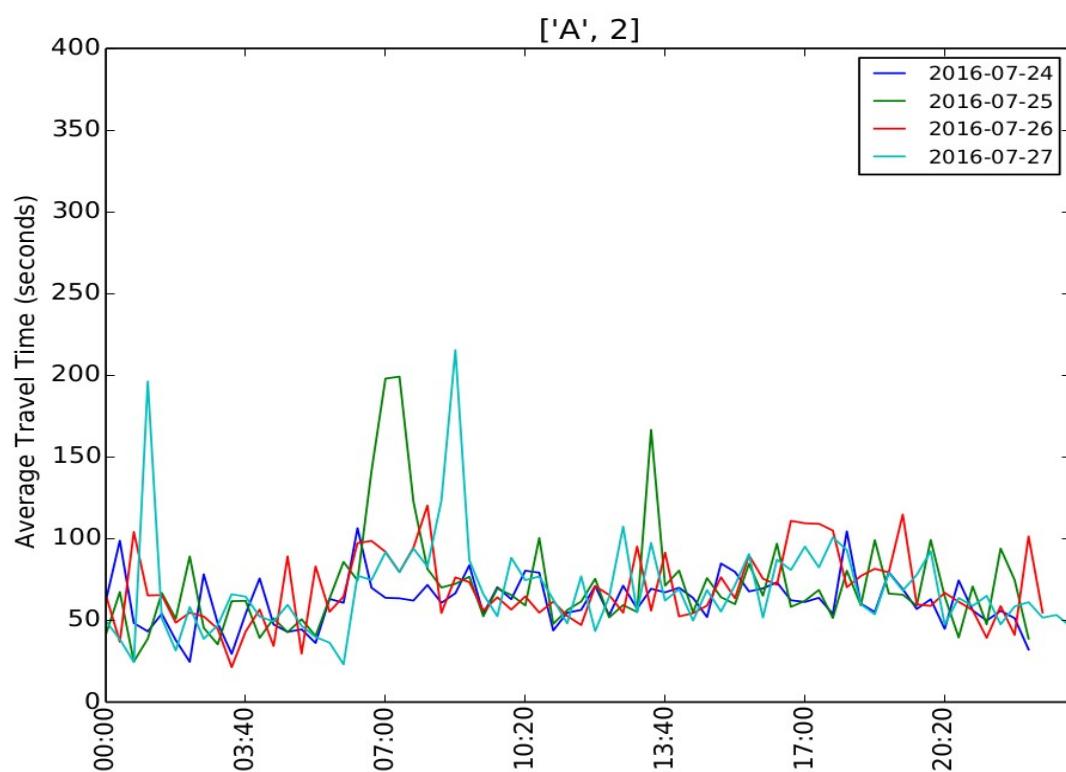
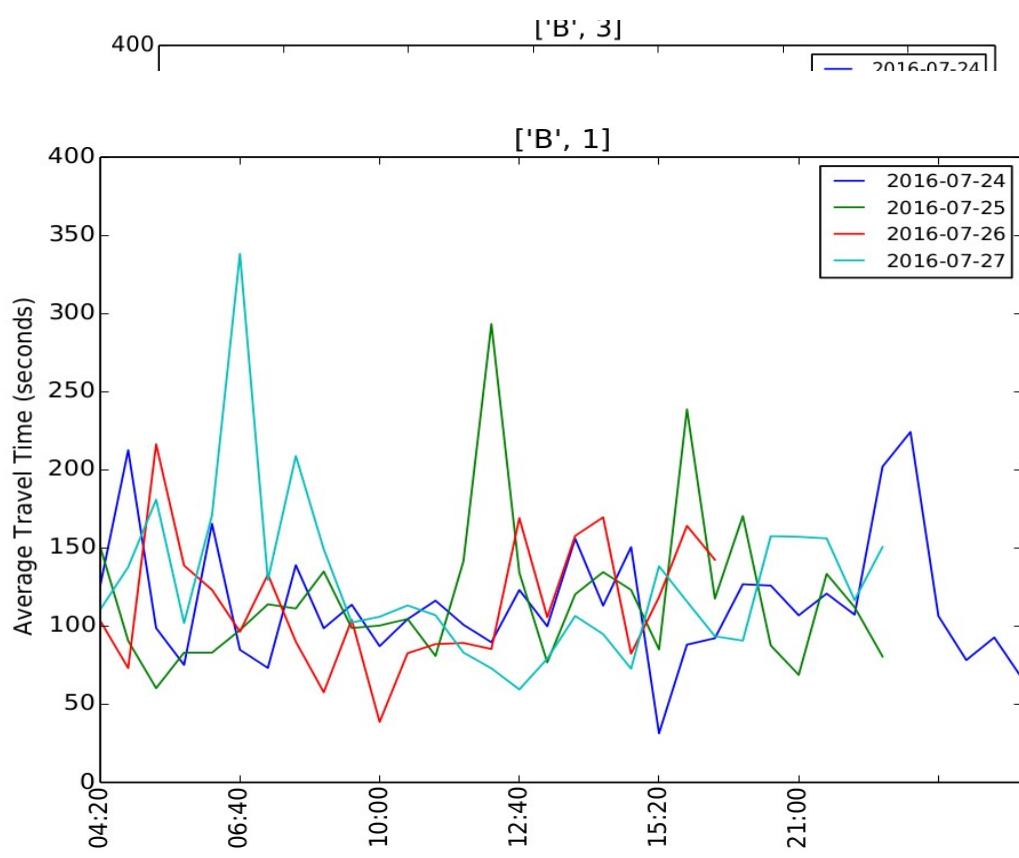
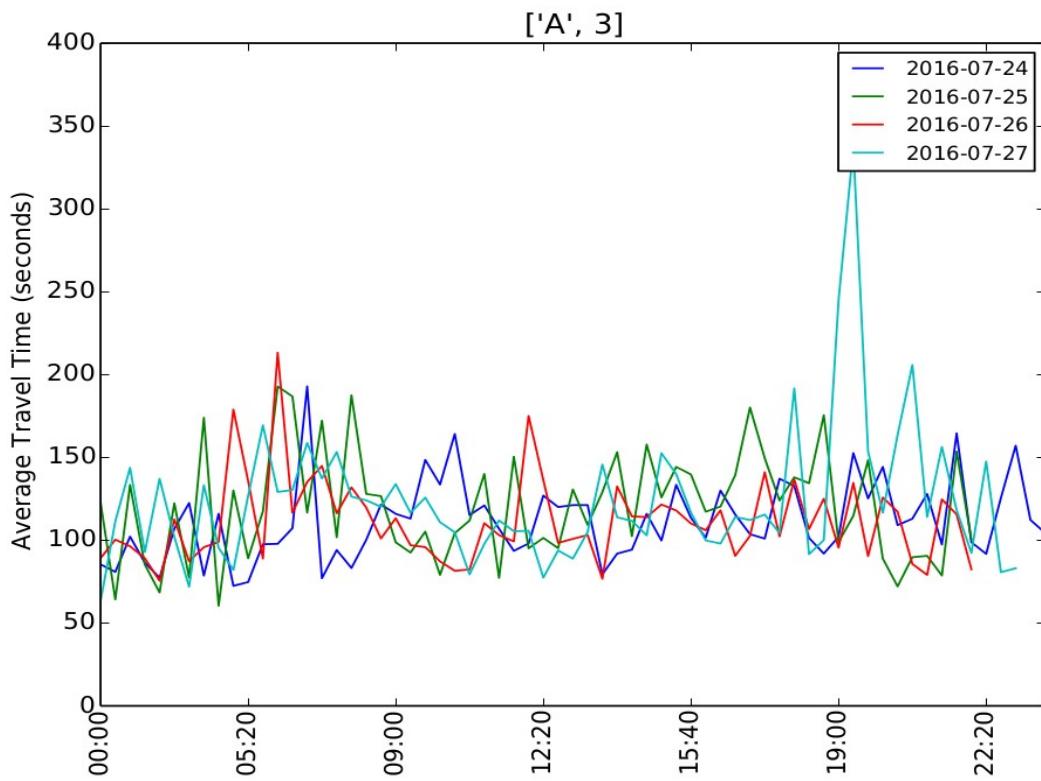


Figura 4.7: Continuación de la gráfica de la Figura 4.4

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, anchura y longitud de las carreteras que las forman (Figura 4.5).

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





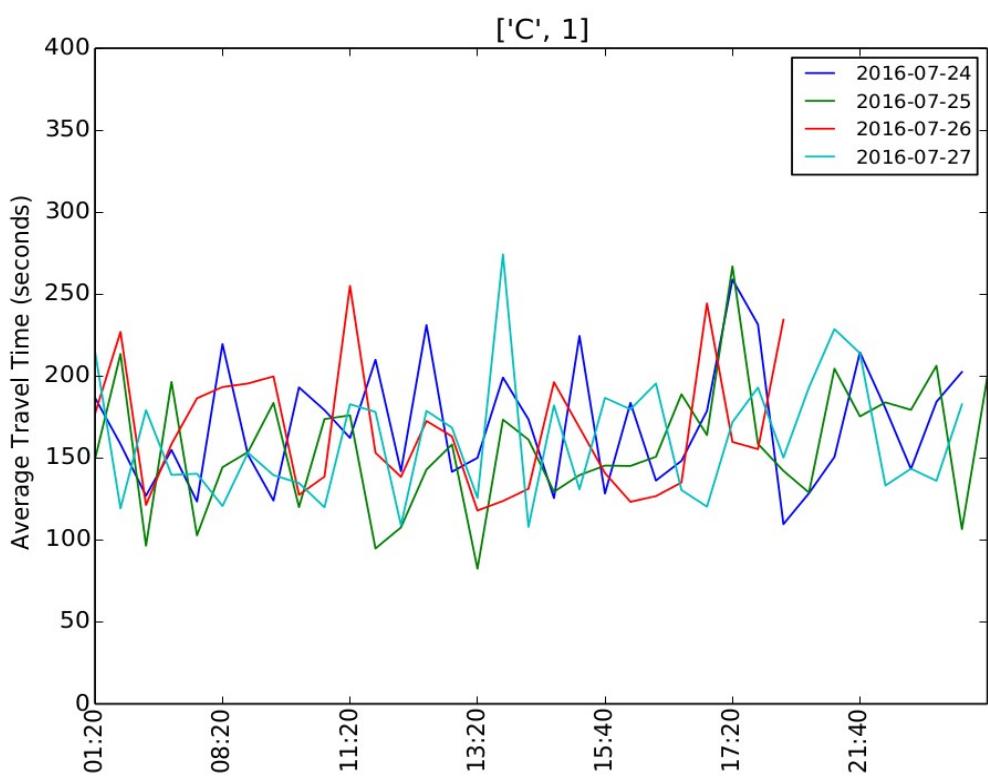
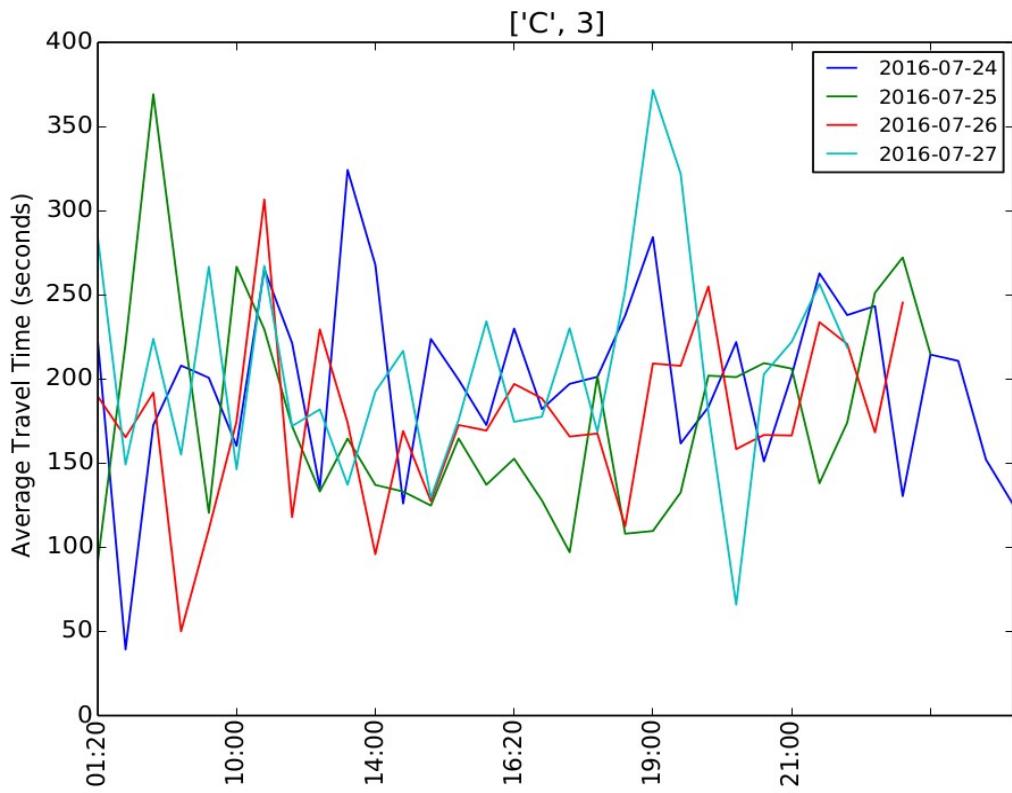


Figura 4.12: Tiempo promedio de viaje por horas en algunos días en la ruta C-1



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.

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

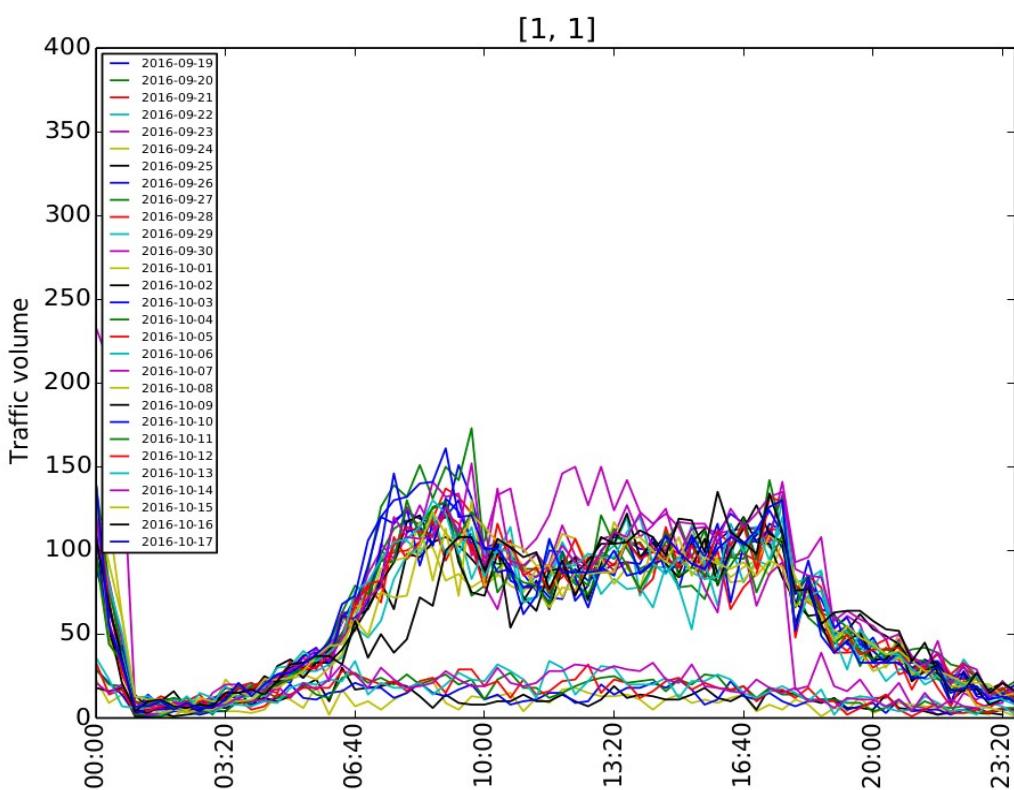
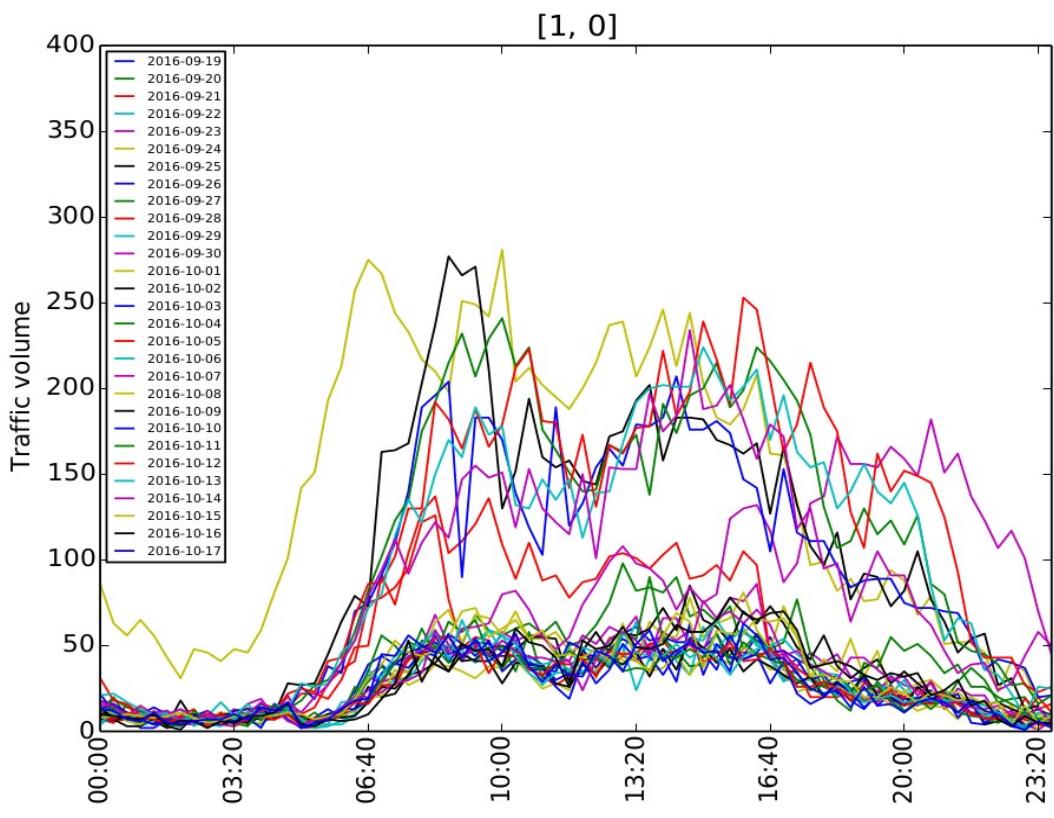
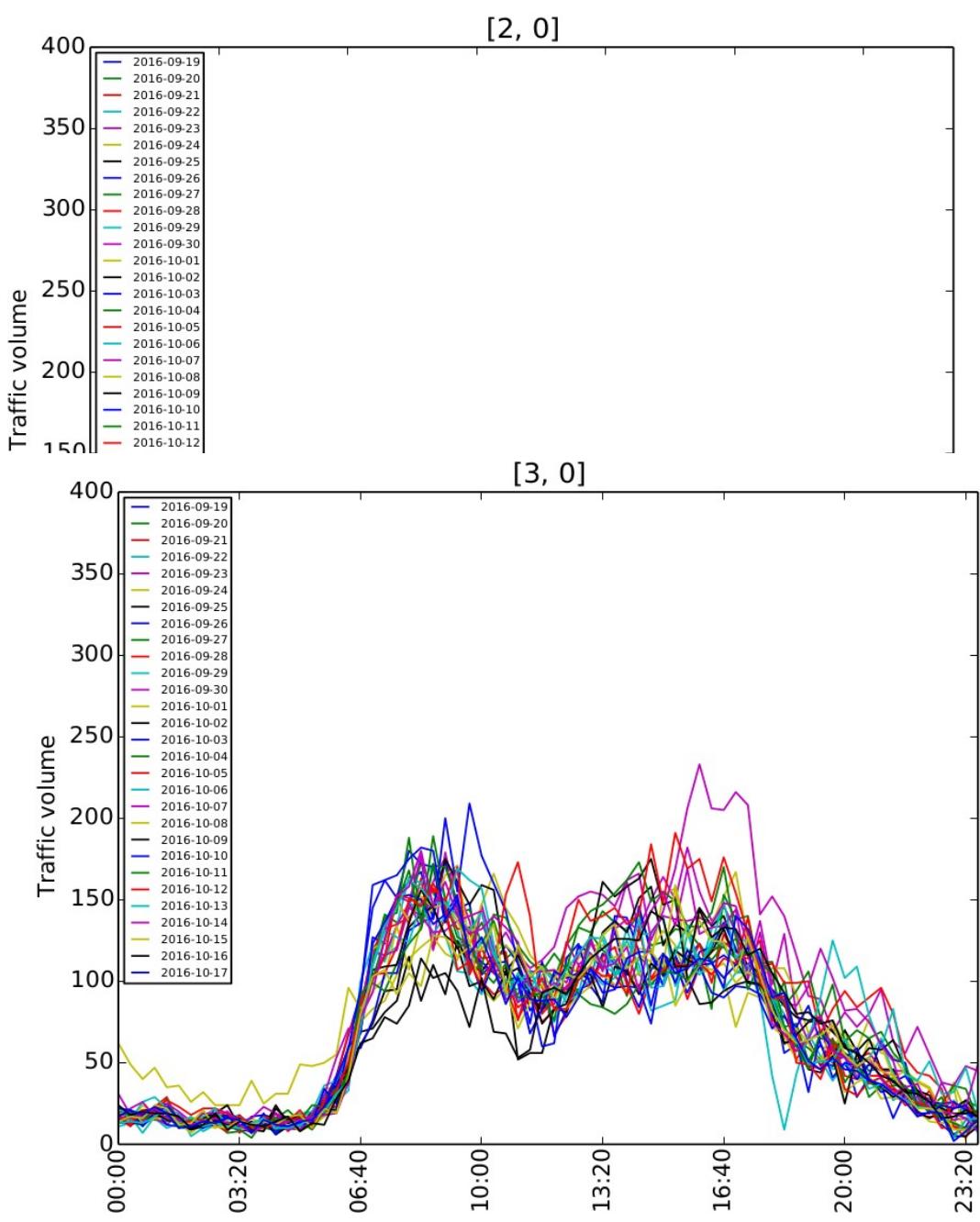
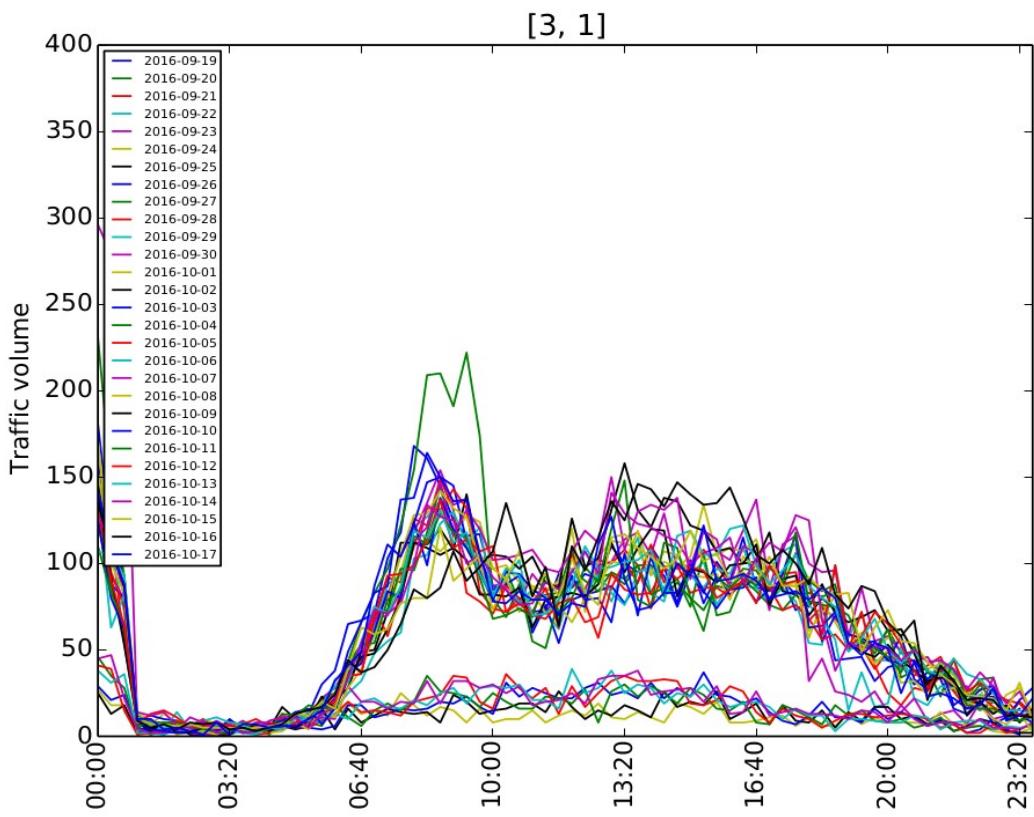


Figura 4.15: Barrera de peaje 1 en la dirección de salida





Como podemos visualizar en las gráficas anteriores, cada uno de los pares *barrera de peaje-dirección* posee un determinado patrón en cuanto al número de vehículos a lo largo de un día. No obstante, si nos fijamos detenidamente, se aprecia que en todas las gráficas la forma de las gráficas es la misma: en la primeras horas del día hay poco tráfico; en las primeras horas de la mañana

hay un pico en el número de vehículos y disminuye en las últimas horas del horario de mañana y, a medida que avanza la tarde, el tráfico va disminuyendo. Esto se corresponde con un día típico.

En algunas gráficas (como la que se visualiza en la **Figura 4.14**) se observa que en algunos días se sigue el patrón del número de vehículos normal pero no la forma de las gráficas de la mayoría de los días. Aquellos días que poseen dicho comportamiento se considera *ruido* y es necesario eliminarlo para que esos valores no influyan negativamente a la hora de realizar las predicciones del volumen de tráfico.

4.4 Predicciones del tiempo promedio de viaje

El primer objetivo a conseguir propuesto por la competición es estimar el *tiempo promedio de viaje* que se va a producir en una serie de intervalos de 20 minutos en unos días determinados (según la **Figura 4.2**). Para llevar a cabo esto, se ha procedido a desarrollar una serie de aproximaciones en las que se aplican diferentes técnicas de aprendizaje automático y se analiza y evalúa el comportamiento de los mismos sobre los datos contenidos en las bases de datos creadas para su tratamiento.

4.4.1 Primera aproximación

La primera aproximación de predicciones del tiempo promedio de viaje desarrollada parte de la *generación de unas vistas minables* sobre las bases de datos de los datos modificados y de los datos de prueba. Con el propósito de ejecutar el primer conjunto de predicciones, se llevó a cabo la creación de una nueva estructura de los datos para pasarla como entrada a los algoritmos a utilizar para estimar valores. 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 (º)
<i>wind_speed</i>	float	Velocidad del viento (m/s)
<i>temperature</i>	float	Temperatura(ºC)
<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 4.1: Vista minable creada sobre los datos a utilizar para entrenamiento y testeo de la primera aproximación de predicciones del tiempo promedio de viaje

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 4.19: 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 4.20: 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 llenar 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 4.21: 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 llenar 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 4.22: 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 llenar el valor de estas columnas. En el caso de que ocurriera alguna de estas dos circunstancias, esta columna adquiría el valor medio de los tiempos promedios de viaje de aquellas filas de la tabla cuyo intervalo de tiempo y el tipo de día (día de la semana) fuera el mismo que el de la fila en consideración. Por ejemplo, si la fila tomada en consideración corresponde a un **lunes** en el intervalo de tiempo **9:20-9:40** y la columna que corresponde al *tiempo promedio de viaje* 20 minutos antes de esa ventana de tiempo no se puede llenar 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 llenar los siguientes intervalos de tiempo de 20 minutos a predecir contiguos a la ventana de tiempo rellenada con la función anterior (por ejemplo, llenar 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 4.23: 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 llenar 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 4.24: 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 4.25: 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 4.25**. 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 4.1** 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_traveltimes = pd.DataFrame(rows, columns=colnames)
X_train = dataframe_traveltimes.iloc[:, 0:14]
y_train = dataframe_traveltimes.iloc[:, 14]

```

Figura 4.26: Obtención de los datos de entrenamiento para la ruta e intervalo en consideración

- A continuación se accede a la base de datos ***tfgtest*** para obtener la vista que contiene los datos de 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_traveltimes = pd.DataFrame(rows, columns=colnames)
X_test = dataframe_traveltimes.iloc[:, 0:14]

```

Figura 4.27: 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 4.28: Entrenamiento y predicción del tiempo promedio de viaje en los distintos días de predicción con el modelo XGBoost

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

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

Figura 4.29: 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_prediccciones_intervalos["XGBoost"] += y_test_sum
```

Figura 4.30: Acumulación de los errores correspondiente a los días a predecir de cada uno de los intervalos a estimar

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

Figura 4.31: 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_prediccciones_rutas[key] += errores_prediccciones_intervalos[key]/len(time_intervals)
```

Figura 4.32: Acumulación de los errores del segundo sumatorio de la fórmula del error MAPE

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

Figura 4.33: 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.

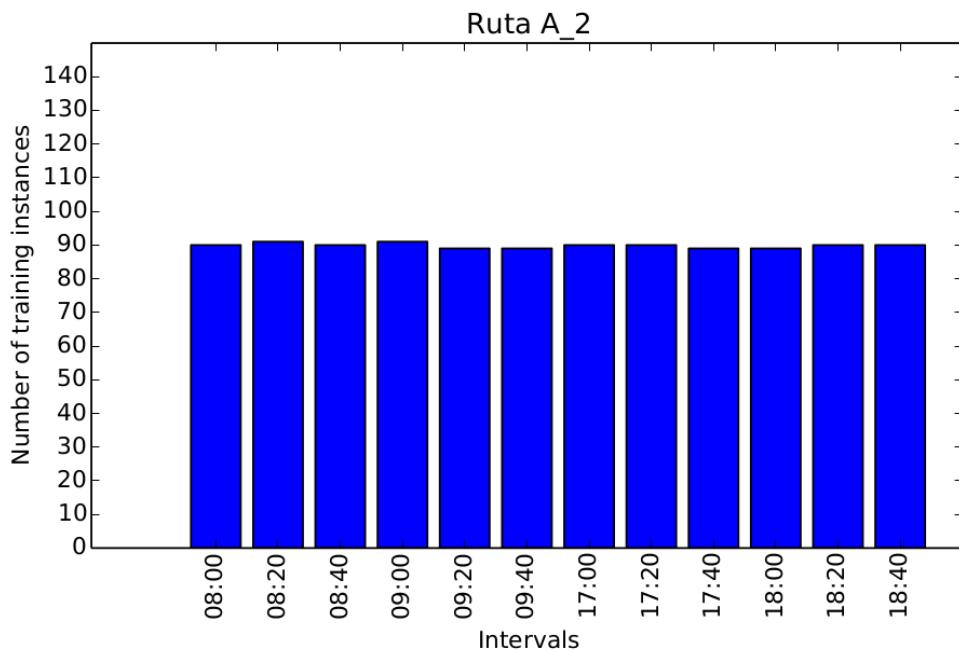
Tras llevar a cabo todo este proceso, se obtuvieron los resultados expuestos en la **Tabla 8.20**. Para cada una de las técnicas de minería de datos utilizadas para realizar un modelo predictivo sobre los datos se han dado los siguientes *errores MAPE*:

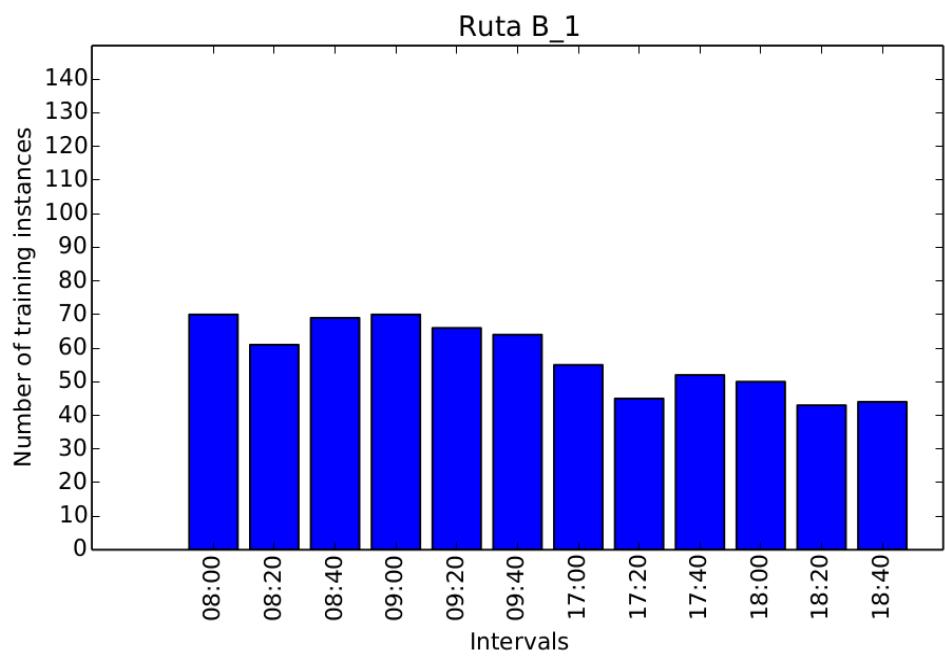
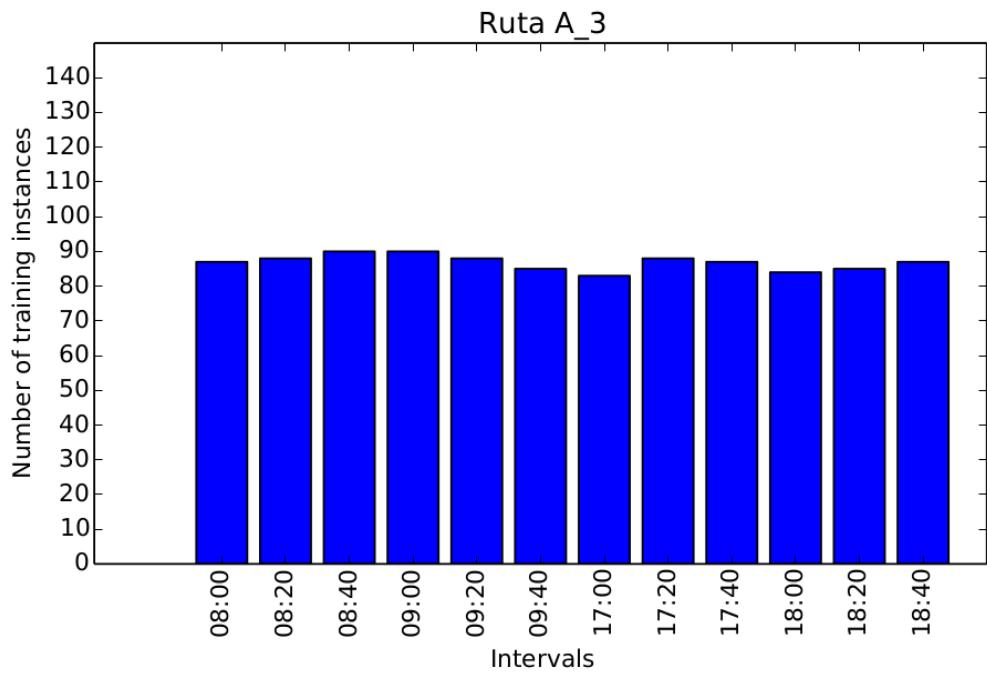
XGBoost	KNN	LightGBM	MLP	SVR	Linear Regression
0.2416	0.2107	0.2194	0.4639	0.2065	0.2613

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

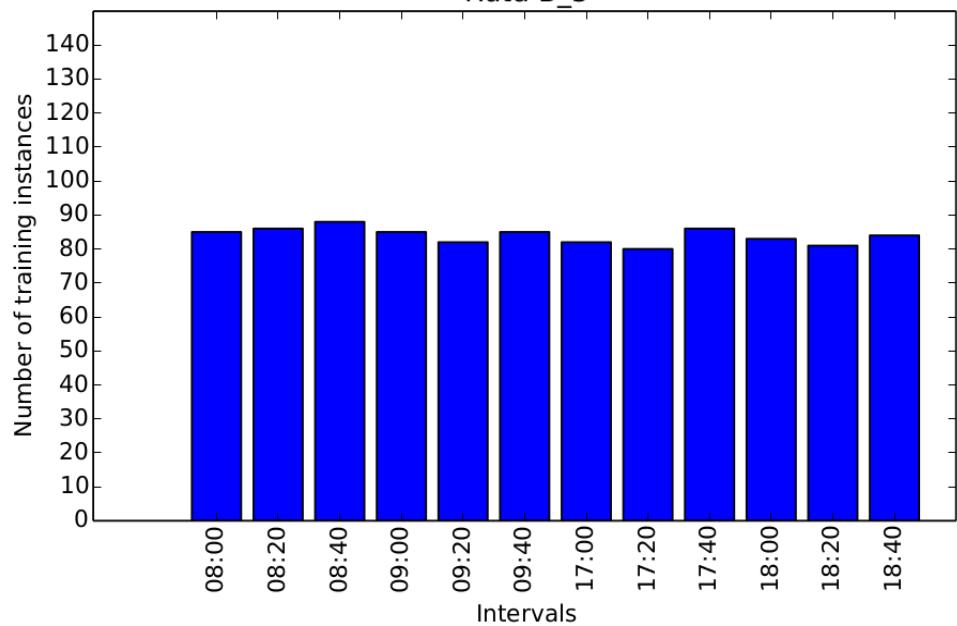
Como se puede apreciar en los valores de *error MAPE* anteriores, el algoritmo de aprendizaje automático que ha experimentado mejor rendimiento ha sido las *Máquinas de Soporte Vectorial aplicadas a la Regresión*, con un error de **0.2065**. No obstante, todos los algoritmos han estado prácticamente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

Con respecto a estos errores, cabe señalar que las técnicas de minería de datos utilizadas se ven afectadas, al igual que las *Redes Neuronales*, por el limitado número de instancias de entrenamiento que se les proporciona, a lo que se añade la falta de datos de tráfico de una serie de intervalos de tiempo en el conjunto de entrenamiento, lo que provoca que éste tenga menor tamaño. El número de instancias de entrenamiento empleadas se visualiza en las siguientes gráficas:

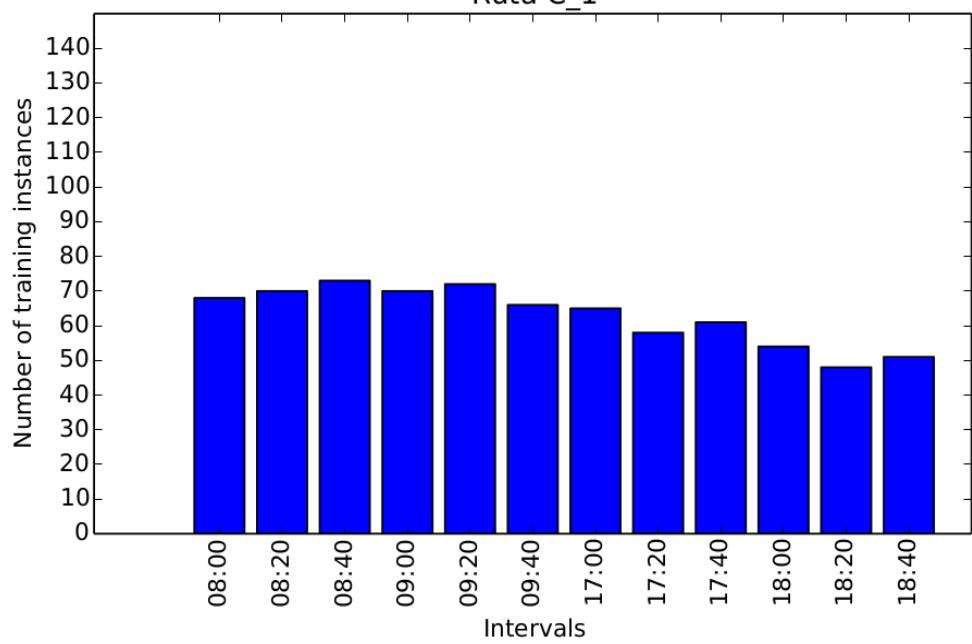


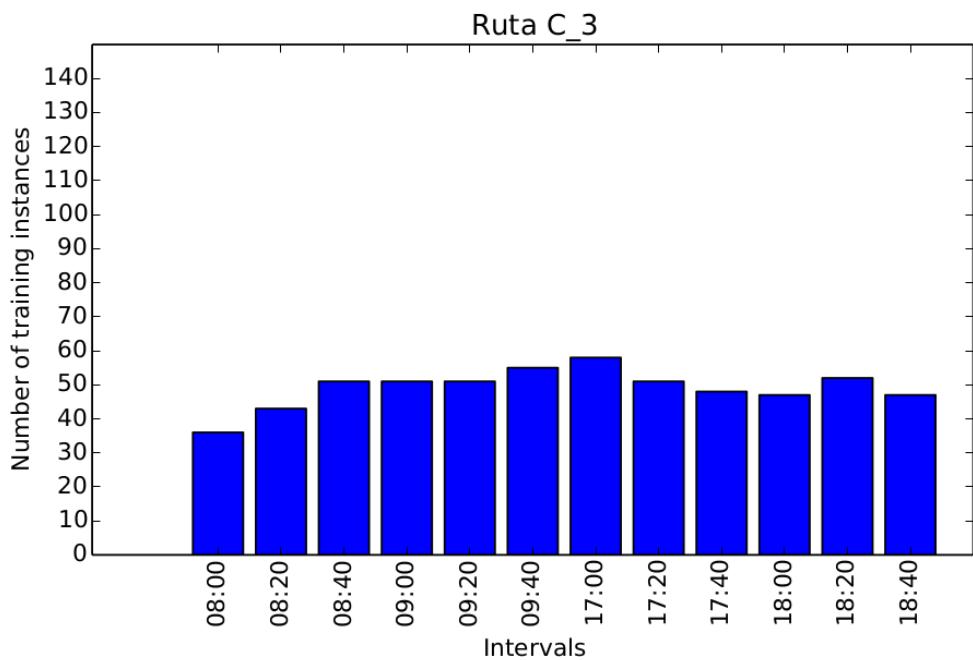


Ruta B_3



Ruta C_1





En todas las gráficas se visualiza que, por cada ruta e intervalo de tiempo a pronosticar, el número de instancias de entrenamiento no superan las 100 instancias. En el caso de que la competición nos hubiera suministrado una mayor número de datos de entrenamiento, los errores hubieran disminuido notablemente.

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 modelo *ARIMA* a construir, es imprescindible tener los valores reales del *tiempo promedio de viaje*. Para ello, el primer paso para efectuar las predicciones es obtener dichos valores reales de la base de datos oportuna:

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

Figura 4.40: 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 llenar 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 4.41: Cálculo del valor del tiempo promedio de viaje de una parte de aquellas rutas e intervalos de las que no disponemos datos

4.4.2.2 Realización de estimaciones

A continuación, para una *ruta* y *día* concretos, se procede a realizar las estimaciones en los intervalos de tiempo a predecir. Es decir, como se trata de predicciones de series temporales, se deben hacer por cada par *ruta-día* para estimar la evolución de la serie temporal en las ventanas de tiempo de dicho par. Para ello, por cada par *ruta-día*, se crean dos modelos *ARIMA*: uno para predecir los datos de la serie temporal en los intervalos de tiempo de 20 minutos incluidos en la ventana de tiempo **8:00-10:00** y otro modelo para estimar aquellos datos en los intervalos de tiempo de 20 minutos incluidos en la ventana de tiempo 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='tfgtest1' user='javisunami' host='localhost' password='javier123'")
except:
    print("I am unable to connect to the database")
cur = conn.cursor()
query = "select time_window[1], avg_travel_time from travel_time_intersection_to_tollgate_test1 where intersection_id = '" + str(route[0]) + "' AND tollgate_id = " + str(route[1]) +
+ " AND (time_window[1].time BETWEEN " + intervalo1 + ") AND (time_window[1].date = DATE '" + str(day) + "') order by time_window;"
cur.execute(query)
rows = cur.fetchall()
df2 = pd.DataFrame.from_records(rows, columns=['date', 'avg_travel_time'])
result_dataframe = pd.concat([df1_aux, df2])

```

Figura 4.42: 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(disp=0)
                forecast = model_fit.forecast(steps=6)[0]
                new_forecast=[]
                for element in rows2:
                    new_forecast.append(forecast[((datetime.datetime(2018,1,1,element[0].hour,element[0].minute, 0)-hora_de_referencia)/1200).seconds])
                mse = mean_squared_error(valores_reales, new_forecast)
                print("MSE : ", mse, " BEST_SCORE: ", best_score)
                if mse < best_score:
                    print("BEST_SCORE : ", best_score)
                    best_score, best_cfg = mse, orderr
            except:
                continue
print('Best ARIMAs MSE=%.3f' % (best_cfg, best_score))

```

Figura 4.43: 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

4.4.3 Tercera aproximación

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

4.4.3.1 Preparación de los datos

Para poder aplicar el método de la *ventana deslizante* sobre los datos temporales, es imprescindible añadir a dichos datos aquellos intervalos de tiempo de los que no se dispone ningún dato sobre el tiempo promedio de viaje en los mismos. Para ello, es fundamental aplicar la transformación vista en la **Figura 4.41**. Este código establece la media de los valores de la serie temporal cuya ventana de tiempo es igual al intervalo de tiempo que se quiere añadir a la serie temporal y donde el día de la semana de éstos sean el mismo para dicho intervalo.

A partir de esto, en primer lugar, para cada una de las rutas y días propuestos a predecir, se sigue el procedimiento previo para llenar los datos faltantes de la serie temporal (**Figura 4.41**). A continuación, concatenamos a los tiempos promedios de viaje de la serie temporal de una ruta determinada aquellos datos de las dos horas previas de un día a predecir, de manera que poseemos una serie temporal que alcanza hasta el primer intervalo de tiempo a estimar en ese día:

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

Figura 4.44: Concatenación de las dos horas previas a los intervalos de tiempo a predecir un día determinado con la serie temporal de una ruta

Tras este paso, se procede a utilizar el método de la *ventana deslizante*. Este método consiste en, dada una secuencia de números para un conjunto de datos de series de temporales, reestructurar los datos para que parezcan un *problema de aprendizaje supervisado*. Podemos llevar a cabo esto usando **instantes de tiempo anteriores** como *variables de entrada* y usar el **siguiente instante de tiempo** como variable de salida. Para ello, debemos elegir un determinado *retraso de tiempo* para establecer el número de instantes de tiempo hacia detrás que se van a emplear como atributos de entrada para los algoritmos de aprendizaje automático. Por ejemplo, si queremos que el *tiempo promedio de viaje* del siguiente instante de tiempo se prediga utilizando los valores de **5** instantes de tiempo anteriores, el retraso de tiempo es **5** :

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

Figura 4.45: Serie temporal a problema de aprendizaje supervisado

En la última línea de la **Figura 4.45** se eliminan las primeras 5 líneas debido a que, como hemos elegido un retraso de tiempo de 5 instantes de tiempo, los primeros 5 intervalos de tiempo van a tener datos faltantes puesto que no se disponen de más datos hacia atrás (por ejemplo el cuarto intervalo de tiempo no va a tener el *tiempo promedio de viaje* en el instante **t-5** puesto que no se proporciona dicho valor). Un ejemplo de la estructura se genera a partir del método de la *ventana deslizante* es el siguiente:

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

Figura 4.46: Estructura generada con el

método de la ventana deslizante

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

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

Para cada ruta, día e intervalo a estimar, se sigue el proceso comentado previamente. Después de obtener todos los valores a predecir, se obtiene el *error MAPE* correspondiente comparando los datos reales de la base de datos *tfgrealvalues* con los valores obtenidos.

Poner resultados con los distintos algoritmos

Comentar la razón por la que se dan errores MAPE más altos que en el *tiempo promedio de viaje* → Disponemos de más datos reales en cuanto al volumen de tráfico.

4.5 Predicciones del volumen de tráfico

El segundo objetivo a conseguir propuesto por la competición es predecir el **volumen de tráfico** que se va a producir en una serie de intervalos de 20 minutos en unos días determinados (según la **Figura 4.2**). Debido a que este parámetro del flujo de tráfico a predecir también evoluciona en el tiempo (al igual que el *tiempo promedio de viaje*), se ha procedido a

aplicar de forma similar las aproximaciones de predicción desarrolladas para la estimación del tiempo promedio de viaje puesto que en este caso también se trata con **series temporales**. Sin embargo, en este caso el objetivo no es estimar valores de volumen de tráfico por ruta, sino predecir tales valores por cada uno de los pares *barrera de peaje-dirección* contemplados en la topología de la red de carreteras dada por la competición. A continuación, se expone detalladamente el proceso de construcción de las diferentes técnicas de minería de datos escogidas para la tarea en cuestión.

4.5.1 Primera aproximación

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

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

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 (º)
<i>wind_speed</i>	float	Velocidad del viento (m/s)
<i>temperature</i>	float	Temperatura(ºC)
<i>rel_humidity</i>	float	Humedad relativa
<i>precipitation</i>	float	Precipitaciones (mm)
<i>volume</i>	float	Volumen de tráfico

Tabla 4.3: Vista minable creada sobre los datos a utilizar para entrenamiento y testeo de la primera aproximación de predicciones del volumen de tráfico

Como podemos apreciar, la tabla construida como formato de entrada tanto de los datos de entrenamiento como los de prueba para los diferentes algoritmos de aprendizaje automático se asemeja bastante a la **Tabla 4.1**. La única diferencia entre dichas tablas es la variable de tráfico a predecir.

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

4.5.2 Segunda aproximación

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

Por un lado, en los datos proporcionados por la competición sobre el *volumen de tráfico* en intervalos de tiempo de 20 minutos hay bastantes menos ventanas de tiempo que faltan en las tablas relacionadas con esta variable de tráfico. De esta forma, no ha sido necesario calcular tantas veces, para cada ventana de tiempo faltante, el *valor medio del volumen de tráfico* de aquellas filas de entrenamiento cuya ventana de tiempo coincide con el intervalo de tiempo cuyo valor de *volumen de tráfico* hay que llenar. No obstante, el inconveniente encontrado en dicha información es la cantidad suministrada de la misma. Mientras que para las tareas de predicción del *tiempo promedio de viaje* se proporcionaron datos de 4 meses (desde Julio hasta Octubre), para la estimación del *volumen de tráfico* se suministran datos de 2 meses (Septiembre y Octubre). Por lo tanto, la información de entrenamiento que se aporte al entrenamiento de los modelos de aprendizaje automático va a ser bastante menor en este caso y, de este modo, va a conllevar una dificultad añadida a la fase de predicción del *volumen de tráfico*.

Por otra parte, el patrón que se da en los datos proporcionados del *volumen de tráfico* se mantiene bastante más similar entre días que en la información suministrada para la predicción del *tiempo promedio de viaje*. Para apreciar esto, se expone la siguiente gráfica:

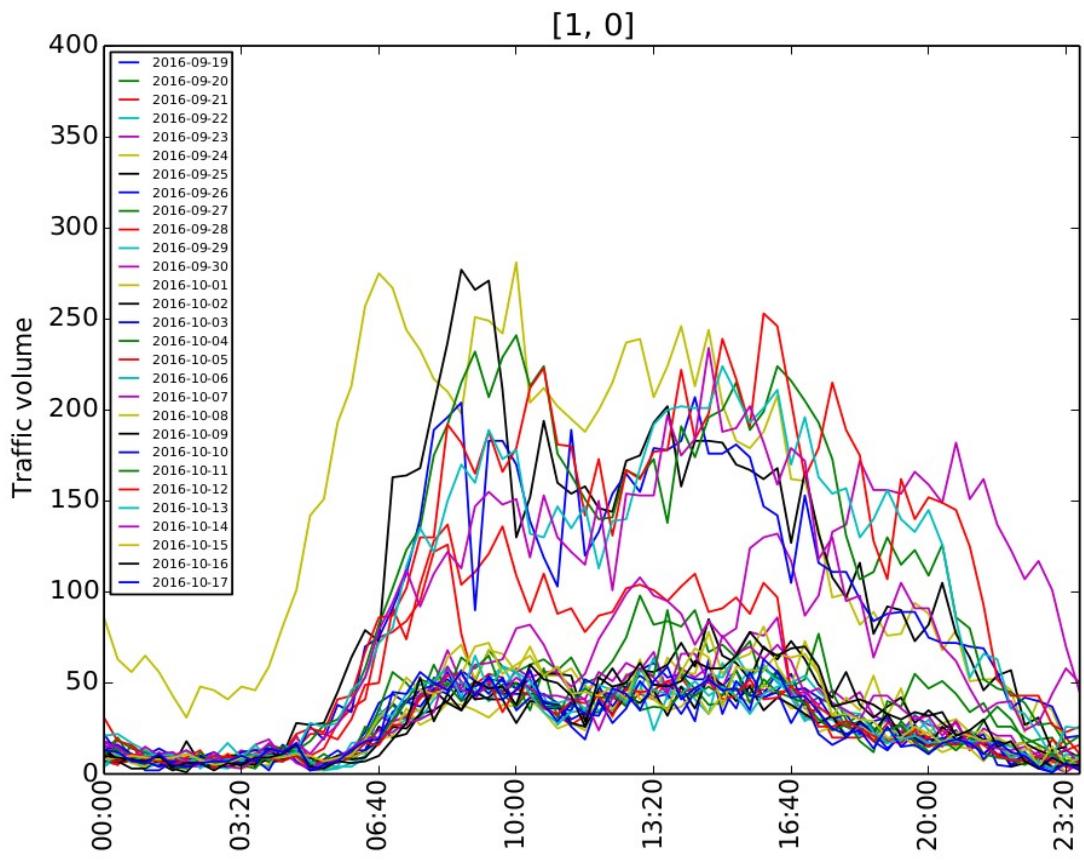


Figura 4.47: Ejemplo de gráfica del volumen de tráfico en una ruta determinada en los días de entrenamiento

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

```

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

```

Figura 4.48: Eliminación del ruido de los datos proporcionados para la predicción del volumen de tráfico

Para aquellos pares *barreras de peaje-dirección de conducción* que presentan ruido en sus gráficas, se determinan aquellos días que lo generan y se filtran en la estructuras pertinentes para su eliminación. En el caso de la gráfica visualizada en la **Figura 4.47**, tras el proceso de exclusión de los datos ruidosos ésta se queda de la siguiente manera:

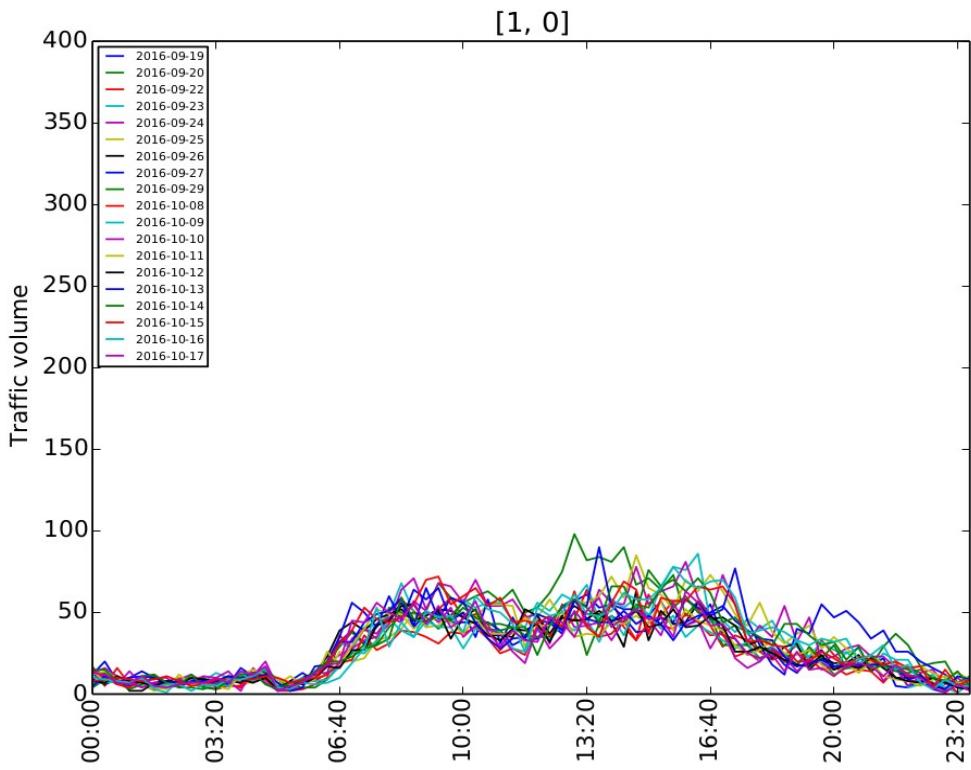


Figura 4.49: Gráfica resultante de la eliminación de ruido

Como se puede apreciar en la **Figura 4.49**, podemos ver con notoriedad la ausencia de ruido en las series temporales con respecto a la **Figura 4.47**. La eliminación de dichos valores que no siguen la tendencia de la mayoría de los datos en ese par *barrera de peaje-dirección de conducción* propicia que se puedan crear modelos más detallados y precisos para llevar a cabo las estimaciones oportunas del *volumen de tráfico*.

Tras realizar el filtrado de la información del *volumen de tráfico* en cada uno de los pares *barrera de peaje-dirección de conducción*, se proporciona la misma a las técnicas de minería de datos siguiendo las mismas pautas que para el *tiempo promedio de viaje* con el objetivo de que ejecuten las fases de entrenamiento y testeо para predecir el *volumen de tráfico*. A partir de esto, se alcanzaron los siguientes resultados:

Poner los resultados de las predicciones

4.5.3 Tercera aproximación

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

Previamente a la aplicación de dicho método sobre los datos de entrenamiento suministrados para la predicción del *volumen de tráfico*, se procede a ejecutar el filtrado de datos mencionado en la **Figura 4.48** para suprimir el ruido presente en las series temporales. A continuación de esto, se genera la estructura apropiada mediante la ejecución del método indicado con anterioridad (**Figura 4.45**), originando de este modo la distribución de los datos visualizados en la **Figura 4.46**. Seguidamente, se provee a las técnicas de minería de datos de la información indispensable para que puedan desempeñar su función de la mejor forma posible, que es estimar el *volumen de tráfico* en los pares *barrera de peaje-dirección de conducción* y ventanas de tiempo solicitadas a predecir.

Tras la ejecución de los algoritmos de aprendizaje automático pertinentes, se obtuvieron los valores predichos que se exponen a continuación:

Poner los resultados de las predicciones

Inconvenientes del proyecto

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

Capítulo 5

Conclusiones y líneas futuras

5.1.1 Conclusiones

En este proyecto se han aprendido los fundamentos de la minería de datos, así como el proceso de desarrollo de un trabajo que se requiere seguir en esta disciplina. Además, se ha adquirido experiencia en cuanto a utilización de distintas técnicas de aprendizaje automático con el fin de aplicarlas a tareas relacionadas con el flujo de tráfico en carreteras como son el *tiempo promedio de viaje* de los vehículos y el *volumen de tráfico*.

Por otra parte, nos hemos cultivado en la utilización del lenguaje de programación *Python* como herramienta para las labores de construcción de modelos de predicción y en el empleo del software *PostgreSQL* como almacén de datos para la información proporcionada por la competición *KDDCup2017*, indispensable para la ejecución de este trabajo. De igual manera se han adquirido una serie de conocimientos imprescindibles para organizar de forma adecuada el desarrollo de un proyecto con el fin de llevarlo a cabo de forma exitosa.

5.1.2 Líneas futuras

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

En relación a lo comentado previamente, se desea mejorar los resultados de predicción obtenidos con los datos de competición mediante la adquisición de una mayor cantidad y variedad de datos de tráfico por parte del Cabildo

de Tenerife. De la misma manera, se pretende realizar un estudio más exhaustivo de los modelos utilizados con el fin de optimizarlos y obtener unas estimaciones más precisas sobre el comportamiento futuro del flujo de tráfico de la isla.

Capítulo 6

Summary and Conclusions

6.1.1 Conclusions

In this project the fundamentals of data mining have been learned, as well as the process of developing a work that needs to be followed in this discipline. In addition, experience has been gained in the use of different automatic learning techniques to apply them to road traffic flow related tasks such as *average vehicle travel time* and *traffic volume*.

On the other hand, we have cultivated in the use of the *Python* programming language as a tool for the construction of prediction models and in the use of *PostgreSQL* software as a data warehouse for the information provided by the KDDCup2017 competition, indispensable for the execution of this work. Likewise, essential knowledge has been acquired to properly organize the development of a project in order to carry it out successfully.

6.1.2 Future lines

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

In relation to what was previously mentioned, we also wish to improve the prediction results obtained with the competition data by means of the acquisition of a greater quantity and variety of traffic data by the town council of Tenerife. Similarly, the aim is to carry out a more exhaustive study of the models used in order to optimise them and obtain more precise estimates of the future behaviour of the island's traffic flow.

Capítulo 7

Presupuesto

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

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

Tabla 7.1: Presupuesto del material utilizado en el proyecto

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

Personal	Horas de trabajo	Coste/Hora	Total
1	300	35€	10500€

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

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

Capítulo 8

Anexo

8.1.1 Tablas con los datos de la competición

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 8.1: Tabla *vehicle_routes*

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 8.2: Tabla *road_links*

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 8.3: Tabla *vehicle_trajectories_training*

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 (comprendido 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 8.4: Tabla *traffic_volume_tollgates_training*

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 (º)
<i>wind_speed</i>	float	Velocidad del viento (m/s)
<i>temperature</i>	float	Temperatura(ºC)
<i>rel_humidity</i>	float	Humedad relativa
<i>precipitation</i>	float	Precipitaciones (mm)

Tabla 8.5: Tabla *weather_data*

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 8.6: Tabla *travel_time_intersection_to_tollgate*

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 8.7: Tabla *traffic_volume_tollgates*

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 8.8: Tabla *road_links_modified*

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 8.9: Tabla *vehicle_routes_modified*

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 8.10: Tabla *vehicle_trajectories_training_modified*

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

Tabla 8.11: Tabla *traffic_volume_tollgates_training_modified*

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 8.12: Tabla *travel_time_intersection_to_tollgate_modified*

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 8.13: Tabla *traffic_volume_tollgates_modified*

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 8.14: Tabla *travel_time_intersection_to_tollgate_test*

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 8.15: Tabla *traffic_volume_tollgates_test*

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 8.16: Tabla *tabla_resultado_average_travel_time*

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 8.17: Tabla *tabla_resultado_traffic_volume*

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 8.18: Tabla *travel_time_intersection_to_tollgate_real*

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 8.19: Tabla *traffic_volume_tollgates_real*

8.1.2 Tablas con los valores predichos

8.1.2.1 Primera aproximación de predicciones del tiempo promedio de viaje

Ruta	Día	Intervalo de tiempo	Valor real	XGBoost	Linear Regression	LightGBM	MLP	SVR	KNN
(A, 2)	18	(08:00, 08:20)	77.77	93.0074	72.2685	134.232	68.72	73.818	84.448
(A, 2)	19	(08:00, 08:20)	91.72	69.7644	66.1888	51.8285	73.5283	71.975	77.41
(A, 2)	20	(08:00, 08:20)	87.24	81.2721	82.2975	124.664	72.0673	77.2487	77.086
(A, 2)	21	(08:00, 08:20)	58.58	71.9845	77.063	56.8453	68.1516	76.1447	75.259
(A, 2)	22	(08:00, 08:20)	71.43	72.3178	82.1531	128.097	75.1749	75.286	82.9216
(A, 2)	23	(08:00, 08:20)	56.62	63.6303	74.3089	69.7743	72.619	75.359	77.7616
(A, 2)	24	(08:00, 08:20)	70.59	75.8253	52.0667	51.5617	55.6193	77.382	81.6518
(A, 2)	18	(08:20, 08:40)	70.3	82.9185	72.0392	92.2364	68.5178	75.5633	91.364
(A, 2)	19	(08:20, 08:40)	92.28	74.4696	78.0542	63.9466	79.4859	77.1307	88.6971
(A, 2)	20	(08:20, 08:40)	75.48	101.792	84.9278	79.8342	77.4325	77.06	88.8504
(A, 2)	21	(08:20, 08:40)	71.5	64.7404	71.3097	78.0078	65.9472	74.9523	82.3624
(A, 2)	22	(08:20, 08:40)	56.06	79.4782	71.5301	83.566	80.6545	74.5963	89.39

(A, 2)	23	(08:20, 08:40)	65.06	85.6301	63.1095	75.7347	74.9676	76.5013	89.8354
(A, 2)	24	(08:20, 08:40)	85.32	75.6038	55.0834	81.0165	60.7429	71.577	80.7136
(A, 2)	18	(08:40, 09:00)	77.68	75.6815	70.5982	138.97	72.6705	77.085	81.5761
(A, 2)	19	(08:40, 09:00)	103.94	73.93	73.8903	87.7205	76.3549	75.0523	92.6942
(A, 2)	20	(08:40, 09:00)	79.6	77.241	80.4382	115.298	81.292	74.7753	82.0577
(A, 2)	21	(08:40, 09:00)	55.92	79.9283	79.7314	73.6058	70.1258	77.01	85.3607
(A, 2)	22	(08:40, 09:00)	62.2	79.4521	88.8981	43.4749	84.2019	75.4753	81.9897
(A, 2)	23	(08:40, 09:00)	65.96	83.8568	91.5279	131.423	80.3376	78.0403	84.6704
(A, 2)	24	(08:40, 09:00)	63.33	86.1083	78.6046	126.306	65.1164	75.9123	82.6216
(A, 2)	18	(09:00, 09:20)	79.28	76.4886	78.9958	60.0614	70.0719	71.2857	78.7413
(A, 2)	19	(09:00, 09:20)	79.27	83.7714	94.5467	70.6341	77.17	71.7243	79.4027
(A, 2)	20	(09:00, 09:20)	103.35	80.0893	82.612	65.4769	75.7784	72.4573	76.8074
(A, 2)	21	(09:00, 09:20)	99.53	81.3782	73.3968	82.635	70.1962	73.2093	78.1232
(A, 2)	22	(09:00, 09:20)	79.54	98.415	81.8228	76.3854	75.2145	75.5327	83.1109

(A, 2)	23	(09:00, 09:20)	83.79	92.911	106.192	114.31	84.0593	78.802	82.1518
(A, 2)	24	(09:00, 09:20)	79.18	68.2674	83.474	49.2868	72.7638	72.536	78.6331
(A, 2)	18	(09:20, 09:40)	83.27	69.4068	59.9007	99.375	66.1601	71.6537	76.4759
(A, 2)	19	(09:20, 09:40)	81.22	70.1282	59.3698	62.9984	71.7677	70.6667	79.6768
(A, 2)	20	(09:20, 09:40)	97.05	69.7126	77.1015	70.0211	72.5997	74.2033	78.7385
(A, 2)	21	(09:20, 09:40)	83.67	75.7819	81.4334	52.4511	66.8326	72.3407	78.5279
(A, 2)	22	(09:20, 09:40)	64.81	82.1512	91.6771	58.2664	75.9102	74.6007	82.1423
(A, 2)	23	(09:20, 09:40)	64.03	78.2555	92.3906	40.8982	78.0992	80.0063	79.1898
(A, 2)	24	(09:20, 09:40)	76.7	62.5986	79.0316	89.1118	69.1427	73.115	77.1081
(A, 2)	18	(09:40, 10:00)	77.67	67.5763	59.7718	63.603	70.9221	71.3743	76.8859
(A, 2)	19	(09:40, 10:00)	79.82	75.9953	59.3075	36.9366	70.1561	71.478	77.3945
(A, 2)	20	(09:40, 10:00)	67.95	74.5428	69.0225	42.7576	72.494	70.4443	77.3988
(A, 2)	21	(09:40, 10:00)	73.3	64.6382	64.2039	59.5692	69.6676	70.601	77.4275

(A, 2)	22	(09:40, 10:00)	70.94	71.8642	78.478	29.6478	74.3811	71.0943	77.1864
(A, 2)	23	(09:40, 10:00)	66.64	78.3418	80.365	56.9965	75.9127	73.5307	78.0749
(A, 2)	24	(09:40, 10:00)	51.01	68.4564	72.7427	60.0783	70.4184	72.6367	75.8094
(A, 2)	18	(17:00, 17:20)	75	80.1229	71.3428	30.6993	65.9659	72.5043	74.68
(A, 2)	19	(17:00, 17:20)	60	66.5746	82.0334	55.6239	69.3688	71.5113	71.502
(A, 2)	20	(17:00, 17:20)	96.52	66.5517	71.3656	36.7568	67.1024	72.5217	72.1338
(A, 2)	21	(17:00, 17:20)	101.05	71.2576	65.9389	47.5294	67.3713	74.7413	73.6703
(A, 2)	22	(17:00, 17:20)	71.32	65.4525	67.4352	79.933	66.1282	74.9297	70.5607
(A, 2)	23	(17:00, 17:20)	94.08	56.6072	59.4648	97.9337	63.7928	69.097	67.3282
(A, 2)	24	(17:00, 17:20)	85.03	75.2909	61.9568	94.1973	65.1384	69.9587	75.3093
(A, 2)	18	(17:20, 17:40)	63.3	74.3798	74.2968	59.8798	69.984	71.0733	72.6759
(A, 2)	19	(17:20, 17:40)	56.98	69.2088	63.5774	55.3178	59.6637	68.9233	71.1543
(A, 2)	20	(17:20, 17:40)	54.93	72.3008	65.154	59.0433	60.761	68.8733	69.1083

(A, 2)	21	(17:20, 17:40)	82.62	76.7271	74.724	58.4398	76.4578	70.9927	73.4365
(A, 2)	22	(17:20, 17:40)	63.61	59.928	64.4829	122.867	61.6535	68.623	68.6224
(A, 2)	23	(17:20, 17:40)	67.87	80.508	86.3829	159.463	71.9302	71.0333	73.9082
(A, 2)	24	(17:20, 17:40)	68.98	66.5347	65.8161	92.3087	65.5082	68.3067	72.2716
(A, 2)	18	(17:40, 18:00)	68.39	73.4686	66.0826	16.3292	69.1395	69.0743	71.6265
(A, 2)	19	(17:40, 18:00)	66.91	81.8478	70.2883	88.905	67.9525	69.191	77.7532
(A, 2)	20	(17:40, 18:00)	82.28	71.9132	65.2089	57.2404	66.5571	65.427	72.0121
(A, 2)	21	(17:40, 18:00)	62	80.6525	82.7713	64.4027	76.4947	69.1463	72.7722
(A, 2)	22	(17:40, 18:00)	77.17	56.5046	59.4484	58.2928	63.5029	66.8377	68.9833
(A, 2)	23	(17:40, 18:00)	63.09	76.1125	51.1393	118.156	65.4916	70.0357	75.3644
(A, 2)	24	(17:40, 18:00)	53.84	63.4301	57.9893	65.235	59.7201	66.765	69.842
(A, 2)	18	(18:00, 18:20)	88.83	80.1705	74.7409	56.1605	75.0355	70.055	71.8325
(A, 2)	19	(18:00, 18:20)	63.8	81.6973	62.0856	66.2139	71.0047	70.5897	71.6947

(A, 2)	20	(18:00, 18:20)	75.25	74.3205	65.2218	72.9452	68.3008	69.4493	69.9512
(A, 2)	21	(18:00, 18:20)	76.75	95.2763	79.3472	72.1243	77.8059	66.6783	76.177
(A, 2)	22	(18:00, 18:20)	69.61	61.8462	64.2634	81.0293	65.1702	65.7497	66.4879
(A, 2)	23	(18:00, 18:20)	63.97	74.1325	58.4609	144.015	59.4313	65.5723	70.0966
(A, 2)	24	(18:00, 18:20)	65.13	70.6689	67.2805	45.3164	68.2248	68.453	69.9551
(A, 2)	18	(18:20, 18:40)	59.33	76.292	68.7405	9.8815	66.0484	64.557	65.9122
(A, 2)	19	(18:20, 18:40)	55.74	61.2274	58.9649	49.4649	56.1904	62.406	64.9245
(A, 2)	20	(18:20, 18:40)	50.34	58.0277	57.4886	69.7578	61.2182	59.1787	63.6285
(A, 2)	21	(18:20, 18:40)	81.04	65.1726	62.4663	29.7943	63.4944	61.4217	65.1195
(A, 2)	22	(18:20, 18:40)	66.35	60.725	53.9249	101.78	57.0856	61.3817	63.6812
(A, 2)	23	(18:20, 18:40)	60.32	69.777	54.4751	155.454	65.5933	62.2253	67.2518
(A, 2)	24	(18:20, 18:40)	61.35	63.8874	60.4818	90.506	64.028	62.9487	66.1046
(A, 2)	18	(18:40, 19:00)	52.23	64.9748	66.8952	32.3282	63.4677	67.0717	67.9033

(A, 2)	19	(18:40, 19:00)	84.02	91.58	68.6502	63.531	64.3062	68.641	71.4746
(A, 2)	20	(18:40, 19:00)	57.25	67.1395	67.2292	91.5772	62.3044	65.788	67.5058
(A, 2)	21	(18:40, 19:00)	70.41	67.3278	64.429	-3.62554	62.561	65.4017	67.1893
(A, 2)	22	(18:40, 19:00)	58.64	61.8651	65.3331	66.8593	62.18	65.4257	65.5621
(A, 2)	23	(18:40, 19:00)	69.44	69.5977	69.6512	98.2194	65.1616	66.7963	69.0765
(A, 2)	24	(18:40, 19:00)	62.85	82.6076	67.6535	42.4129	64.9151	66.3743	70.8982
(A, 3)	18	(08:00, 08:20)	197.5	291.855	231.614	318.488	161.526	159.044	189.233
(A, 3)	19	(08:00, 08:20)	319.67	205.394	202.479	231.264	155.259	155.051	155.716
(A, 3)	20	(08:00, 08:20)	183.76	302.198	211.393	241.012	158.78	164.673	184.293
(A, 3)	21	(08:00, 08:20)	258.88	177.741	175.829	183.105	146.074	142.49	145.544
(A, 3)	22	(08:00, 08:20)	120.94	189.981	208.171	183.495	161.216	159.327	155.407
(A, 3)	23	(08:00, 08:20)	91.06	141.934	164.638	136.542	134.3	126.978	146.468
(A, 3)	24	(08:00, 08:20)	265.38	131.967	118.773	147.948	108.708	124.96	139.405

(A, 3)	18	(08:20, 08:40)	166.9	183.319	167.888	151.626	149.492	158.264	183.462
(A, 3)	19	(08:20, 08:40)	328.02	169.332	207.276	51.9857	151.438	148.5	161.282
(A, 3)	20	(08:20, 08:40)	242.81	172.746	183.443	159.943	152.336	155.931	172.357
(A, 3)	21	(08:20, 08:40)	270.64	170.681	206.639	242.312	141.917	143.249	160.087
(A, 3)	22	(08:20, 08:40)	113.81	173.947	243.149	87.1638	159.62	159.958	169.761
(A, 3)	23	(08:20, 08:40)	90.74	167.513	221.465	121.925	126.987	129.145	168.374
(A, 3)	24	(08:20, 08:40)	234.99	112.633	150.747	140.363	107.346	113.913	149.608
(A, 3)	18	(08:40, 09:00)	253.82	139.208	148.971	117.208	161.143	155.903	154.185
(A, 3)	19	(08:40, 09:00)	326.7	224.403	180.339	199.634	171.908	158.474	172.022
(A, 3)	20	(08:40, 09:00)	224.49	139.856	167.982	145.656	168.322	161.922	157.77
(A, 3)	21	(08:40, 09:00)	240.81	144.472	173.585	141.714	149.967	151.565	152.041
(A, 3)	22	(08:40, 09:00)	195.13	206.096	191.271	167.251	176.117	173.639	166.209
(A, 3)	23	(08:40, 09:00)	111.3	157.271	154.857	242.902	141.034	135.953	154.927

(A, 3)	24	(08:40, 09:00)	181.75	128.842	112.432	25.8763	110.386	122.865	147.435
(A, 3)	18	(09:00, 09:20)	240.16	162.318	116.908	109.943	155.41	139.005	171.387
(A, 3)	19	(09:00, 09:20)	339.07	176.425	183.652	45.9504	175.379	148.836	169.826
(A, 3)	20	(09:00, 09:20)	201.63	156.908	145.565	194.803	163.141	150.171	166.283
(A, 3)	21	(09:00, 09:20)	237.97	157.567	166.357	172.614	154.087	149.691	162.768
(A, 3)	22	(09:00, 09:20)	116.25	176.157	201.068	275.237	184.014	180.487	172.939
(A, 3)	23	(09:00, 09:20)	116.89	158.793	176.468	133.586	143.969	142.647	169.833
(A, 3)	24	(09:00, 09:20)	262.43	167.779	98.8867	67.916	118.936	128.289	167.92
(A, 3)	18	(09:20, 09:40)	179.19	148.566	139.668	128.942	138.02	137.866	160.319
(A, 3)	19	(09:20, 09:40)	316.84	121.755	153.053	275.386	156.015	150.153	161.551
(A, 3)	20	(09:20, 09:40)	263.23	160.379	150.219	173.426	146.317	144.387	160.936
(A, 3)	21	(09:20, 09:40)	250.89	147.527	150.124	156.72	143.8	146.593	157.885
(A, 3)	22	(09:20, 09:40)	124.06	210.572	192.901	153.865	165.598	162.626	182.488

(A, 3)	23	(09:20, 09:40)	103.09	145.639	123.838	145.737	138.41	148.109	156.96
(A, 3)	24	(09:20, 09:40)	146.56	105.849	128.336	167.043	120.96	135.673	152.154
(A, 3)	18	(09:40, 10:00)	136.98	137.844	123.29	122.804	124.014	136.736	146.097
(A, 3)	19	(09:40, 10:00)	199.89	138.173	139.7	139.299	130.782	133.186	141.972
(A, 3)	20	(09:40, 10:00)	182.37	136.849	136.599	136.751	127.921	133.051	141.12
(A, 3)	21	(09:40, 10:00)	194.2	122.508	128.341	75.1586	122.927	129.987	138.319
(A, 3)	22	(09:40, 10:00)	147.83	135.375	162.022	13.9397	134.458	135.343	140.482
(A, 3)	23	(09:40, 10:00)	95.62	133.005	126.289	105.701	122.947	140.931	140.967
(A, 3)	24	(09:40, 10:00)	92.95	131.834	135.32	200.887	123.439	137.491	139.578
(A, 3)	18	(17:00, 17:20)	136.34	138.079	131.586	133.746	127.199	120.656	126.213
(A, 3)	19	(17:00, 17:20)	152.27	115.566	114.251	121.953	116.677	121.038	122.495
(A, 3)	20	(17:00, 17:20)	139.66	138.906	123.026	90.407	120.913	122.908	122.361
(A, 3)	21	(17:00, 17:20)	135.93	117.16	124.309	162.584	119.703	122.251	123.494

(A, 3)	22	(17:00, 17:20)	110.48	107.828	104.876	110.461	116.534	119.306	119.571
(A, 3)	23	(17:00, 17:20)	112.65	141.875	126.453	229.265	116.83	119.095	122.955
(A, 3)	24	(17:00, 17:20)	121.97	123.816	176.717	125.191	135.931	133.13	136.571
(A, 3)	18	(17:20, 17:40)	143.7	130	119.059	86.3041	123.963	121.009	120.973
(A, 3)	19	(17:20, 17:40)	123.18	128.083	113.566	116.025	119.639	116.744	122.337
(A, 3)	20	(17:20, 17:40)	126.32	137.137	115.22	96.7663	120.376	117.069	122.124
(A, 3)	21	(17:20, 17:40)	121.93	120.733	96.5357	80.8611	111.187	116.671	116.873
(A, 3)	22	(17:20, 17:40)	113.84	110.484	117.283	148.399	116.02	117.795	117.09
(A, 3)	23	(17:20, 17:40)	107.82	109.554	107.351	130.426	109.499	111.004	116.792
(A, 3)	24	(17:20, 17:40)	124.4	124.403	105.718	146.099	125.982	118.742	120.911
(A, 3)	18	(17:40, 18:00)	100.13	118.669	125.812	73.9263	120.046	116.329	117.522
(A, 3)	19	(17:40, 18:00)	112.64	113.886	114.543	118.383	113.872	112.042	116.076
(A, 3)	20	(17:40, 18:00)	124.92	113.901	112.571	76.7809	117.798	115.908	117.025

(A, 3)	21	(17:40, 18:00)	107.05	112.175	112.222	89.7078	114.491	111.528	116.379
(A, 3)	22	(17:40, 18:00)	124.66	108.211	105.249	125.719	113.388	113.323	115.032
(A, 3)	23	(17:40, 18:00)	112.35	122.736	119	150.392	120.075	113.25	118.246
(A, 3)	24	(17:40, 18:00)	95.07	121.687	125.76	103.046	119.038	117.392	116.035
(A, 3)	18	(18:00, 18:20)	122.91	131.075	123.094	102.156	125.722	120.141	121.041
(A, 3)	19	(18:00, 18:20)	110.19	113.296	115.437	101.283	116.879	120.852	118.441
(A, 3)	20	(18:00, 18:20)	139.16	116.007	118.879	114.69	118.584	118.751	122.903
(A, 3)	21	(18:00, 18:20)	113.41	111.798	114.754	42.7171	118.51	117.865	119.211
(A, 3)	22	(18:00, 18:20)	88.7	111.739	114.634	173.72	116.122	121.045	121.076
(A, 3)	23	(18:00, 18:20)	98.51	115.979	85.1482	192.248	120.588	120.132	121.371
(A, 3)	24	(18:00, 18:20)	109.61	106.498	105.408	131.123	115.245	117.729	117.995
(A, 3)	18	(18:20, 18:40)	112.75	115.63	104.564	104.942	112.43	114.734	121.237
(A, 3)	19	(18:20, 18:40)	100.43	116.035	118.866	134.41	116.211	118.029	122

(A, 3)	20	(18:20, 18:40)	128.43	117.779	122.86	79.3299	114.393	116.982	120.47
(A, 3)	21	(18:20, 18:40)	123.13	127.98	102.674	114.315	111.366	115.347	122.084
(A, 3)	22	(18:20, 18:40)	103.83	109.878	132.245	71.7087	113.087	115.211	119.896
(A, 3)	23	(18:20, 18:40)	103.75	125.851	99.4124	264.054	110.575	116.521	125.447
(A, 3)	24	(18:20, 18:40)	109.73	115.21	113.744	86.5431	111.212	114.285	121.893
(A, 3)	18	(18:40, 19:00)	120.58	139.18	129.106	130.345	119.117	134.503	140.979
(A, 3)	19	(18:40, 19:00)	150.64	140.805	150.793	196.002	119.762	133.836	141.603
(A, 3)	20	(18:40, 19:00)	95.71	139.81	135.538	93.2642	114.333	132.87	125.368
(A, 3)	21	(18:40, 19:00)	107.53	134.164	147.039	90.1793	116.585	121.37	133.077
(A, 3)	22	(18:40, 19:00)	170.18	142.646	163.19	34.648	116.676	135.122	124.604
(A, 3)	23	(18:40, 19:00)	102.92	134.013	136.29	143.801	118.372	136.19	124.1
(A, 3)	24	(18:40, 19:00)	165.64	140.613	125.384	143.674	113.916	135.969	133.084
(B, 1)	18	(08:00, 08:20)	88.71	126.486	155.396	162.599	142.76	121.22	127.793

(B, 1)	19	(08:00, 08:20)	92.31	125.185	120.377	206.53	119.214	117.116	130.812
(B, 1)	20	(08:00, 08:20)	108.92	119.039	88.5429	124.957	116.876	118.068	124.48
(B, 1)	21	(08:00, 08:20)	126.64	115.089	89.4541	122.793	104.413	113.82	121.091
(B, 1)	22	(08:00, 08:20)	105.17	141.664	138.146	150.161	136.741	120.758	126.832
(B, 1)	23	(08:00, 08:20)	139.86	129.246	106.429	81.6432	118.416	119.334	131.65
(B, 1)	24	(08:00, 08:20)	-	-	-	-	-	-	-
(B, 1)	18	(08:20, 08:40)	136.07	116.626	130.244	139.45	139.542	121.382	119.661
(B, 1)	19	(08:20, 08:40)	135.66	129.121	110.84	134.708	119.33	118.016	123.546
(B, 1)	20	(08:20, 08:40)	138.17	106.266	121.129	111.225	117.343	116.022	113.956
(B, 1)	21	(08:20, 08:40)	163	126.151	125.791	131.757	121.49	116.712	123.414
(B, 1)	22	(08:20, 08:40)	116.39	114.654	130.235	149.139	131.857	121.291	116.285
(B, 1)	23	(08:20, 08:40)	111.58	125.334	122.893	97.8212	124.222	115.158	121.687
(B, 1)	24	(08:20, 08:40)	-	-	-	-	-	-	-

(B, 1)	18	(08:40, 09:00)	108.32	124.846	112.161	102.136	129.305	117.344	120.479
(B, 1)	19	(08:40, 09:00)	153.98	131.738	111.604	281.524	117.269	122.624	123.901
(B, 1)	20	(08:40, 09:00)	83.41	148.472	129.757	138.309	117.145	122.014	124.542
(B, 1)	21	(08:40, 09:00)	118.02	138.1	134.419	148.611	118.386	120.765	122.192
(B, 1)	22	(08:40, 09:00)	107.39	110.312	123.951	88.8589	124.485	115.578	116.564
(B, 1)	23	(08:40, 09:00)	-	-	-	-	-	-	-
(B, 1)	24	(08:40, 09:00)	112.63	113.768	112.776	102.525	125.473	118.715	120.534
(B, 1)	18	(09:00, 09:20)	109.72	110.084	104.547	162.201	125.898	116.054	121.816
(B, 1)	19	(09:00, 09:20)	117.8	128.134	106.678	170.961	122.218	120.781	131.135
(B, 1)	20	(09:00, 09:20)	119.25	121.975	121.98	91.081	119.66	117.916	125.988
(B, 1)	21	(09:00, 09:20)	123.06	118.533	126.196	122.045	124.392	117.833	124.863
(B, 1)	22	(09:00, 09:20)	130.69	115.556	97.5342	200.4	118.342	122.513	125.86
(B, 1)	23	(09:00, 09:20)	189.01	168.914	178.048	124.277	139.004	127.232	145.069

(B, 1)	24	(09:00, 09:20)	95.66	130.524	112.704	169.499	129.088	116.32	130.454
(B, 1)	18	(09:20, 09:40)	121.11	121.785	82.1244	140.977	103.71	128.636	149.734
(B, 1)	19	(09:20, 09:40)	137.92	104.115	105.142	76.54	105.947	128.908	114.874
(B, 1)	20	(09:20, 09:40)	105.32	114.832	133.308	80.8062	111.127	124.713	116.454
(B, 1)	21	(09:20, 09:40)	115.15	106.977	132.191	126.647	108.048	125.427	116.476
(B, 1)	22	(09:20, 09:40)	144.58	99.5241	111.452	169.649	105.899	122.84	114.453
(B, 1)	23	(09:20, 09:40)	-	-	-	-	-	-	-
(B, 1)	24	(09:20, 09:40)	124.09	105.982	95.4845	182.384	112.105	120.266	115.813
(B, 1)	18	(09:40, 10:00)	103.39	109.587	100.944	216.742	117.395	126.666	128.1
(B, 1)	19	(09:40, 10:00)	142.04	119.149	118.548	132.664	122.224	120.719	126.532
(B, 1)	20	(09:40, 10:00)	132.5	111.688	123.48	133.175	118.411	127.359	121.279
(B, 1)	21	(09:40, 10:00)	97.77	105.753	121.04	125.674	118.482	123.532	123.167
(B, 1)	22	(09:40, 10:00)	126.19	119.712	124.005	121.192	118.552	126.793	123.751

(B, 1)	23	(09:40, 10:00)	107.28	109.158	96.3853	203.472	108.539	123.389	118.849
(B, 1)	24	(09:40, 10:00)	111.86	117.954	107.331	263.481	115.599	125.156	122.454
(B, 1)	18	(17:00, 17:20)	120	132.68	149.243	98.9372	137.108	122.341	129.217
(B, 1)	19	(17:00, 17:20)	120.2	199.833	134.703	168.887	125.06	121.692	146.698
(B, 1)	20	(17:00, 17:20)	87.61	146.66	124.919	54.0861	123.369	124.994	133.706
(B, 1)	21	(17:00, 17:20)	174.46	155.71	146.173	161.133	130.306	121.085	137.739
(B, 1)	22	(17:00, 17:20)	160.13	111.83	109.42	54.195	118.696	116.944	124.076
(B, 1)	23	(17:00, 17:20)	84.84	113.953	111.635	159.425	135.743	123.606	126.522
(B, 1)	24	(17:00, 17:20)	114.14	144.606	143.576	207.576	141.221	127.585	130.568
(B, 1)	18	(17:20, 17:40)	150.77	104.426	97.375	45.0382	114.052	129.517	128.973
(B, 1)	19	(17:20, 17:40)	-	-	-	-	-	-	-
(B, 1)	20	(17:20, 17:40)	173.74	109.173	126.06	105.616	115.961	132.341	126.109
(B, 1)	21	(17:20, 17:40)	221.76	156.812	151.074	118.067	129.013	137.055	143.353

(B, 1)	22	(17:20, 17:40)	155.09	109.948	106.576	149.144	109.824	130.542	126.392
(B, 1)	23	(17:20, 17:40)	94.28	155.757	125.279	131.779	120.911	138.988	143.361
(B, 1)	24	(17:20, 17:40)	128.66	104.311	127.821	159.698	121.099	140.074	125.53
(B, 1)	18	(17:40, 18:00)	154.23	150.887	209.189	277.53	150.27	152.01	151.257
(B, 1)	19	(17:40, 18:00)	115.66	129.094	161.024	169.487	142.159	145.733	142.896
(B, 1)	20	(17:40, 18:00)	140.4	162.412	151.395	109.332	131.213	138.457	141.429
(B, 1)	21	(17:40, 18:00)	-	-	-	-	-	-	-
(B, 1)	22	(17:40, 18:00)	111.85	159.473	143.658	175.994	126.634	139.25	141.779
(B, 1)	23	(17:40, 18:00)	136.49	125.981	62.2564	234.696	121.387	137.853	144.459
(B, 1)	24	(17:40, 18:00)	123.62	196.032	207.096	211.417	153.469	150.848	151.344
(B, 1)	18	(18:00, 18:20)	131.32	142.85	130.112	130.274	122.331	147.661	136.062
(B, 1)	19	(18:00, 18:20)	141.59	114.026	167.746	92.1014	131.064	145.98	139.113
(B, 1)	20	(18:00, 18:20)	21.7	127.688	127.14	35.809	123.406	141.779	136.44

(B, 1)	21	(18:00, 18:20)	392.65	121.179	154.099	211.574	123.316	141.499	137.398
(B, 1)	22	(18:00, 18:20)	115.5	107.341	136.285	34.6509	124.222	144.286	134.811
(B, 1)	23	(18:00, 18:20)	125.39	102.492	60.6311	233.06	112.924	141.574	127.974
(B, 1)	24	(18:00, 18:20)	146.24	144.639	135.668	134.534	124.563	145.722	140.325
(B, 1)	18	(18:20, 18:40)	131.49	328.421	160.386	50.4194	140.426	159.041	181.428
(B, 1)	19	(18:20, 18:40)	-	-	-	-	-	-	-
(B, 1)	20	(18:20, 18:40)	143.75	226.945	171.363	167.533	128.281	145.138	158.298
(B, 1)	21	(18:20, 18:40)	-	-	-	-	-	-	-
(B, 1)	22	(18:20, 18:40)	175.15	165.419	177.682	206.171	124.72	152.66	144.919
(B, 1)	23	(18:20, 18:40)	104.97	174.514	142.346	357.05	116.731	149.85	141.689
(B, 1)	24	(18:20, 18:40)	-	-	-	-	-	-	-
(B, 1)	18	(18:40, 19:00)	175.32	151.712	244.602	121.063	145.538	157.494	165.602
(B, 1)	19	(18:40, 19:00)	-	-	-	-	-	-	-

(B, 1)	20	(18:40, 19:00)	137.51	139.734	161.621	52.382	130.1	135.997	143.293
(B, 1)	21	(18:40, 19:00)	188.54	183.125	218.739	192.56	143.484	150.85	158.676
(B, 1)	22	(18:40, 19:00)	-	-	-	-	-	-	-
(B, 1)	23	(18:40, 19:00)	124.76	159.172	202.976	326.082	132.714	143.274	156.722
(B, 1)	24	(18:40, 19:00)	125.79	119.562	144.515	134.745	120.952	136.607	144.71
(B, 3)	18	(08:00, 08:20)	126.4	61.9316	114.063	145.269	98.034	101.348	98.4081
(B, 3)	19	(08:00, 08:20)	70.3	121.802	118.325	84.5358	104.272	107.138	112.796
(B, 3)	20	(08:00, 08:20)	93.88	108.484	114.669	135.845	108.856	109.485	112.043
(B, 3)	21	(08:00, 08:20)	85.64	103.89	98.0985	219.16	110.659	106.682	110.244
(B, 3)	22	(08:00, 08:20)	103.23	57.0606	101.358	68.3638	105.353	106.501	95.5829
(B, 3)	23	(08:00, 08:20)	106.67	104.343	92.8237	254.975	108.754	107.806	110.202
(B, 3)	24	(08:00, 08:20)	120.81	102.858	104.994	65.2731	110.637	108.864	114.094
(B, 3)	18	(08:20, 08:40)	115.52	97.54	95.7422	81.2144	97.4289	101.797	112.747

(B, 3)	19	(08:20, 08:40)	145.63	109.151	78.2478	126.03	93.6685	103.391	115.872
(B, 3)	20	(08:20, 08:40)	87.78	93.6714	104.188	75.0034	105.953	104.15	115.129
(B, 3)	21	(08:20, 08:40)	99.17	148.21	112.525	186.583	102.373	106.887	123.834
(B, 3)	22	(08:20, 08:40)	111.49	92.911	101.253	74.8812	105.47	104.031	111.395
(B, 3)	23	(08:20, 08:40)	85.55	107.189	108.357	190.504	106.184	109.404	115.263
(B, 3)	24	(08:20, 08:40)	98.33	100.911	92.6686	151.88	94.4336	104.095	107.699
(B, 3)	18	(08:40, 09:00)	98.86	133.903	116.45	128.141	106.639	105.877	122.272
(B, 3)	19	(08:40, 09:00)	120	114.203	140.962	256.689	121.406	111.941	123.012
(B, 3)	20	(08:40, 09:00)	103.01	112.133	130.85	172.186	118.865	112.867	120.965
(B, 3)	21	(08:40, 09:00)	102.09	104.076	116.485	142.26	102.784	108.38	117.354
(B, 3)	22	(08:40, 09:00)	65.55	113.648	129.217	135.298	118.83	108.27	119.089
(B, 3)	23	(08:40, 09:00)	91.14	105.629	123.544	99.318	109.868	109.134	118.685
(B, 3)	24	(08:40, 09:00)	107.49	92.5865	106.571	78.0131	102.888	103.525	113.971

(B, 3)	18	(09:00, 09:20)	131.15	115.463	114.249	127.921	101.195	107.054	126.985
(B, 3)	19	(09:00, 09:20)	129.17	111.667	132.755	89.5855	110.957	109.531	127.299
(B, 3)	20	(09:00, 09:20)	94.06	105.955	148.893	117.417	117.869	116.319	131.881
(B, 3)	21	(09:00, 09:20)	103.95	115.639	123.601	109.791	112.418	109.083	125.364
(B, 3)	22	(09:00, 09:20)	92.04	107.542	126.234	127.811	115.891	111.557	126.016
(B, 3)	23	(09:00, 09:20)	64.23	105.892	75.8662	76.3932	104.642	112.538	125.871
(B, 3)	24	(09:00, 09:20)	147.68	107.594	87.6282	78.5874	97.2497	104.342	120.269
(B, 3)	18	(09:20, 09:40)	151.95	132.676	107.7	163.841	109.279	112.342	129.314
(B, 3)	19	(09:20, 09:40)	108.18	115.576	108.145	111.254	110.647	110.802	127.334
(B, 3)	20	(09:20, 09:40)	114.88	109.006	128.414	112.829	119.316	113.718	130.182
(B, 3)	21	(09:20, 09:40)	82.41	135.261	144	152.089	123.75	124.994	131.622
(B, 3)	22	(09:20, 09:40)	116.15	133.582	114.407	141.349	119.175	112.657	131.562
(B, 3)	23	(09:20, 09:40)	120.49	113.624	107.613	81.6542	118.062	116.03	126.483

(B, 3)	24	(09:20, 09:40)	124.16	105.467	116.635	111.51	113.732	112.613	125.281
(B, 3)	18	(09:40, 10:00)	117.69	111.949	119.535	130.034	111.251	107.402	114.505
(B, 3)	19	(09:40, 10:00)	112.14	106.27	120.39	124.116	117.42	105.741	115.074
(B, 3)	20	(09:40, 10:00)	70.85	110.284	122.557	100.08	121.086	113.048	115.349
(B, 3)	21	(09:40, 10:00)	107.51	108.946	108.874	103.523	115.72	109.891	116.019
(B, 3)	22	(09:40, 10:00)	114.06	112.044	119.536	108.739	117.287	108.851	115.687
(B, 3)	23	(09:40, 10:00)	105.35	112.119	106.296	51.6283	105.471	107.253	112.761
(B, 3)	24	(09:40, 10:00)	136.56	104.401	101.616	92.1938	98.5549	106.028	111.215
(B, 3)	18	(17:00, 17:20)	80.49	125.468	103.23	124.486	92.5772	106.702	119.028
(B, 3)	19	(17:00, 17:20)	106.34	111.748	105.585	71.4086	105.58	103.42	117.784
(B, 3)	20	(17:00, 17:20)	116.79	111.598	111.857	94.2966	102.554	104.629	112.904
(B, 3)	21	(17:00, 17:20)	84.76	129.368	115.164	95.1283	104.2	107.098	110.212
(B, 3)	22	(17:00, 17:20)	87.86	97.1639	95.0354	159.836	92.7728	102.43	108.704

(B, 3)	23	(17:00, 17:20)	88.62	86.6962	110.588	260.796	98.9538	113.116	108.205
(B, 3)	24	(17:00, 17:20)	110.46	97.9109	102.48	95.7468	102.278	105.916	115.825
(B, 3)	18	(17:20, 17:40)	97.56	108.997	103.853	74.181	101.361	107.166	111.791
(B, 3)	19	(17:20, 17:40)	105.73	91.9644	115.126	165.18	111.336	109.844	109.458
(B, 3)	20	(17:20, 17:40)	93.09	92.9871	107.987	64.9488	106.636	108.144	108.966
(B, 3)	21	(17:20, 17:40)	108.29	122.073	111.773	103.076	103.889	109.636	112.51
(B, 3)	22	(17:20, 17:40)	103.59	99.6457	105.698	102.162	101.612	107.811	108.274
(B, 3)	23	(17:20, 17:40)	104.55	108.073	120.199	237.384	109.693	115.131	109.489
(B, 3)	24	(17:20, 17:40)	57.83	107.342	121.126	107.949	118.717	118.397	111.059
(B, 3)	18	(17:40, 18:00)	104.47	114.338	98.666	139.767	90.5048	108.416	108.341
(B, 3)	19	(17:40, 18:00)	101.06	98.9788	107.242	141.482	109.023	111.726	116.683
(B, 3)	20	(17:40, 18:00)	87.7	97.267	99.5185	31.5133	98.3132	110.324	109.898
(B, 3)	21	(17:40, 18:00)	130.19	106.898	99.093	67.1113	101.32	111.259	109.771

(B, 3)	22	(17:40, 18:00)	120.99	113.331	106.625	125.722	96.3231	111.573	108.179
(B, 3)	23	(17:40, 18:00)	90.96	103.892	103.666	105.28	98.7509	110.26	107.736
(B, 3)	24	(17:40, 18:00)	118.39	107.469	96.8074	160.702	105.181	111.446	116.845
(B, 3)	18	(18:00, 18:20)	125.47	84.8727	112.082	47.8627	104.969	115.231	106.359
(B, 3)	19	(18:00, 18:20)	87.42	105.082	119.6	129.643	114.416	114.81	110.508
(B, 3)	20	(18:00, 18:20)	99.22	113.958	119.145	126.551	110.27	122.408	112.241
(B, 3)	21	(18:00, 18:20)	137.44	94.6796	97.6465	52.3641	101.269	116.817	107.773
(B, 3)	22	(18:00, 18:20)	95.22	92.7508	108.211	183.439	102.005	112.841	107.589
(B, 3)	23	(18:00, 18:20)	109.01	138.828	115.748	292.549	103.443	110.273	111.79
(B, 3)	24	(18:00, 18:20)	104.99	122.252	115.815	62.8807	117.125	117.69	112.96
(B, 3)	18	(18:20, 18:40)	111.18	81.4264	93.0808	27.23	104.937	144.599	171.056
(B, 3)	19	(18:20, 18:40)	110.38	93.6291	207.143	93.2979	94.8594	145.577	135.655
(B, 3)	20	(18:20, 18:40)	80.24	106.586	170.193	30.864	102.762	148.93	116.292

(B, 3)	21	(18:20, 18:40)	120.71	110.863	255.743	941.032	105.323	146.808	143.376
(B, 3)	22	(18:20, 18:40)	72.77	104.482	171.458	59.6382	102.946	144.52	115.255
(B, 3)	23	(18:20, 18:40)	81.63	117.34	142.709	752.936	117.608	146.889	115.675
(B, 3)	24	(18:20, 18:40)	129.71	81.3249	164.042	62.5484	105.019	146.246	116.33
(B, 3)	18	(18:40, 19:00)	117.39	122.336	106.574	59.9307	113.195	114.214	123.699
(B, 3)	19	(18:40, 19:00)	80.84	138.996	122.923	95.4608	112.921	115.685	120.427
(B, 3)	20	(18:40, 19:00)	106.83	104.846	119.982	34.1957	111.542	112.771	114.33
(B, 3)	21	(18:40, 19:00)	60.02	108.589	134.05	81.6866	113.416	119.915	114.466
(B, 3)	22	(18:40, 19:00)	88.6	109.834	122.953	44.0829	114.198	117.933	115.32
(B, 3)	23	(18:40, 19:00)	104.62	95.8395	117.764	190.852	115.17	119.201	115.065
(B, 3)	24	(18:40, 19:00)	46.32	103.724	105.565	87.8061	111.045	120.966	114.846
(C, 1)	18	(08:00, 08:20)	134.51	158.003	180.136	173.388	180.725	161.314	177.81
(C, 1)	19	(08:00, 08:20)	158.18	144.642	149.433	155.51	168.427	160.136	164.664

(C, 1)	20	(08:00, 08:20)	200.43	180.698	222.051	206.521	174.824	167.613	167.589
(C, 1)	21	(08:00, 08:20)	147.11	174.011	195.107	223.007	172.288	166.101	163.818
(C, 1)	22	(08:00, 08:20)	-	-	-	-	-	-	-
(C, 1)	23	(08:00, 08:20)	159.65	163.286	194.788	48.6186	170.268	161.145	169.166
(C, 1)	24	(08:00, 08:20)	159.83	141.92	139.968	175.324	162.581	160.371	163.381
(C, 1)	18	(08:20, 08:40)	183.06	186.413	150.644	176.816	177.085	162.267	174.119
(C, 1)	19	(08:20, 08:40)	189.34	166.253	138.846	134.354	172.489	162.055	172.429
(C, 1)	20	(08:20, 08:40)	178.94	148.953	163.065	196.806	154.071	159.1	168.366
(C, 1)	21	(08:20, 08:40)	193.38	145.848	173.64	191.84	152.821	163.072	165.11
(C, 1)	22	(08:20, 08:40)	169.57	176.839	156.822	195.509	160.74	159.72	170.08
(C, 1)	23	(08:20, 08:40)	186.79	166.792	198.852	269.718	186.611	166.17	173.601
(C, 1)	24	(08:20, 08:40)	156.72	172.449	173.679	192.871	173.71	171.893	174.513
(C, 1)	18	(08:40, 09:00)	-	-	-	-	-	-	-

(C, 1)	19	(08:40, 09:00)	140.53	168.51	163.033	187.533	164.882	160.746	172.312
(C, 1)	20	(08:40, 09:00)	200.76	161.663	236.155	226.061	175.334	165.295	174.87
(C, 1)	21	(08:40, 09:00)	-	-	-	-	-	-	-
(C, 1)	22	(08:40, 09:00)	173.51	139.915	195.875	153.936	165.362	162.145	164.341
(C, 1)	23	(08:40, 09:00)	153.82	156.078	187.97	117.833	159.987	164.308	177.254
(C, 1)	24	(08:40, 09:00)	157.63	166.173	168.076	221.181	166.489	165.889	177.879
(C, 1)	18	(09:00, 09:20)	158.02	178.21	147.281	218.011	166.845	161.472	175.356
(C, 1)	19	(09:00, 09:20)	158.96	176.584	161.17	152.83	167.753	169.612	176.385
(C, 1)	20	(09:00, 09:20)	138.53	189.094	174.254	122.684	169.287	173.632	181.478
(C, 1)	21	(09:00, 09:20)	149.93	146.066	165.978	200.95	167.559	174.671	172.812
(C, 1)	22	(09:00, 09:20)	187.36	152.338	151.596	266.676	162.195	165.985	172.427
(C, 1)	23	(09:00, 09:20)	211.64	204.411	193.862	253.261	180.386	169.684	186.685
(C, 1)	24	(09:00, 09:20)	-	-	-	-	-	-	-

(C, 1)	18	(09:20, 09:40)	130.53	152.97	173.203	173.319	169.279	183.371	179.254
(C, 1)	19	(09:20, 09:40)	250.19	132.75	166.501	159.856	174.085	173.055	174.605
(C, 1)	20	(09:20, 09:40)	182.56	165.943	167.689	169.017	159.825	171.424	174.845
(C, 1)	21	(09:20, 09:40)	182.18	169.359	180.837	182.338	164.561	174.88	173.461
(C, 1)	22	(09:20, 09:40)	187.97	172.038	182.538	200.826	163.53	176.702	181.565
(C, 1)	23	(09:20, 09:40)	153.05	195.044	203.119	132.083	179.718	185.12	187.478
(C, 1)	24	(09:20, 09:40)	136.74	153.746	205.63	221.331	181.794	185.072	187.329
(C, 1)	18	(09:40, 10:00)	151.99	197.508	205.722	326.666	193.059	185.026	191.584
(C, 1)	19	(09:40, 10:00)	220.91	192.135	192.697	226.885	181.564	183.933	186.668
(C, 1)	20	(09:40, 10:00)	152.07	185.333	205.306	185.281	178.517	179.174	184.72
(C, 1)	21	(09:40, 10:00)	173.31	176.688	174.334	152.138	175.402	181.984	178.403
(C, 1)	22	(09:40, 10:00)	-	-	-	-	-	-	-
(C, 1)	23	(09:40, 10:00)	447.96	180.253	219.32	218.112	182.815	195.2	185.372

(C, 1)	24	(09:40, 10:00)	205.9	179.944	171.481	195.24	175.536	183.204	188.011
(C, 1)	18	(17:00, 17:20)	417.44	212.881	220.998	50.9242	183.137	187.458	196.155
(C, 1)	19	(17:00, 17:20)	233.4	163.859	195.49	371.028	186.706	191.928	197.261
(C, 1)	20	(17:00, 17:20)	226.48	175.555	219.457	153.125	180.065	177.066	179.977
(C, 1)	21	(17:00, 17:20)	265.45	216.862	219.617	185.701	187.418	191.554	199.064
(C, 1)	22	(17:00, 17:20)	331.19	195.575	210.743	255.79	182.29	186.221	188.661
(C, 1)	23	(17:00, 17:20)	227.86	215.593	237.314	201	182.682	186.359	186.885
(C, 1)	24	(17:00, 17:20)	190.15	171.897	238.374	462.43	184.033	191.074	197.124
(C, 1)	18	(17:20, 17:40)	185.55	185.837	225.954	171.636	210.074	190.309	207.408
(C, 1)	19	(17:20, 17:40)	190.28	189.778	246.013	326.916	207.031	198.51	208.961
(C, 1)	20	(17:20, 17:40)	312.25	214.27	183.861	177.136	187.793	186.472	205.732
(C, 1)	21	(17:20, 17:40)	465.17	164.036	218.489	256.964	216.646	195.337	196.542
(C, 1)	22	(17:20, 17:40)	240.21	203.638	225.085	202.873	198.548	198.555	202.895

(C, 1)	23	(17:20, 17:40)	229.92	221.266	219.074	432.774	212.789	204.739	196.542
(C, 1)	24	(17:20, 17:40)	154.65	202.787	175.702	193.714	212.233	206.376	205.976
(C, 1)	18	(17:40, 18:00)	238.49	211.687	235.384	153.523	188.828	203.728	211.718
(C, 1)	19	(17:40, 18:00)	198.43	172.148	181.958	390.595	188.937	200.384	208.299
(C, 1)	20	(17:40, 18:00)	-	-	-	-	-	-	-
(C, 1)	21	(17:40, 18:00)	482.11	323.587	240.995	189.651	195.243	209.499	233.555
(C, 1)	22	(17:40, 18:00)	229.13	164.865	184.394	131.301	181.884	189.17	189.702
(C, 1)	23	(17:40, 18:00)	221.27	215.003	212.815	239.186	193.758	201.282	207.847
(C, 1)	24	(17:40, 18:00)	160.04	243.488	295.962	333.461	210.203	220.584	215.397
(C, 1)	18	(18:00, 18:20)	179.15	238.629	224.633	161.134	204.006	198.174	218.459
(C, 1)	19	(18:00, 18:20)	283.97	234.029	175.463	260.481	188.287	187.612	205.466
(C, 1)	20	(18:00, 18:20)	174.53	171.485	185.405	159.22	189.688	185.076	193.694
(C, 1)	21	(18:00, 18:20)	487.41	230.789	190.554	64.334	204.608	196.937	201.168

(C, 1)	22	(18:00, 18:20)	271.24	189.148	222.856	303.73	191.034	190.935	195.183
(C, 1)	23	(18:00, 18:20)	200.4	211.677	300.712	656.626	205.393	191.571	212.085
(C, 1)	24	(18:00, 18:20)	271.2	213.99	172.504	359.41	200.239	196.915	213.457
(C, 1)	18	(18:20, 18:40)	263.94	310.08	265.772	139.488	229.966	240.698	241.809
(C, 1)	19	(18:20, 18:40)	314.9	246.97	319.726	393.127	230.735	230.526	248.838
(C, 1)	20	(18:20, 18:40)	149.01	223.689	246.001	138.862	216.321	207.161	219.105
(C, 1)	21	(18:20, 18:40)	490.26	180.849	283.865	286.16	224.308	232.83	218.883
(C, 1)	22	(18:20, 18:40)	245.61	234.969	188.094	143.976	217.188	217.608	239.244
(C, 1)	23	(18:20, 18:40)	224.13	217.851	97.0562	309.831	198.63	223.995	243.201
(C, 1)	24	(18:20, 18:40)	260.35	288.776	297.399	402.59	215.545	228.84	241.611
(C, 1)	18	(18:40, 19:00)	390.85	321.876	259.39	172.565	221.766	220.513	244.575
(C, 1)	19	(18:40, 19:00)	-	-	-	-	-	-	-
(C, 1)	20	(18:40, 19:00)	165.61	240.417	217.348	91.1571	205.073	213.169	218.702

(C, 1)	21	(18:40, 19:00)	317.67	259.436	246.779	160.327	214.323	216.323	225.826
(C, 1)	22	(18:40, 19:00)	233.02	200.804	195.819	206.83	198.691	209.501	206.864
(C, 1)	23	(18:40, 19:00)	122.38	223.651	186.291	96.8539	190.651	205.421	212.946
(C, 1)	24	(18:40, 19:00)	-	-	-	-	-	-	-
(C, 3)	18	(08:00, 08:20)	-	-	-	-	-	-	-
(C, 3)	19	(08:00, 08:20)	188.63	101.185	148.846	139.762	151.162	160.54	157.78
(C, 3)	20	(08:00, 08:20)	-	-	-	-	-	-	-
(C, 3)	21	(08:00, 08:20)	-	-	-	-	-	-	-
(C, 3)	22	(08:00, 08:20)	196.43	128.157	119.869	160.108	145.73	160.117	159.014
(C, 3)	23	(08:00, 08:20)	-	-	-	-	-	-	-
(C, 3)	24	(08:00, 08:20)	-	-	-	-	-	-	-
(C, 3)	18	(08:20, 08:40)	233.55	153.581	188.405	128.286	149.067	182.809	168.34
(C, 3)	19	(08:20, 08:40)	175.91	130.648	78.3342	146.392	135.539	177.555	168.622

(C, 3)	20	(08:20, 08:40)	141.54	144.878	181.252	650.54	167.069	190.226	171.978
(C, 3)	21	(08:20, 08:40)	124.56	123.446	81.2824	190.708	145.375	178.623	168.45
(C, 3)	22	(08:20, 08:40)	-	-	-	-	-	-	-
(C, 3)	23	(08:20, 08:40)	-	-	-	-	-	-	-
(C, 3)	24	(08:20, 08:40)	-	-	-	-	-	-	-
(C, 3)	18	(08:40, 09:00)	155.31	181.271	176.797	304.782	156.481	179.403	196.909
(C, 3)	19	(08:40, 09:00)	202.04	185.255	154.536	110.894	158.409	190.976	196.956
(C, 3)	20	(08:40, 09:00)	194.8	177.542	150.719	722.155	160.816	197.289	212.439
(C, 3)	21	(08:40, 09:00)	214.37	135.377	162.712	117.487	162.996	196.349	184.594
(C, 3)	22	(08:40, 09:00)	-	-	-	-	-	-	-
(C, 3)	23	(08:40, 09:00)	-	-	-	-	-	-	-
(C, 3)	24	(08:40, 09:00)	141.63	136.879	43.1853	102.387	155.071	167.909	189.387
(C, 3)	18	(09:00, 09:20)	216.72	115.925	110.664	191.779	154.939	182.773	185.346

(C, 3)	19	(09:00, 09:20)	209.15	244.197	230.803	271.439	169.645	180.781	225.665
(C, 3)	20	(09:00, 09:20)	148.35	122.87	-234.855	453.861	120.458	181.131	182.967
(C, 3)	21	(09:00, 09:20)	106.63	155.332	156.608	140.72	174.707	185.381	179.889
(C, 3)	22	(09:00, 09:20)	134.16	118.401	94.5804	172.037	164.17	178.918	179.502
(C, 3)	23	(09:00, 09:20)	93.53	170.152	155.972	224.731	186.383	190.44	189.396
(C, 3)	24	(09:00, 09:20)	-	-	-	-	-	-	-
(C, 3)	18	(09:20, 09:40)	178.4	190.511	171.445	304.459	168.747	186.557	185.442
(C, 3)	19	(09:20, 09:40)	143.38	241.009	196.105	348.195	195.011	190.028	203.808
(C, 3)	20	(09:20, 09:40)	209.36	251.023	230.06	1322.53	156.422	185.045	202.99
(C, 3)	21	(09:20, 09:40)	155.75	215.933	164.941	189.799	166.339	187.49	196.898
(C, 3)	22	(09:20, 09:40)	174.1	225.008	191.41	338.02	177.031	184.68	197.438
(C, 3)	23	(09:20, 09:40)	211.69	177.948	151.691	305.117	171.795	184.213	190.775
(C, 3)	24	(09:20, 09:40)	142.25	159.82	141.766	187.475	163.907	185.28	176.074

(C, 3)	18	(09:40, 10:00)	144.3	177.996	149.766	214.769	151.312	169.293	174.692
(C, 3)	19	(09:40, 10:00)	168.16	190.439	136.541	189.511	154.964	170.763	184.023
(C, 3)	20	(09:40, 10:00)	-	-	-	-	-	-	-
(C, 3)	21	(09:40, 10:00)	131.44	168.273	144.833	106.403	153.872	171.824	169.728
(C, 3)	22	(09:40, 10:00)	135.19	180.798	156.533	224.79	161.769	169.632	181.88
(C, 3)	23	(09:40, 10:00)	-	-	-	-	-	-	-
(C, 3)	24	(09:40, 10:00)	169.27	203.72	178.94	149.755	160.755	169.947	186.115
(C, 3)	18	(17:00, 17:20)	189.83	151.051	148.174	151.919	179.979	166.669	174.867
(C, 3)	19	(17:00, 17:20)	108.94	173.773	219.229	350.31	198.286	182.113	189.726
(C, 3)	20	(17:00, 17:20)	143.44	181.164	168.261	206.136	186.642	176.177	182.414
(C, 3)	21	(17:00, 17:20)	-	-	-	-	-	-	-
(C, 3)	22	(17:00, 17:20)	199.61	143.109	162.121	214.37	182.943	176.747	178.729
(C, 3)	23	(17:00, 17:20)	-	-	-	-	-	-	-

(C, 3)	24	(17:00, 17:20)	161.79	192.209	146.381	223.048	180.605	181.45	184.518
(C, 3)	18	(17:20, 17:40)	-	-	-	-	-	-	-
(C, 3)	19	(17:20, 17:40)	185.13	137.375	135.641	176.947	165.13	173.185	180.236
(C, 3)	20	(17:20, 17:40)	138.56	189.237	172.634	157.45	171.531	175.565	189.222
(C, 3)	21	(17:20, 17:40)	193.78	179.298	166.214	135.121	180.395	174.73	186.259
(C, 3)	22	(17:20, 17:40)	167.41	173.863	167.6	199.415	174.456	175.684	177.882
(C, 3)	23	(17:20, 17:40)	198.63	234.926	270.599	461.862	210.043	186.95	194.816
(C, 3)	24	(17:20, 17:40)	165.87	206.318	184.247	156.275	177.434	176.951	184.291
(C, 3)	18	(17:40, 18:00)	-	-	-	-	-	-	-
(C, 3)	19	(17:40, 18:00)	-	-	-	-	-	-	-
(C, 3)	20	(17:40, 18:00)	173.81	207.874	219.25	168.091	197.559	208.994	198.963
(C, 3)	21	(17:40, 18:00)	254.98	186.718	204.159	323.949	201.125	201.587	202.342
(C, 3)	22	(17:40, 18:00)	245.12	204.996	188.914	134.412	204.211	199.374	200.168

(C, 3)	23	(17:40, 18:00)	200.92	214.142	181.263	309.773	197.71	202.021	203.186
(C, 3)	24	(17:40, 18:00)	111.53	217.03	200.787	244.766	191.452	205.621	200.796
(C, 3)	18	(18:00, 18:20)	84.47	260.902	222.549	256.665	182.309	185.515	225.729
(C, 3)	19	(18:00, 18:20)	273.95	243.242	143.446	139.484	180.085	180.187	185.892
(C, 3)	20	(18:00, 18:20)	-	-	-	-	-	-	-
(C, 3)	21	(18:00, 18:20)	273.59	248.507	226.854	271.689	191.833	180.803	210.698
(C, 3)	22	(18:00, 18:20)	167.08	189.509	194.596	261.944	179.234	178.802	190.151
(C, 3)	23	(18:00, 18:20)	218.88	232.536	283.523	334.382	174.178	178.653	185.26
(C, 3)	24	(18:00, 18:20)	-	-	-	-	-	-	-
(C, 3)	18	(18:20, 18:40)	-	-	-	-	-	-	-
(C, 3)	19	(18:20, 18:40)	123.48	244.668	174.37	243.746	185.073	210.349	214.787
(C, 3)	20	(18:20, 18:40)	211.71	229.954	201.95	217.71	194.144	210.289	209.836
(C, 3)	21	(18:20, 18:40)	216.57	180.554	186.814	228.128	193.829	212.242	205.99

(C, 3)	22	(18:20, 18:40)	224.5	195.294	239.2	365.226	184.377	197.881	201.71
(C, 3)	23	(18:20, 18:40)	-	-	-	-	-	-	-
(C, 3)	24	(18:20, 18:40)	109.12	239.503	231.408	301.803	191.589	214.26	227.289
(C, 3)	18	(18:40, 19:00)	204.89	202.444	235.471	365.508	178.545	194.4	189.526
(C, 3)	19	(18:40, 19:00)	170.41	205	213.986	386.545	178.887	191.357	184.928
(C, 3)	20	(18:40, 19:00)	-	-	-	-	-	-	-
(C, 3)	21	(18:40, 19:00)	242.64	145.93	217.435	228.645	184.548	194.997	183.845
(C, 3)	22	(18:40, 19:00)	241.57	182.631	147.511	280.754	177.936	185.531	179.548
(C, 3)	23	(18:40, 19:00)	138.99	189.147	78.5217	412.753	177.786	173.133	181.718
(C, 3)	24	(18:40, 19:00)	-	-	-	-	-	-	-

Tabla 8.20: Tabla de predicciones de la primera aproximación de predicciones del tiempo promedio de viaje (La fila con un '-' corresponde a una ruta, día e intervalo de tiempo de la que no se dispone dato real para comparar)

8.1.2.2 Segunda aproximación de predicciones del tiempo promedio de viaje

Ruta	Día	Intervalo de tiempo	Valor Real	Predicho	Modelo ARIMA
(A, 2)	2016-10-18	(08:00, 08:20)	77.77	67.3234	(4, 2, 0)
(A, 2)	2016-10-19	(08:00, 08:20)	91.72	73.5959	(6, 2, 0)
(A, 2)	2016-10-20	(08:00, 08:20)	87.24	71.6124	(6, 2, 0)
(A, 2)	2016-10-21	(08:00, 08:20)	58.58	60.593	(6, 2, 0)
(A, 2)	2016-10-22	(08:00, 08:20)	71.43	69.1888	(4, 1, 0)
(A, 2)	2016-10-23	(08:00, 08:20)	56.62	64.2131	(3, 0, 0)
(A, 2)	2016-10-24	(08:00, 08:20)	70.59	67.9566	(4, 0, 0)
(A, 2)	2016-10-18	(08:20, 08:40)	70.3	69.7792	(4, 2, 0)
(A, 2)	2016-10-19	(08:20, 08:40)	92.28	78.0572	(6, 2, 0)
(A, 2)	2016-10-20	(08:20, 08:40)	75.48	74.0897	(6, 2, 0)
(A, 2)	2016-10-21	(08:20, 08:40)	71.5	62.7237	(6, 2, 0)
(A, 2)	2016-10-22	(08:20, 08:40)	56.06	67.02	(4, 1, 0)

(A, 2)	2016-10-23	(08:20, 08:40)	65.06	66.5419	(3, 0, 0)
(A, 2)	2016-10-24	(08:20, 08:40)	85.32	69.438	(4, 0, 0)
(A, 2)	2016-10-18	(08:40, 09:00)	77.68	69.2461	(4, 2, 0)
(A, 2)	2016-10-19	(08:40, 09:00)	103.94	86.3266	(6, 2, 0)
(A, 2)	2016-10-20	(08:40, 09:00)	79.6	78.0069	(6, 2, 0)
(A, 2)	2016-10-21	(08:40, 09:00)	55.92	66.8687	(6, 2, 0)
(A, 2)	2016-10-22	(08:40, 09:00)	62.2	67.8311	(4, 1, 0)
(A, 2)	2016-10-23	(08:40, 09:00)	65.96	67.9854	(3, 0, 0)
(A, 2)	2016-10-24	(08:40, 09:00)	63.33	69.1887	(4, 0, 0)
(A, 2)	2016-10-18	(09:00, 09:20)	79.28	75.5784	(4, 2, 0)
(A, 2)	2016-10-19	(09:00, 09:20)	79.27	90.7341	(6, 2, 0)
(A, 2)	2016-10-20	(09:00, 09:20)	103.35	82.4682	(6, 2, 0)
(A, 2)	2016-10-21	(09:00, 09:20)	99.53	72.8415	(6, 2, 0)
(A, 2)	2016-10-22	(09:00, 09:20)	79.54	69.8257	(4, 1, 0)

(A, 2)	2016-10-23	(09:00, 09:20)	83.79	68.3739	(3, 0, 0)
(A, 2)	2016-10-24	(09:00, 09:20)	79.18	69.6321	(4, 0, 0)
(A, 2)	2016-10-18	(09:20, 09:40)	83.27	78.5127	(4, 2, 0)
(A, 2)	2016-10-19	(09:20, 09:40)	81.22	96.3532	(6, 2, 0)
(A, 2)	2016-10-20	(09:20, 09:40)	97.05	83.3785	(6, 2, 0)
(A, 2)	2016-10-21	(09:20, 09:40)	83.67	75.472	(6, 2, 0)
(A, 2)	2016-10-22	(09:20, 09:40)	64.81	72.1449	(4, 1, 0)
(A, 2)	2016-10-23	(09:20, 09:40)	64.03	68.8093	(3, 0, 0)
(A, 2)	2016-10-24	(09:20, 09:40)	76.7	69.3373	(4, 0, 0)
(A, 2)	2016-10-18	(09:40, 10:00)	77.67	80.6916	(4, 2, 0)
(A, 2)	2016-10-19	(09:40, 10:00)	79.82	94.5415	(6, 2, 0)
(A, 2)	2016-10-20	(09:40, 10:00)	67.95	84.8199	(6, 2, 0)
(A, 2)	2016-10-21	(09:40, 10:00)	73.3	78.2074	(6, 2, 0)

(A, 2)	2016-10-22	(09:40, 10:00)	70.94	69.6257	(4, 1, 0)
(A, 2)	2016-10-23	(09:40, 10:00)	66.64	69.0453	(3, 0, 0)
(A, 2)	2016-10-24	(09:40, 10:00)	51.01	69.391	(4, 0, 0)
(A, 2)	2016-10-18	(17:00, 17:20)	75	73.1611	(4, 1, 1)
(A, 2)	2016-10-19	(17:00, 17:20)	60	63.3334	(8, 2, 0)
(A, 2)	2016-10-20	(17:00, 17:20)	96.52	66.6135	(9, 1, 4)
(A, 2)	2016-10-21	(17:00, 17:20)	101.05	80.2676	(9, 1, 0)
(A, 2)	2016-10-22	(17:00, 17:20)	71.32	69.2547	(9, 1, 4)
(A, 2)	2016-10-23	(17:00, 17:20)	94.08	78.5015	(6, 2, 0)
(A, 2)	2016-10-24	(17:00, 17:20)	85.03	66.5701	(6, 1, 0)
(A, 2)	2016-10-18	(17:20, 17:40)	63.3	72.5596	(4, 1, 1)
(A, 2)	2016-10-19	(17:20, 17:40)	56.98	64.4938	(8, 2, 0)
(A, 2)	2016-10-20	(17:20, 17:40)	54.93	66.5082	(9, 1, 4)

(A, 2)	2016-10-21	(17:20, 17:40)	82.62	79.0325	(9, 1, 0)
(A, 2)	2016-10-22	(17:20, 17:40)	63.61	68.5209	(9, 1, 4)
(A, 2)	2016-10-23	(17:20, 17:40)	67.87	75.8005	(6, 2, 0)
(A, 2)	2016-10-24	(17:20, 17:40)	68.98	66.2613	(6, 1, 0)
(A, 2)	2016-10-18	(17:40, 18:00)	68.39	71.4523	(4, 1, 1)
(A, 2)	2016-10-19	(17:40, 18:00)	66.91	64.9213	(8, 2, 0)
(A, 2)	2016-10-20	(17:40, 18:00)	82.28	67.3303	(9, 1, 4)
(A, 2)	2016-10-21	(17:40, 18:00)	62	78.1616	(9, 1, 0)
(A, 2)	2016-10-22	(17:40, 18:00)	77.17	68.731	(9, 1, 4)
(A, 2)	2016-10-23	(17:40, 18:00)	63.09	70.3188	(6, 2, 0)
(A, 2)	2016-10-24	(17:40, 18:00)	53.84	64.1436	(6, 1, 0)
(A, 2)	2016-10-18	(18:00, 18:20)	88.83	70.5129	(4, 1, 1)
(A, 2)	2016-10-19	(18:00, 18:20)	63.8	65.0467	(8, 2, 0)

(A, 2)	2016-10-20	(18:00, 18:20)	75.25	67.1398	(9, 1, 4)
(A, 2)	2016-10-21	(18:00, 18:20)	76.75	78.4769	(9, 1, 0)
(A, 2)	2016-10-22	(18:00, 18:20)	69.61	68.4482	(9, 1, 4)
(A, 2)	2016-10-23	(18:00, 18:20)	63.97	69.906	(6, 2, 0)
(A, 2)	2016-10-24	(18:00, 18:20)	65.13	65.2262	(6, 1, 0)
(A, 2)	2016-10-18	(18:20, 18:40)	59.33	69.4325	(4, 1, 1)
(A, 2)	2016-10-19	(18:20, 18:40)	55.74	65.7423	(8, 2, 0)
(A, 2)	2016-10-20	(18:20, 18:40)	50.34	67.697	(9, 1, 4)
(A, 2)	2016-10-21	(18:20, 18:40)	81.04	79.0713	(9, 1, 0)
(A, 2)	2016-10-22	(18:20, 18:40)	66.35	68.4443	(9, 1, 4)
(A, 2)	2016-10-23	(18:20, 18:40)	60.32	67.2641	(6, 2, 0)
(A, 2)	2016-10-24	(18:20, 18:40)	61.35	66.9148	(6, 1, 0)
(A, 2)	2016-10-18	(18:40, 19:00)	52.23	69.0847	(4, 1, 1)

(A, 2)	2016-10-19	(18:40, 19:00)	84.02	66.6811	(8, 2, 0)
(A, 2)	2016-10-20	(18:40, 19:00)	57.25	67.5196	(9, 1, 4)
(A, 2)	2016-10-21	(18:40, 19:00)	70.41	80.9093	(9, 1, 0)
(A, 2)	2016-10-22	(18:40, 19:00)	58.64	68.0604	(9, 1, 4)
(A, 2)	2016-10-23	(18:40, 19:00)	69.44	68.7497	(6, 2, 0)
(A, 2)	2016-10-24	(18:40, 19:00)	62.85	65.3445	(6, 1, 0)
(A, 3)	2016-10-18	(08:00, 08:20)	197.5	183.151	(3, 1, 0)
(A, 3)	2016-10-19	(08:00, 08:20)	319.67	220.148	(4, 2, 0)
(A, 3)	2016-10-20	(08:00, 08:20)	183.76	192.855	(9, 2, 0)
(A, 3)	2016-10-21	(08:00, 08:20)	258.88	188.499	(3, 2, 0)
(A, 3)	2016-10-22	(08:00, 08:20)	120.94	133.25	(3, 1, 0)
(A, 3)	2016-10-23	(08:00, 08:20)	91.06	102.831	(6, 1, 0)
(A, 3)	2016-10-24	(08:00, 08:20)	265.38	188.387	(4, 1, 0)

(A, 3)	2016-10-18	(08:20, 08:40)	166.9	175.312	(3, 1, 0)
(A, 3)	2016-10-19	(08:20, 08:40)	328.02	232.855	(4, 2, 0)
(A, 3)	2016-10-20	(08:20, 08:40)	242.81	205.03	(9, 2, 0)
(A, 3)	2016-10-21	(08:20, 08:40)	270.64	201.655	(3, 2, 0)
(A, 3)	2016-10-22	(08:20, 08:40)	113.81	135.569	(3, 1, 0)
(A, 3)	2016-10-23	(08:20, 08:40)	90.74	102.943	(6, 1, 0)
(A, 3)	2016-10-24	(08:20, 08:40)	234.99	185.162	(4, 1, 0)
(A, 3)	2016-10-18	(08:40, 09:00)	253.82	180.548	(3, 1, 0)
(A, 3)	2016-10-19	(08:40, 09:00)	326.7	263.473	(4, 2, 0)
(A, 3)	2016-10-20	(08:40, 09:00)	224.49	221.843	(9, 2, 0)
(A, 3)	2016-10-21	(08:40, 09:00)	240.81	220.258	(3, 2, 0)
(A, 3)	2016-10-22	(08:40, 09:00)	195.13	139.016	(3, 1, 0)
(A, 3)	2016-10-23	(08:40, 09:00)	111.3	104	(6, 1, 0)

(A, 3)	2016-10-24	(08:40, 09:00)	181.75	189.545	(4, 1, 0)
(A, 3)	2016-10-18	(09:00, 09:20)	240.16	184.639	(3, 1, 0)
(A, 3)	2016-10-19	(09:00, 09:20)	339.07	287.016	(4, 2, 0)
(A, 3)	2016-10-20	(09:00, 09:20)	201.63	236.23	(9, 2, 0)
(A, 3)	2016-10-21	(09:00, 09:20)	237.97	222.991	(3, 2, 0)
(A, 3)	2016-10-22	(09:00, 09:20)	116.25	138.338	(3, 1, 0)
(A, 3)	2016-10-23	(09:00, 09:20)	116.89	104.276	(6, 1, 0)
(A, 3)	2016-10-24	(09:00, 09:20)	262.43	190.75	(4, 1, 0)
(A, 3)	2016-10-18	(09:20, 09:40)	179.19	181.216	(3, 1, 0)
(A, 3)	2016-10-19	(09:20, 09:40)	316.84	305.097	(4, 2, 0)
(A, 3)	2016-10-20	(09:20, 09:40)	263.23	250.308	(9, 2, 0)
(A, 3)	2016-10-21	(09:20, 09:40)	250.89	245.057	(3, 2, 0)
(A, 3)	2016-10-22	(09:20, 09:40)	124.06	137.002	(3, 1, 0)

(A, 3)	2016-10-23	(09:20, 09:40)	103.09	102.857	(6, 1, 0)
(A, 3)	2016-10-24	(09:20, 09:40)	146.56	192.018	(4, 1, 0)
(A, 3)	2016-10-18	(09:40, 10:00)	136.98	180.934	(3, 1, 0)
(A, 3)	2016-10-19	(09:40, 10:00)	199.89	322.459	(4, 2, 0)
(A, 3)	2016-10-20	(09:40, 10:00)	182.37	261.332	(9, 2, 0)
(A, 3)	2016-10-21	(09:40, 10:00)	194.2	256.136	(3, 2, 0)
(A, 3)	2016-10-22	(09:40, 10:00)	147.83	137.57	(3, 1, 0)
(A, 3)	2016-10-23	(09:40, 10:00)	95.62	104.184	(6, 1, 0)
(A, 3)	2016-10-24	(09:40, 10:00)	92.95	189.898	(4, 1, 0)
(A, 3)	2016-10-18	(17:00, 17:20)	136.34	121.671	(8, 0, 1)
(A, 3)	2016-10-19	(17:00, 17:20)	152.27	118.809	(6, 2, 0)
(A, 3)	2016-10-20	(17:00, 17:20)	139.66	123.094	(9, 1, 0)
(A, 3)	2016-10-21	(17:00, 17:20)	135.93	121.77	(5, 2, 0)

(A, 3)	2016-10-22	(17:00, 17:20)	110.48	114.927	(8, 2, 0)
(A, 3)	2016-10-23	(17:00, 17:20)	112.65	112.236	(8, 1, 1)
(A, 3)	2016-10-24	(17:00, 17:20)	121.97	117.735	(5, 1, 0)
(A, 3)	2016-10-18	(17:20, 17:40)	143.7	120.263	(8, 0, 1)
(A, 3)	2016-10-19	(17:20, 17:40)	123.18	120.585	(6, 2, 0)
(A, 3)	2016-10-20	(17:20, 17:40)	126.32	123.332	(9, 1, 0)
(A, 3)	2016-10-21	(17:20, 17:40)	121.93	114.974	(5, 2, 0)
(A, 3)	2016-10-22	(17:20, 17:40)	113.84	117.028	(8, 2, 0)
(A, 3)	2016-10-23	(17:20, 17:40)	107.82	109.441	(8, 1, 1)
(A, 3)	2016-10-24	(17:20, 17:40)	124.4	118.538	(5, 1, 0)
(A, 3)	2016-10-18	(17:40, 18:00)	100.13	117.671	(8, 0, 1)
(A, 3)	2016-10-19	(17:40, 18:00)	112.64	119.155	(6, 2, 0)
(A, 3)	2016-10-20	(17:40, 18:00)	124.92	123.506	(9, 1, 0)

(A, 3)	2016-10-21	(17:40, 18:00)	107.05	112.938	(5, 2, 0)
(A, 3)	2016-10-22	(17:40, 18:00)	124.66	116.428	(8, 2, 0)
(A, 3)	2016-10-23	(17:40, 18:00)	112.35	107.068	(8, 1, 1)
(A, 3)	2016-10-24	(17:40, 18:00)	95.07	118.959	(5, 1, 0)
(A, 3)	2016-10-18	(18:00, 18:20)	122.91	115.192	(8, 0, 1)
(A, 3)	2016-10-19	(18:00, 18:20)	110.19	118.912	(6, 2, 0)
(A, 3)	2016-10-20	(18:00, 18:20)	139.16	122.518	(9, 1, 0)
(A, 3)	2016-10-21	(18:00, 18:20)	113.41	107.631	(5, 2, 0)
(A, 3)	2016-10-22	(18:00, 18:20)	88.7	115.395	(8, 2, 0)
(A, 3)	2016-10-23	(18:00, 18:20)	98.51	104.074	(8, 1, 1)
(A, 3)	2016-10-24	(18:00, 18:20)	109.61	120.014	(5, 1, 0)
(A, 3)	2016-10-18	(18:20, 18:40)	112.75	112.745	(8, 0, 1)
(A, 3)	2016-10-19	(18:20, 18:40)	100.43	118.814	(6, 2, 0)

(A, 3)	2016-10-20	(18:20, 18:40)	128.43	124.381	(9, 1, 0)
(A, 3)	2016-10-21	(18:20, 18:40)	123.13	110.701	(5, 2, 0)
(A, 3)	2016-10-22	(18:20, 18:40)	103.83	112.823	(8, 2, 0)
(A, 3)	2016-10-23	(18:20, 18:40)	103.75	102.224	(8, 1, 1)
(A, 3)	2016-10-24	(18:20, 18:40)	109.73	120.731	(5, 1, 0)
(A, 3)	2016-10-18	(18:40, 19:00)	120.58	111.473	(8, 0, 1)
(A, 3)	2016-10-19	(18:40, 19:00)	150.64	124.661	(6, 2, 0)
(A, 3)	2016-10-20	(18:40, 19:00)	95.71	123.054	(9, 1, 0)
(A, 3)	2016-10-21	(18:40, 19:00)	107.53	106.079	(5, 2, 0)
(A, 3)	2016-10-22	(18:40, 19:00)	170.18	115.786	(8, 2, 0)
(A, 3)	2016-10-23	(18:40, 19:00)	102.92	103.437	(8, 1, 1)
(A, 3)	2016-10-24	(18:40, 19:00)	165.64	122.606	(5, 1, 0)
(B, 1)	2016-10-18	(08:00, 08:20)	88.71	100.833	(3, 1, 0)

(B, 1)	2016-10-19	(08:00, 08:20)	92.31	130.749	(7, 2, 0)
(B, 1)	2016-10-20	(08:00, 08:20)	108.92	108.326	(4, 1, 0)
(B, 1)	2016-10-21	(08:00, 08:20)	126.64	110.668	(3, 1, 3)
(B, 1)	2016-10-22	(08:00, 08:20)	105.17	121.164	(8, 1, 0)
(B, 1)	2016-10-23	(08:00, 08:20)	139.86	134.432	(7, 2, 0)
(B, 1)	2016-10-24	(08:00, 08:20)	-	-	-
(B, 1)	2016-10-18	(08:20, 08:40)	136.07	110.007	(3, 1, 0)
(B, 1)	2016-10-19	(08:20, 08:40)	135.66	135.152	(7, 2, 0)
(B, 1)	2016-10-20	(08:20, 08:40)	138.17	113.95	(4, 1, 0)
(B, 1)	2016-10-21	(08:20, 08:40)	163	111.177	(3, 1, 3)
(B, 1)	2016-10-22	(08:20, 08:40)	116.39	122.149	(8, 1, 0)
(B, 1)	2016-10-23	(08:20, 08:40)	111.58	142.785	(7, 2, 0)
(B, 1)	2016-10-24	(08:20, 08:40)	-	-	-

(B, 1)	2016-10-18	(08:40, 09:00)	108.32	109.567	(3, 1, 0)
(B, 1)	2016-10-19	(08:40, 09:00)	153.98	140.904	(7, 2, 0)
(B, 1)	2016-10-20	(08:40, 09:00)	83.41	112.637	(4, 1, 0)
(B, 1)	2016-10-21	(08:40, 09:00)	118.02	111.479	(3, 1, 3)
(B, 1)	2016-10-22	(08:40, 09:00)	107.39	122.428	(8, 1, 0)
(B, 1)	2016-10-23	(08:40, 09:00)	-	-	-
(B, 1)	2016-10-24	(08:40, 09:00)	112.63	108.966	(6, 1, 0)
(B, 1)	2016-10-18	(09:00, 09:20)	109.72	111.6	(3, 1, 0)
(B, 1)	2016-10-19	(09:00, 09:20)	117.8	139.721	(7, 2, 0)
(B, 1)	2016-10-20	(09:00, 09:20)	119.25	113.407	(4, 1, 0)
(B, 1)	2016-10-21	(09:00, 09:20)	123.06	111.687	(3, 1, 3)
(B, 1)	2016-10-22	(09:00, 09:20)	130.69	124.11	(8, 1, 0)
(B, 1)	2016-10-23	(09:00, 09:20)	189.01	149.463	(7, 2, 0)

(B, 1)	2016-10-24	(09:00, 09:20)	95.66	107.736	(6, 1, 0)
(B, 1)	2016-10-18	(09:20, 09:40)	121.11	108.547	(3, 1, 0)
(B, 1)	2016-10-19	(09:20, 09:40)	137.92	142.621	(7, 2, 0)
(B, 1)	2016-10-20	(09:20, 09:40)	105.32	113.97	(4, 1, 0)
(B, 1)	2016-10-21	(09:20, 09:40)	115.15	111.807	(3, 1, 3)
(B, 1)	2016-10-22	(09:20, 09:40)	144.58	124.486	(8, 1, 0)
(B, 1)	2016-10-23	(09:20, 09:40)	-	-	-
(B, 1)	2016-10-24	(09:20, 09:40)	124.09	111.267	(6, 1, 0)
(B, 1)	2016-10-18	(09:40, 10:00)	103.39	109.897	(3, 1, 0)
(B, 1)	2016-10-19	(09:40, 10:00)	142.04	145.443	(7, 2, 0)
(B, 1)	2016-10-20	(09:40, 10:00)	132.5	112.683	(4, 1, 0)
(B, 1)	2016-10-21	(09:40, 10:00)	97.77	111.843	(3, 1, 3)
(B, 1)	2016-10-22	(09:40, 10:00)	126.19	125.997	(8, 1, 0)

(B, 1)	2016-10-23	(09:40, 10:00)	107.28	148.762	(7, 2, 0)
(B, 1)	2016-10-24	(09:40, 10:00)	111.86	108.067	(6, 1, 0)
(B, 1)	2016-10-18	(17:00, 17:20)	120	127.89	(7, 2, 0)
(B, 1)	2016-10-19	(17:00, 17:20)	120.2	125.471	(9, 1, 0)
(B, 1)	2016-10-20	(17:00, 17:20)	87.61	119.061	(3, 0, 0)
(B, 1)	2016-10-21	(17:00, 17:20)	174.46	158.508	(9, 2, 0)
(B, 1)	2016-10-22	(17:00, 17:20)	160.13	142.181	(5, 2, 0)
(B, 1)	2016-10-23	(17:00, 17:20)	84.84	130.556	(9, 1, 4)
(B, 1)	2016-10-24	(17:00, 17:20)	114.14	127.387	(6, 2, 0)
(B, 1)	2016-10-18	(17:20, 17:40)	150.77	134.449	(7, 2, 0)
(B, 1)	2016-10-19	(17:20, 17:40)	-	-	-
(B, 1)	2016-10-20	(17:20, 17:40)	173.74	118.887	(3, 0, 0)
(B, 1)	2016-10-21	(17:20, 17:40)	221.76	165.457	(9, 2, 0)

(B, 1)	2016-10-22	(17:20, 17:40)	155.09	137.397	(5, 2, 0)
(B, 1)	2016-10-23	(17:20, 17:40)	94.28	130.652	(9, 1, 4)
(B, 1)	2016-10-24	(17:20, 17:40)	128.66	127.941	(6, 2, 0)
(B, 1)	2016-10-18	(17:40, 18:00)	154.23	139.35	(7, 2, 0)
(B, 1)	2016-10-19	(17:40, 18:00)	115.66	128.985	(9, 1, 0)
(B, 1)	2016-10-20	(17:40, 18:00)	140.4	116.093	(3, 0, 0)
(B, 1)	2016-10-21	(17:40, 18:00)	-	-	-
(B, 1)	2016-10-22	(17:40, 18:00)	111.85	135.798	(5, 2, 0)
(B, 1)	2016-10-23	(17:40, 18:00)	136.49	127.559	(9, 1, 4)
(B, 1)	2016-10-24	(17:40, 18:00)	123.62	129.946	(6, 2, 0)
(B, 1)	2016-10-18	(18:00, 18:20)	131.32	141.77	(7, 2, 0)
(B, 1)	2016-10-19	(18:00, 18:20)	141.59	129.95	(9, 1, 0)
(B, 1)	2016-10-20	(18:00, 18:20)	21.7	112.375	(3, 0, 0)

(B, 1)	2016-10-21	(18:00, 18:20)	392.65	182.254	(9, 2, 0)
(B, 1)	2016-10-22	(18:00, 18:20)	115.5	136.333	(5, 2, 0)
(B, 1)	2016-10-23	(18:00, 18:20)	125.39	122.762	(9, 1, 4)
(B, 1)	2016-10-24	(18:00, 18:20)	146.24	134.323	(6, 2, 0)
(B, 1)	2016-10-18	(18:20, 18:40)	131.49	143.447	(7, 2, 0)
(B, 1)	2016-10-19	(18:20, 18:40)	-	-	-
(B, 1)	2016-10-20	(18:20, 18:40)	143.75	111.224	(3, 0, 0)
(B, 1)	2016-10-21	(18:20, 18:40)	-	-	-
(B, 1)	2016-10-22	(18:20, 18:40)	175.15	143.408	(5, 2, 0)
(B, 1)	2016-10-23	(18:20, 18:40)	104.97	120.546	(9, 1, 4)
(B, 1)	2016-10-24	(18:20, 18:40)	-	-	-
(B, 1)	2016-10-18	(18:40, 19:00)	175.32	153.434	(7, 2, 0)
(B, 1)	2016-10-19	(18:40, 19:00)	-	-	-

(B, 1)	2016-10-20	(18:40, 19:00)	137.51	110.058	(3, 0, 0)
(B, 1)	2016-10-21	(18:40, 19:00)	188.54	186.878	(9, 2, 0)
(B, 1)	2016-10-22	(18:40, 19:00)	-	-	-
(B, 1)	2016-10-23	(18:40, 19:00)	124.76	115.556	(9, 1, 4)
(B, 1)	2016-10-24	(18:40, 19:00)	125.79	135.476	(6, 2, 0)
(B, 3)	2016-10-18	(08:00, 08:20)	126.4	96.7275	(3, 0, 0)
(B, 3)	2016-10-19	(08:00, 08:20)	70.3	97.3071	(6, 0, 0)
(B, 3)	2016-10-20	(08:00, 08:20)	93.88	95.9048	(6, 1, 0)
(B, 3)	2016-10-21	(08:00, 08:20)	85.64	92.3652	(4, 0, 0)
(B, 3)	2016-10-22	(08:00, 08:20)	103.23	99.8892	(4, 1, 0)
(B, 3)	2016-10-23	(08:00, 08:20)	106.67	97.2715	(8, 1, 0)
(B, 3)	2016-10-24	(08:00, 08:20)	120.81	115.315	(4, 2, 0)
(B, 3)	2016-10-18	(08:20, 08:40)	115.52	95.6306	(3, 0, 0)

(B, 3)	2016-10-19	(08:20, 08:40)	145.63	100.498	(6, 0, 0)
(B, 3)	2016-10-20	(08:20, 08:40)	87.78	97.6871	(6, 1, 0)
(B, 3)	2016-10-21	(08:20, 08:40)	99.17	97.5147	(4, 0, 0)
(B, 3)	2016-10-22	(08:20, 08:40)	111.49	100.613	(4, 1, 0)
(B, 3)	2016-10-23	(08:20, 08:40)	85.55	96.5507	(8, 1, 0)
(B, 3)	2016-10-24	(08:20, 08:40)	98.33	116.724	(4, 2, 0)
(B, 3)	2016-10-18	(08:40, 09:00)	98.86	98.1679	(3, 0, 0)
(B, 3)	2016-10-19	(08:40, 09:00)	120	99.5223	(6, 0, 0)
(B, 3)	2016-10-20	(08:40, 09:00)	103.01	97.8472	(6, 1, 0)
(B, 3)	2016-10-21	(08:40, 09:00)	102.09	95.5547	(4, 0, 0)
(B, 3)	2016-10-22	(08:40, 09:00)	65.55	97.1587	(4, 1, 0)
(B, 3)	2016-10-23	(08:40, 09:00)	91.14	96.6991	(8, 1, 0)
(B, 3)	2016-10-24	(08:40, 09:00)	107.49	117.483	(4, 2, 0)

(B, 3)	2016-10-18	(09:00, 09:20)	131.15	99.1605	(3, 0, 0)
(B, 3)	2016-10-19	(09:00, 09:20)	129.17	98.1697	(6, 0, 0)
(B, 3)	2016-10-20	(09:00, 09:20)	94.06	94.9163	(6, 1, 0)
(B, 3)	2016-10-21	(09:00, 09:20)	103.95	99.2009	(4, 0, 0)
(B, 3)	2016-10-22	(09:00, 09:20)	92.04	100.124	(4, 1, 0)
(B, 3)	2016-10-23	(09:00, 09:20)	64.23	97.5793	(8, 1, 0)
(B, 3)	2016-10-24	(09:00, 09:20)	147.68	122.096	(4, 2, 0)
(B, 3)	2016-10-18	(09:20, 09:40)	151.95	99.6144	(3, 0, 0)
(B, 3)	2016-10-19	(09:20, 09:40)	108.18	97.1835	(6, 0, 0)
(B, 3)	2016-10-20	(09:20, 09:40)	114.88	96.8646	(6, 1, 0)
(B, 3)	2016-10-21	(09:20, 09:40)	82.41	98.5959	(4, 0, 0)
(B, 3)	2016-10-22	(09:20, 09:40)	116.15	101.527	(4, 1, 0)
(B, 3)	2016-10-23	(09:20, 09:40)	120.49	97.3361	(8, 1, 0)

(B, 3)	2016-10-24	(09:20, 09:40)	124.16	127.849	(4, 2, 0)
(B, 3)	2016-10-18	(09:40, 10:00)	117.69	100.109	(3, 0, 0)
(B, 3)	2016-10-19	(09:40, 10:00)	112.14	98.6609	(6, 0, 0)
(B, 3)	2016-10-20	(09:40, 10:00)	70.85	93.8524	(6, 1, 0)
(B, 3)	2016-10-21	(09:40, 10:00)	107.51	99.4322	(4, 0, 0)
(B, 3)	2016-10-22	(09:40, 10:00)	114.06	100.073	(4, 1, 0)
(B, 3)	2016-10-23	(09:40, 10:00)	105.35	97.9472	(8, 1, 0)
(B, 3)	2016-10-24	(09:40, 10:00)	136.56	128.76	(4, 2, 0)
(B, 3)	2016-10-18	(17:00, 17:20)	80.49	96.6798	(3, 0, 0)
(B, 3)	2016-10-19	(17:00, 17:20)	106.34	104.676	(3, 1, 2)
(B, 3)	2016-10-20	(17:00, 17:20)	116.79	105.799	(3, 1, 2)
(B, 3)	2016-10-21	(17:00, 17:20)	84.76	104.147	(4, 2, 0)
(B, 3)	2016-10-22	(17:00, 17:20)	87.86	101.986	(3, 1, 1)

(B, 3)	2016-10-23	(17:00, 17:20)	88.62	104.588	(3, 2, 4)
(B, 3)	2016-10-24	(17:00, 17:20)	110.46	87.9087	(5, 2, 0)
(B, 3)	2016-10-18	(17:20, 17:40)	97.56	96.0225	(3, 0, 0)
(B, 3)	2016-10-19	(17:20, 17:40)	105.73	100.243	(3, 1, 2)
(B, 3)	2016-10-20	(17:20, 17:40)	93.09	104.97	(3, 1, 2)
(B, 3)	2016-10-21	(17:20, 17:40)	108.29	105.7	(4, 2, 0)
(B, 3)	2016-10-22	(17:20, 17:40)	103.59	101.837	(3, 1, 1)
(B, 3)	2016-10-23	(17:20, 17:40)	104.55	100.728	(3, 2, 4)
(B, 3)	2016-10-24	(17:20, 17:40)	57.83	83.8028	(5, 2, 0)
(B, 3)	2016-10-18	(17:40, 18:00)	104.47	97.7386	(3, 0, 0)
(B, 3)	2016-10-19	(17:40, 18:00)	101.06	100.498	(3, 1, 2)
(B, 3)	2016-10-20	(17:40, 18:00)	87.7	101.033	(3, 1, 2)
(B, 3)	2016-10-21	(17:40, 18:00)	130.19	100.462	(4, 2, 0)

(B, 3)	2016-10-22	(17:40, 18:00)	120.99	101	(3, 1, 1)
(B, 3)	2016-10-23	(17:40, 18:00)	90.96	100.482	(3, 2, 4)
(B, 3)	2016-10-24	(17:40, 18:00)	118.39	90.9407	(5, 2, 0)
(B, 3)	2016-10-18	(18:00, 18:20)	125.47	99.1419	(3, 0, 0)
(B, 3)	2016-10-19	(18:00, 18:20)	87.42	99.4833	(3, 1, 2)
(B, 3)	2016-10-20	(18:00, 18:20)	99.22	99.9889	(3, 1, 2)
(B, 3)	2016-10-21	(18:00, 18:20)	137.44	105.61	(4, 2, 0)
(B, 3)	2016-10-22	(18:00, 18:20)	95.22	99.7879	(3, 1, 1)
(B, 3)	2016-10-23	(18:00, 18:20)	109.01	101.173	(3, 2, 4)
(B, 3)	2016-10-24	(18:00, 18:20)	104.99	92.1596	(5, 2, 0)
(B, 3)	2016-10-18	(18:20, 18:40)	111.18	99.5881	(3, 0, 0)
(B, 3)	2016-10-19	(18:20, 18:40)	110.38	99.9527	(3, 1, 2)
(B, 3)	2016-10-20	(18:20, 18:40)	80.24	99.9018	(3, 1, 2)

(B, 3)	2016-10-21	(18:20, 18:40)	120.71	100.183	(4, 2, 0)
(B, 3)	2016-10-22	(18:20, 18:40)	72.77	99.5467	(3, 1, 1)
(B, 3)	2016-10-23	(18:20, 18:40)	81.63	100.278	(3, 2, 4)
(B, 3)	2016-10-24	(18:20, 18:40)	129.71	97.6569	(5, 2, 0)
(B, 3)	2016-10-18	(18:40, 19:00)	117.39	100.06	(3, 0, 0)
(B, 3)	2016-10-19	(18:40, 19:00)	80.84	99.3369	(3, 1, 2)
(B, 3)	2016-10-20	(18:40, 19:00)	106.83	99.3208	(3, 1, 2)
(B, 3)	2016-10-21	(18:40, 19:00)	60.02	96.7511	(4, 2, 0)
(B, 3)	2016-10-22	(18:40, 19:00)	88.6	99.3577	(3, 1, 1)
(B, 3)	2016-10-23	(18:40, 19:00)	104.62	100.953	(3, 2, 4)
(B, 3)	2016-10-24	(18:40, 19:00)	46.32	93.7879	(5, 2, 0)
(C, 1)	2016-10-18	(08:00, 08:20)	134.51	155.108	(7, 2, 3)
(C, 1)	2016-10-19	(08:00, 08:20)	158.18	138.289	(3, 2, 0)

(C, 1)	2016-10-20	(08:00, 08:20)	200.43	163.648	(6, 1, 0)
(C, 1)	2016-10-21	(08:00, 08:20)	147.11	168.763	(3, 1, 0)
(C, 1)	2016-10-22	(08:00, 08:20)	-	-	-
(C, 1)	2016-10-23	(08:00, 08:20)	159.65	159.525	(9, 2, 0)
(C, 1)	2016-10-24	(08:00, 08:20)	159.83	161.796	(9, 0, 0)
(C, 1)	2016-10-18	(08:20, 08:40)	183.06	159.964	(7, 2, 3)
(C, 1)	2016-10-19	(08:20, 08:40)	189.34	140.863	(3, 2, 0)
(C, 1)	2016-10-20	(08:20, 08:40)	178.94	160.612	(6, 1, 0)
(C, 1)	2016-10-21	(08:20, 08:40)	193.38	175.789	(3, 1, 0)
(C, 1)	2016-10-22	(08:20, 08:40)	169.57	162.293	(8, 2, 0)
(C, 1)	2016-10-23	(08:20, 08:40)	186.79	166.378	(9, 2, 0)
(C, 1)	2016-10-24	(08:20, 08:40)	156.72	154.126	(9, 0, 0)
(C, 1)	2016-10-18	(08:40, 09:00)	-	-	-

(C, 1)	2016-10-19	(08:40, 09:00)	140.53	139.789	(3, 2, 0)
(C, 1)	2016-10-20	(08:40, 09:00)	200.76	167.367	(6, 1, 0)
(C, 1)	2016-10-21	(08:40, 09:00)	-	-	-
(C, 1)	2016-10-22	(08:40, 09:00)	173.51	172.324	(8, 2, 0)
(C, 1)	2016-10-23	(08:40, 09:00)	153.82	170.73	(9, 2, 0)
(C, 1)	2016-10-24	(08:40, 09:00)	157.63	152.197	(9, 0, 0)
(C, 1)	2016-10-18	(09:00, 09:20)	158.02	145.393	(7, 2, 3)
(C, 1)	2016-10-19	(09:00, 09:20)	158.96	138.817	(3, 2, 0)
(C, 1)	2016-10-20	(09:00, 09:20)	138.53	164.239	(6, 1, 0)
(C, 1)	2016-10-21	(09:00, 09:20)	149.93	171.895	(3, 1, 0)
(C, 1)	2016-10-22	(09:00, 09:20)	187.36	176.289	(8, 2, 0)
(C, 1)	2016-10-23	(09:00, 09:20)	211.64	173.216	(9, 2, 0)
(C, 1)	2016-10-24	(09:00, 09:20)	-	-	-

(C, 1)	2016-10-18	(09:20, 09:40)	130.53	149.739	(7, 2, 3)
(C, 1)	2016-10-19	(09:20, 09:40)	250.19	143.71	(3, 2, 0)
(C, 1)	2016-10-20	(09:20, 09:40)	182.56	173.329	(6, 1, 0)
(C, 1)	2016-10-21	(09:20, 09:40)	182.18	172.85	(3, 1, 0)
(C, 1)	2016-10-22	(09:20, 09:40)	187.97	188.148	(8, 2, 0)
(C, 1)	2016-10-23	(09:20, 09:40)	153.05	166.611	(9, 2, 0)
(C, 1)	2016-10-24	(09:20, 09:40)	136.74	142.605	(9, 0, 0)
(C, 1)	2016-10-18	(09:40, 10:00)	151.99	138.379	(7, 2, 3)
(C, 1)	2016-10-19	(09:40, 10:00)	220.91	142.921	(3, 2, 0)
(C, 1)	2016-10-20	(09:40, 10:00)	152.07	176.384	(6, 1, 0)
(C, 1)	2016-10-21	(09:40, 10:00)	173.31	173.489	(3, 1, 0)
(C, 1)	2016-10-22	(09:40, 10:00)	-	-	-
(C, 1)	2016-10-23	(09:40, 10:00)	447.96	157.397	(9, 2, 0)

(C, 1)	2016-10-24	(09:40, 10:00)	205.9	144.73	(9, 0, 0)
(C, 1)	2016-10-18	(17:00, 17:20)	417.44	206.314	(7, 2, 0)
(C, 1)	2016-10-19	(17:00, 17:20)	233.4	236.028	(9, 2, 0)
(C, 1)	2016-10-20	(17:00, 17:20)	226.48	176.71	(3, 1, 0)
(C, 1)	2016-10-21	(17:00, 17:20)	265.45	249.433	(3, 2, 0)
(C, 1)	2016-10-22	(17:00, 17:20)	331.19	252.553	(4, 2, 0)
(C, 1)	2016-10-23	(17:00, 17:20)	227.86	208.809	(9, 1, 0)
(C, 1)	2016-10-24	(17:00, 17:20)	190.15	193.456	(9, 2, 0)
(C, 1)	2016-10-18	(17:20, 17:40)	185.55	217.209	(7, 2, 0)
(C, 1)	2016-10-19	(17:20, 17:40)	190.28	242.161	(9, 2, 0)
(C, 1)	2016-10-20	(17:20, 17:40)	312.25	175.773	(3, 1, 0)
(C, 1)	2016-10-21	(17:20, 17:40)	465.17	277.178	(3, 2, 0)
(C, 1)	2016-10-22	(17:20, 17:40)	240.21	247.979	(4, 2, 0)

(C, 1)	2016-10-23	(17:20, 17:40)	229.92	205.903	(9, 1, 0)
(C, 1)	2016-10-24	(17:20, 17:40)	154.65	203.486	(9, 2, 0)
(C, 1)	2016-10-18	(17:40, 18:00)	238.49	223.029	(7, 2, 0)
(C, 1)	2016-10-19	(17:40, 18:00)	198.43	257.993	(9, 2, 0)
(C, 1)	2016-10-20	(17:40, 18:00)	-	-	-
(C, 1)	2016-10-21	(17:40, 18:00)	482.11	303.064	(3, 2, 0)
(C, 1)	2016-10-22	(17:40, 18:00)	229.13	251.293	(4, 2, 0)
(C, 1)	2016-10-23	(17:40, 18:00)	221.27	220.794	(9, 1, 0)
(C, 1)	2016-10-24	(17:40, 18:00)	160.04	210.965	(9, 2, 0)
(C, 1)	2016-10-18	(18:00, 18:20)	179.15	220.224	(7, 2, 0)
(C, 1)	2016-10-19	(18:00, 18:20)	283.97	264.033	(9, 2, 0)
(C, 1)	2016-10-20	(18:00, 18:20)	174.53	180.727	(3, 1, 0)
(C, 1)	2016-10-21	(18:00, 18:20)	487.41	320.626	(3, 2, 0)

(C, 1)	2016-10-22	(18:00, 18:20)	271.24	267.432	(4, 2, 0)
(C, 1)	2016-10-23	(18:00, 18:20)	200.4	214.645	(9, 1, 0)
(C, 1)	2016-10-24	(18:00, 18:20)	271.2	217.532	(9, 2, 0)
(C, 1)	2016-10-18	(18:20, 18:40)	263.94	220.329	(7, 2, 0)
(C, 1)	2016-10-19	(18:20, 18:40)	314.9	261.829	(9, 2, 0)
(C, 1)	2016-10-20	(18:20, 18:40)	149.01	178.323	(3, 1, 0)
(C, 1)	2016-10-21	(18:20, 18:40)	490.26	335.041	(3, 2, 0)
(C, 1)	2016-10-22	(18:20, 18:40)	245.61	286.341	(4, 2, 0)
(C, 1)	2016-10-23	(18:20, 18:40)	224.13	219.144	(9, 1, 0)
(C, 1)	2016-10-24	(18:20, 18:40)	260.35	216.869	(9, 2, 0)
(C, 1)	2016-10-18	(18:40, 19:00)	390.85	236.728	(7, 2, 0)
(C, 1)	2016-10-19	(18:40, 19:00)	-	-	-
(C, 1)	2016-10-20	(18:40, 19:00)	165.61	178.793	(3, 1, 0)

(C, 1)	2016-10-21	(18:40, 19:00)	317.67	361.141	(3, 2, 0)
(C, 1)	2016-10-22	(18:40, 19:00)	233.02	285.347	(4, 2, 0)
(C, 1)	2016-10-23	(18:40, 19:00)	122.38	220.607	(9, 1, 0)
(C, 1)	2016-10-24	(18:40, 19:00)	-	-	-
(C, 3)	2016-10-18	(08:00, 08:20)	-	-	-
(C, 3)	2016-10-19	(08:00, 08:20)	188.63	126.674	(5, 0, 0)
(C, 3)	2016-10-20	(08:00, 08:20)	-	-	-
(C, 3)	2016-10-21	(08:00, 08:20)	-	-	-
(C, 3)	2016-10-22	(08:00, 08:20)	196.43	136.737	(7, 2, 0)
(C, 3)	2016-10-23	(08:00, 08:20)	-	-	-
(C, 3)	2016-10-24	(08:00, 08:20)	-	-	-
(C, 3)	2016-10-18	(08:20, 08:40)	233.55	166.354	(7, 2, 0)
(C, 3)	2016-10-19	(08:20, 08:40)	175.91	122.902	(5, 0, 0)

(C, 3)	2016-10-20	(08:20, 08:40)	141.54	170.39	(9, 1, 0)
(C, 3)	2016-10-21	(08:20, 08:40)	124.56	134.809	(3, 0, 0)
(C, 3)	2016-10-22	(08:20, 08:40)	-	-	-
(C, 3)	2016-10-23	(08:20, 08:40)	-	-	-
(C, 3)	2016-10-24	(08:20, 08:40)	-	-	-
(C, 3)	2016-10-18	(08:40, 09:00)	155.31	171.482	(7, 2, 0)
(C, 3)	2016-10-19	(08:40, 09:00)	202.04	126.203	(5, 0, 0)
(C, 3)	2016-10-20	(08:40, 09:00)	194.8	215.52	(9, 1, 0)
(C, 3)	2016-10-21	(08:40, 09:00)	214.37	123.391	(3, 0, 0)
(C, 3)	2016-10-22	(08:40, 09:00)	-	-	-
(C, 3)	2016-10-23	(08:40, 09:00)	-	-	-
(C, 3)	2016-10-24	(08:40, 09:00)	141.63	135.132	(8, 2, 2)
(C, 3)	2016-10-18	(09:00, 09:20)	216.72	187.825	(7, 2, 0)

(C, 3)	2016-10-19	(09:00, 09:20)	209.15	124.477	(5, 0, 0)
(C, 3)	2016-10-20	(09:00, 09:20)	148.35	120.021	(9, 1, 0)
(C, 3)	2016-10-21	(09:00, 09:20)	106.63	126.853	(3, 0, 0)
(C, 3)	2016-10-22	(09:00, 09:20)	134.16	165.948	(7, 2, 0)
(C, 3)	2016-10-23	(09:00, 09:20)	93.53	150.785	(9, 2, 0)
(C, 3)	2016-10-24	(09:00, 09:20)	-	-	-
(C, 3)	2016-10-18	(09:20, 09:40)	178.4	184.736	(7, 2, 0)
(C, 3)	2016-10-19	(09:20, 09:40)	143.38	128.659	(5, 0, 0)
(C, 3)	2016-10-20	(09:20, 09:40)	209.36	227.932	(9, 1, 0)
(C, 3)	2016-10-21	(09:20, 09:40)	155.75	124.958	(3, 0, 0)
(C, 3)	2016-10-22	(09:20, 09:40)	174.1	164.366	(7, 2, 0)
(C, 3)	2016-10-23	(09:20, 09:40)	211.69	179.749	(9, 2, 0)
(C, 3)	2016-10-24	(09:20, 09:40)	142.25	136.992	(8, 2, 2)

(C, 3)	2016-10-18	(09:40, 10:00)	144.3	202.115	(7, 2, 0)
(C, 3)	2016-10-19	(09:40, 10:00)	168.16	125.452	(5, 0, 0)
(C, 3)	2016-10-20	(09:40, 10:00)	-	-	-
(C, 3)	2016-10-21	(09:40, 10:00)	131.44	125.887	(3, 0, 0)
(C, 3)	2016-10-22	(09:40, 10:00)	135.19	177.907	(7, 2, 0)
(C, 3)	2016-10-23	(09:40, 10:00)	-	-	-
(C, 3)	2016-10-24	(09:40, 10:00)	169.27	173.981	(8, 2, 2)
(C, 3)	2016-10-18	(17:00, 17:20)	189.83	147.621	(7, 1, 1)
(C, 3)	2016-10-19	(17:00, 17:20)	108.94	158.694	(8, 2, 1)
(C, 3)	2016-10-20	(17:00, 17:20)	143.44	161.93	(4, 2, 0)
(C, 3)	2016-10-21	(17:00, 17:20)	-	-	-
(C, 3)	2016-10-22	(17:00, 17:20)	199.61	178.314	(8, 2, 0)
(C, 3)	2016-10-23	(17:00, 17:20)	-	-	-

(C, 3)	2016-10-24	(17:00, 17:20)	161.79	155.603	(3, 2, 0)
(C, 3)	2016-10-18	(17:20, 17:40)	-	-	-
(C, 3)	2016-10-19	(17:20, 17:40)	185.13	157.241	(8, 2, 1)
(C, 3)	2016-10-20	(17:20, 17:40)	138.56	170.44	(4, 2, 0)
(C, 3)	2016-10-21	(17:20, 17:40)	193.78	203.624	(9, 2, 0)
(C, 3)	2016-10-22	(17:20, 17:40)	167.41	192.209	(8, 2, 0)
(C, 3)	2016-10-23	(17:20, 17:40)	198.63	183.984	(6, 2, 0)
(C, 3)	2016-10-24	(17:20, 17:40)	165.87	161.002	(3, 2, 0)
(C, 3)	2016-10-18	(17:40, 18:00)	-	-	-
(C, 3)	2016-10-19	(17:40, 18:00)	-	-	-
(C, 3)	2016-10-20	(17:40, 18:00)	173.81	170.222	(4, 2, 0)
(C, 3)	2016-10-21	(17:40, 18:00)	254.98	196.185	(9, 2, 0)
(C, 3)	2016-10-22	(17:40, 18:00)	245.12	221.68	(8, 2, 0)

(C, 3)	2016-10-23	(17:40, 18:00)	200.92	185.852	(6, 2, 0)
(C, 3)	2016-10-24	(17:40, 18:00)	111.53	139.666	(3, 2, 0)
(C, 3)	2016-10-18	(18:00, 18:20)	84.47	132.166	(7, 1, 1)
(C, 3)	2016-10-19	(18:00, 18:20)	273.95	185.399	(8, 2, 1)
(C, 3)	2016-10-20	(18:00, 18:20)	-	-	-
(C, 3)	2016-10-21	(18:00, 18:20)	273.59	220.22	(9, 2, 0)
(C, 3)	2016-10-22	(18:00, 18:20)	167.08	236.747	(8, 2, 0)
(C, 3)	2016-10-23	(18:00, 18:20)	218.88	189.635	(6, 2, 0)
(C, 3)	2016-10-24	(18:00, 18:20)	-	-	-
(C, 3)	2016-10-18	(18:20, 18:40)	-	-	-
(C, 3)	2016-10-19	(18:20, 18:40)	123.48	169.219	(8, 2, 1)
(C, 3)	2016-10-20	(18:20, 18:40)	211.71	179.825	(4, 2, 0)
(C, 3)	2016-10-21	(18:20, 18:40)	216.57	201.991	(9, 2, 0)

(C, 3)	2016-10-22	(18:20, 18:40)	224.5	249.624	(8, 2, 0)
(C, 3)	2016-10-23	(18:20, 18:40)	-	-	-
(C, 3)	2016-10-24	(18:20, 18:40)	109.12	136.909	(3, 2, 0)
(C, 3)	2016-10-18	(18:40, 19:00)	204.89	146.48	(7, 1, 1)
(C, 3)	2016-10-19	(18:40, 19:00)	170.41	175.217	(8, 2, 1)
(C, 3)	2016-10-20	(18:40, 19:00)	-	-	-
(C, 3)	2016-10-21	(18:40, 19:00)	242.64	203.061	(9, 2, 0)
(C, 3)	2016-10-22	(18:40, 19:00)	241.57	206.099	(8, 2, 0)
(C, 3)	2016-10-23	(18:40, 19:00)	138.99	176.405	(6, 2, 0)
(C, 3)	2016-10-24	(18:40, 19:00)	-	-	-

Tabla 8.21: Tabla de predicciones de la segunda aproximación de predicciones del tiempo promedio de viaje (La fila con un '-' corresponde a una ruta, día e intervalo de tiempo de la que no se dispone dato real para comparar)

8.1.2.3 Segunda aproximación de predicciones del volumen de tráfico

*Nota: En la columna **Ruta**, dentro del paréntesis, el primer valor corresponde a la intersección y el segundo a la dirección de conducción (0: entrada, 1: salida)*

Ruta	Día	Intervalo de tiempo	Valor real	Predicho	Modelo ARIMA
(1, 0)	2016-10-18	(08:00, 08:20)	50	42.0	(7, 2, 3)
(1, 0)	2016-10-19	(08:00, 08:20)	48	40.0	(9, 0, 0)
(1, 0)	2016-10-20	(08:00, 08:20)	50	37.0	(9, 0, 0)
(1, 0)	2016-10-21	(08:00, 08:20)	37	41.0	(9, 0, 0)
(1, 0)	2016-10-22	(08:00, 08:20)	35	35.0	(9, 2, 0)
(1, 0)	2016-10-23	(08:00, 08:20)	30	30.0	(9, 2, 0)
(1, 0)	2016-10-24	(08:00, 08:20)	49	42.0	(7, 2, 3)
(1, 0)	2016-10-18	(08:20, 08:40)	41	43.0	(7, 2, 3)
(1, 0)	2016-10-19	(08:20, 08:40)	50	42.0	(9, 0, 0)
(1, 0)	2016-10-20	(08:20, 08:40)	49	38.0	(9, 0, 0)
(1, 0)	2016-10-21	(08:20, 08:40)	46	42.0	(9, 0, 0)
(1, 0)	2016-10-22	(08:20, 08:40)	52	39.0	(9, 2, 0)

(1, 0)	2016-10-23	(08:20, 08:40)	28	35.0	(9, 2, 0)
(1, 0)	2016-10-24	(08:20, 08:40)	47	41.0	(7, 2, 3)
(1, 0)	2016-10-18	(08:40, 09:00)	49	45.0	(7, 2, 3)
(1, 0)	2016-10-19	(08:40, 09:00)	50	44.0	(9, 0, 0)
(1, 0)	2016-10-20	(08:40, 09:00)	62	39.0	(9, 0, 0)
(1, 0)	2016-10-21	(08:40, 09:00)	46	43.0	(9, 0, 0)
(1, 0)	2016-10-22	(08:40, 09:00)	39	43.0	(9, 2, 0)
(1, 0)	2016-10-23	(08:40, 09:00)	37	39.0	(9, 2, 0)
(1, 0)	2016-10-24	(08:40, 09:00)	50	44.0	(7, 2, 3)
(1, 0)	2016-10-18	(09:00, 09:20)	61	45.0	(7, 2, 3)
(1, 0)	2016-10-19	(09:00, 09:20)	42	44.0	(9, 0, 0)
(1, 0)	2016-10-20	(09:00, 09:20)	38	40.0	(9, 0, 0)
(1, 0)	2016-10-21	(09:00, 09:20)	38	43.0	(9, 0, 0)
(1, 0)	2016-10-22	(09:00, 09:20)	57	48.0	(9, 2, 0)

(1, 0)	2016-10-23	(09:00, 09:20)	45	42.0	(9, 2, 0)
(1, 0)	2016-10-24	(09:00, 09:20)	52	43.0	(7, 2, 3)
(1, 0)	2016-10-18	(09:20, 09:40)	39	45.0	(7, 2, 3)
(1, 0)	2016-10-19	(09:20, 09:40)	38	44.0	(9, 0, 0)
(1, 0)	2016-10-20	(09:20, 09:40)	50	40.0	(9, 0, 0)
(1, 0)	2016-10-21	(09:20, 09:40)	45	43.0	(9, 0, 0)
(1, 0)	2016-10-22	(09:20, 09:40)	55	52.0	(9, 2, 0)
(1, 0)	2016-10-23	(09:20, 09:40)	35	46.0	(9, 2, 0)
(1, 0)	2016-10-24	(09:20, 09:40)	48	43.0	(7, 2, 3)
(1, 0)	2016-10-18	(09:40, 10:00)	46	46.0	(7, 2, 3)
(1, 0)	2016-10-19	(09:40, 10:00)	59	44.0	(9, 0, 0)
(1, 0)	2016-10-20	(09:40, 10:00)	44	40.0	(9, 0, 0)
(1, 0)	2016-10-21	(09:40, 10:00)	44	43.0	(9, 0, 0)

(1, 0)	2016-10-22	(09:40, 10:00)	41	56.0	(9, 2, 0)
(1, 0)	2016-10-23	(09:40, 10:00)	42	49.0	(9, 2, 0)
(1, 0)	2016-10-24	(09:40, 10:00)	58	44.0	(7, 2, 3)
(1, 0)	2016-10-18	(17:00, 17:20)	40	39.0	(8, 0, 3)
(1, 0)	2016-10-19	(17:00, 17:20)	32	37.0	(3, 2, 0)
(1, 0)	2016-10-20	(17:00, 17:20)	38	35.0	(3, 2, 0)
(1, 0)	2016-10-21	(17:00, 17:20)	56	57.0	(5, 0, 2)
(1, 0)	2016-10-22	(17:00, 17:20)	49	42.0	(9, 0, 4)
(1, 0)	2016-10-23	(17:00, 17:20)	43	48.0	(6, 0, 4)
(1, 0)	2016-10-24	(17:00, 17:20)	32	33.0	(9, 0, 4)
(1, 0)	2016-10-18	(17:20, 17:40)	42	35.0	(8, 0, 3)
(1, 0)	2016-10-19	(17:20, 17:40)	29	33.0	(3, 2, 0)
(1, 0)	2016-10-20	(17:20, 17:40)	45	33.0	(3, 2, 0)

(1, 0)	2016-10-21	(17:20, 17:40)	40	53.0	(5, 0, 2)
(1, 0)	2016-10-22	(17:20, 17:40)	48	35.0	(9, 0, 4)
(1, 0)	2016-10-23	(17:20, 17:40)	37	44.0	(6, 0, 4)
(1, 0)	2016-10-24	(17:20, 17:40)	39	30.0	(9, 0, 4)
(1, 0)	2016-10-18	(17:40, 18:00)	23	29.0	(8, 0, 3)
(1, 0)	2016-10-19	(17:40, 18:00)	34	32.0	(3, 2, 0)
(1, 0)	2016-10-20	(17:40, 18:00)	35	30.0	(3, 2, 0)
(1, 0)	2016-10-21	(17:40, 18:00)	54	49.0	(5, 0, 2)
(1, 0)	2016-10-22	(17:40, 18:00)	30	32.0	(9, 0, 4)
(1, 0)	2016-10-23	(17:40, 18:00)	46	38.0	(6, 0, 4)
(1, 0)	2016-10-24	(17:40, 18:00)	29	26.0	(9, 0, 4)
(1, 0)	2016-10-18	(18:00, 18:20)	27	25.0	(8, 0, 3)
(1, 0)	2016-10-19	(18:00, 18:20)	27	30.0	(3, 2, 0)

(1, 0)	2016-10-20	(18:00, 18:20)	27	29.0	(3, 2, 0)
(1, 0)	2016-10-21	(18:00, 18:20)	38	46.0	(5, 0, 2)
(1, 0)	2016-10-22	(18:00, 18:20)	27	25.0	(9, 0, 4)
(1, 0)	2016-10-23	(18:00, 18:20)	44	34.0	(6, 0, 4)
(1, 0)	2016-10-24	(18:00, 18:20)	15	20.0	(9, 0, 4)
(1, 0)	2016-10-18	(18:20, 18:40)	23	21.0	(8, 0, 3)
(1, 0)	2016-10-19	(18:20, 18:40)	19	28.0	(3, 2, 0)
(1, 0)	2016-10-20	(18:20, 18:40)	16	26.0	(3, 2, 0)
(1, 0)	2016-10-21	(18:20, 18:40)	41	43.0	(5, 0, 2)
(1, 0)	2016-10-22	(18:20, 18:40)	23	23.0	(9, 0, 4)
(1, 0)	2016-10-23	(18:20, 18:40)	30	32.0	(6, 0, 4)
(1, 0)	2016-10-24	(18:20, 18:40)	18	19.0	(9, 0, 4)
(1, 0)	2016-10-18	(18:40, 19:00)	15	20.0	(8, 0, 3)

(1, 0)	2016-10-19	(18:40, 19:00)	31	26.0	(3, 2, 0)
(1, 0)	2016-10-20	(18:40, 19:00)	17	24.0	(3, 2, 0)
(1, 0)	2016-10-21	(18:40, 19:00)	40	42.0	(5, 0, 2)
(1, 0)	2016-10-22	(18:40, 19:00)	14	20.0	(9, 0, 4)
(1, 0)	2016-10-23	(18:40, 19:00)	22	29.0	(6, 0, 4)
(1, 0)	2016-10-24	(18:40, 19:00)	13	16.0	(9, 0, 4)
(1, 1)	2016-10-18	(08:00, 08:20)	93	97.0	(6, 2, 3)
(1, 1)	2016-10-19	(08:00, 08:20)	118	100.0	(8, 2, 4)
(1, 1)	2016-10-20	(08:00, 08:20)	106	102.0	(3, 2, 1)
(1, 1)	2016-10-21	(08:00, 08:20)	106	97.0	(9, 2, 0)
(1, 1)	2016-10-22	(08:00, 08:20)	79	75.0	(9, 2, 0)
(1, 1)	2016-10-23	(08:00, 08:20)	57	67.0	(8, 2, 0)
(1, 1)	2016-10-24	(08:00, 08:20)	137	126.0	(8, 1, 3)

(1, 1)	2016-10-18	(08:20, 08:40)	134	100.0	(6, 2, 3)
(1, 1)	2016-10-19	(08:20, 08:40)	111	93.0	(8, 2, 4)
(1, 1)	2016-10-20	(08:20, 08:40)	100	100.0	(3, 2, 1)
(1, 1)	2016-10-21	(08:20, 08:40)	107	103.0	(9, 2, 0)
(1, 1)	2016-10-22	(08:20, 08:40)	89	83.0	(9, 2, 0)
(1, 1)	2016-10-23	(08:20, 08:40)	68	74.0	(8, 2, 0)
(1, 1)	2016-10-24	(08:20, 08:40)	151	118.0	(8, 1, 3)
(1, 1)	2016-10-18	(08:40, 09:00)	121	98.0	(6, 2, 3)
(1, 1)	2016-10-19	(08:40, 09:00)	109	94.0	(8, 2, 4)
(1, 1)	2016-10-20	(08:40, 09:00)	114	99.0	(3, 2, 1)
(1, 1)	2016-10-21	(08:40, 09:00)	94	116.0	(9, 2, 0)
(1, 1)	2016-10-22	(08:40, 09:00)	86	87.0	(9, 2, 0)
(1, 1)	2016-10-23	(08:40, 09:00)	84	80.0	(8, 2, 0)

(1, 1)	2016-10-24	(08:40, 09:00)	135	109.0	(8, 1, 3)
(1, 1)	2016-10-18	(09:00, 09:20)	86	99.0	(6, 2, 3)
(1, 1)	2016-10-19	(09:00, 09:20)	121	98.0	(8, 2, 4)
(1, 1)	2016-10-20	(09:00, 09:20)	118	101.0	(3, 2, 1)
(1, 1)	2016-10-21	(09:00, 09:20)	107	128.0	(9, 2, 0)
(1, 1)	2016-10-22	(09:00, 09:20)	87	96.0	(9, 2, 0)
(1, 1)	2016-10-23	(09:00, 09:20)	80	87.0	(8, 2, 0)
(1, 1)	2016-10-24	(09:00, 09:20)	128	115.0	(8, 1, 3)
(1, 1)	2016-10-18	(09:20, 09:40)	108	102.0	(6, 2, 3)
(1, 1)	2016-10-19	(09:20, 09:40)	92	94.0	(8, 2, 4)
(1, 1)	2016-10-20	(09:20, 09:40)	106	101.0	(3, 2, 1)
(1, 1)	2016-10-21	(09:20, 09:40)	132	139.0	(9, 2, 0)
(1, 1)	2016-10-22	(09:20, 09:40)	80	105.0	(9, 2, 0)

(1, 1)	2016-10-23	(09:20, 09:40)	105	95.0	(8, 2, 0)
(1, 1)	2016-10-24	(09:20, 09:40)	116	115.0	(8, 1, 3)
(1, 1)	2016-10-18	(09:40, 10:00)	113	96.0	(6, 2, 3)
(1, 1)	2016-10-19	(09:40, 10:00)	111	97.0	(8, 2, 4)
(1, 1)	2016-10-20	(09:40, 10:00)	115	101.0	(3, 2, 1)
(1, 1)	2016-10-21	(09:40, 10:00)	116	150.0	(9, 2, 0)
(1, 1)	2016-10-22	(09:40, 10:00)	100	112.0	(9, 2, 0)
(1, 1)	2016-10-23	(09:40, 10:00)	96	100.0	(8, 2, 0)
(1, 1)	2016-10-24	(09:40, 10:00)	88	105.0	(8, 1, 3)
(1, 1)	2016-10-18	(17:00, 17:20)	93	96.0	(7, 1, 4)
(1, 1)	2016-10-19	(17:00, 17:20)	90	85.0	(9, 0, 0)
(1, 1)	2016-10-20	(17:00, 17:20)	89	101.0	(7, 0, 4)
(1, 1)	2016-10-21	(17:00, 17:20)	116	107.0	(9, 0, 0)

(1, 1)	2016-10-22	(17:00, 17:20)	74	78.0	(4, 2, 0)
(1, 1)	2016-10-23	(17:00, 17:20)	113	102.0	(8, 0, 3)
(1, 1)	2016-10-24	(17:00, 17:20)	108	101.0	(8, 0, 0)
(1, 1)	2016-10-18	(17:20, 17:40)	72	96.0	(7, 1, 4)
(1, 1)	2016-10-19	(17:20, 17:40)	108	86.0	(9, 0, 0)
(1, 1)	2016-10-20	(17:20, 17:40)	99	103.0	(7, 0, 4)
(1, 1)	2016-10-21	(17:20, 17:40)	149	109.0	(9, 0, 0)
(1, 1)	2016-10-22	(17:20, 17:40)	104	75.0	(4, 2, 0)
(1, 1)	2016-10-23	(17:20, 17:40)	105	98.0	(8, 0, 3)
(1, 1)	2016-10-24	(17:20, 17:40)	112	101.0	(8, 0, 0)
(1, 1)	2016-10-18	(17:40, 18:00)	134	83.0	(7, 1, 4)
(1, 1)	2016-10-19	(17:40, 18:00)	112	84.0	(9, 0, 0)
(1, 1)	2016-10-20	(17:40, 18:00)	113	89.0	(7, 0, 4)

(1, 1)	2016-10-21	(17:40, 18:00)	144	103.0	(9, 0, 0)
(1, 1)	2016-10-22	(17:40, 18:00)	88	73.0	(4, 2, 0)
(1, 1)	2016-10-23	(17:40, 18:00)	106	93.0	(8, 0, 3)
(1, 1)	2016-10-24	(17:40, 18:00)	134	98.0	(8, 0, 0)
(1, 1)	2016-10-18	(18:00, 18:20)	73	81.0	(7, 1, 4)
(1, 1)	2016-10-19	(18:00, 18:20)	63	81.0	(9, 0, 0)
(1, 1)	2016-10-20	(18:00, 18:20)	69	87.0	(7, 0, 4)
(1, 1)	2016-10-21	(18:00, 18:20)	74	100.0	(9, 0, 0)
(1, 1)	2016-10-22	(18:00, 18:20)	47	64.0	(4, 2, 0)
(1, 1)	2016-10-23	(18:00, 18:20)	77	88.0	(8, 0, 3)
(1, 1)	2016-10-24	(18:00, 18:20)	69	96.0	(8, 0, 0)
(1, 1)	2016-10-18	(18:20, 18:40)	84	81.0	(7, 1, 4)
(1, 1)	2016-10-19	(18:20, 18:40)	76	79.0	(9, 0, 0)

(1, 1)	2016-10-20	(18:20, 18:40)	69	86.0	(7, 0, 4)
(1, 1)	2016-10-21	(18:20, 18:40)	81	95.0	(9, 0, 0)
(1, 1)	2016-10-22	(18:20, 18:40)	46	56.0	(4, 2, 0)
(1, 1)	2016-10-23	(18:20, 18:40)	91	86.0	(8, 0, 3)
(1, 1)	2016-10-24	(18:20, 18:40)	84	92.0	(8, 0, 0)
(1, 1)	2016-10-18	(18:40, 19:00)	39	73.0	(7, 1, 4)
(1, 1)	2016-10-19	(18:40, 19:00)	73	78.0	(9, 0, 0)
(1, 1)	2016-10-20	(18:40, 19:00)	71	78.0	(7, 0, 4)
(1, 1)	2016-10-21	(18:40, 19:00)	98	92.0	(9, 0, 0)
(1, 1)	2016-10-22	(18:40, 19:00)	54	52.0	(4, 2, 0)
(1, 1)	2016-10-23	(18:40, 19:00)	49	80.0	(8, 0, 3)
(1, 1)	2016-10-24	(18:40, 19:00)	69	89.0	(8, 0, 0)
(2, 0)	2016-10-18	(08:00, 08:20)	120	101.0	(3, 0, 4)

(2, 0)	2016-10-19	(08:00, 08:20)	127	106.0	(9, 2, 3)
(2, 0)	2016-10-20	(08:00, 08:20)	104	113.0	(4, 0, 0)
(2, 0)	2016-10-21	(08:00, 08:20)	97	94.0	(3, 0, 4)
(2, 0)	2016-10-22	(08:00, 08:20)	80	60.0	(9, 2, 0)
(2, 0)	2016-10-23	(08:00, 08:20)	48	47.0	(6, 2, 0)
(2, 0)	2016-10-24	(08:00, 08:20)	122	118.0	(3, 0, 0)
(2, 0)	2016-10-18	(08:20, 08:40)	130	104.0	(3, 0, 4)
(2, 0)	2016-10-19	(08:20, 08:40)	128	110.0	(9, 2, 3)
(2, 0)	2016-10-20	(08:20, 08:40)	121	114.0	(4, 0, 0)
(2, 0)	2016-10-21	(08:20, 08:40)	117	99.0	(3, 0, 4)
(2, 0)	2016-10-22	(08:20, 08:40)	76	65.0	(9, 2, 0)
(2, 0)	2016-10-23	(08:20, 08:40)	61	53.0	(6, 2, 0)
(2, 0)	2016-10-24	(08:20, 08:40)	122	115.0	(3, 0, 0)

(2, 0)	2016-10-18	(08:40, 09:00)	116	105.0	(3, 0, 4)
(2, 0)	2016-10-19	(08:40, 09:00)	114	111.0	(9, 2, 3)
(2, 0)	2016-10-20	(08:40, 09:00)	106	112.0	(4, 0, 0)
(2, 0)	2016-10-21	(08:40, 09:00)	104	101.0	(3, 0, 4)
(2, 0)	2016-10-22	(08:40, 09:00)	96	71.0	(9, 2, 0)
(2, 0)	2016-10-23	(08:40, 09:00)	57	60.0	(6, 2, 0)
(2, 0)	2016-10-24	(08:40, 09:00)	108	109.0	(3, 0, 0)
(2, 0)	2016-10-18	(09:00, 09:20)	94	102.0	(3, 0, 4)
(2, 0)	2016-10-19	(09:00, 09:20)	103	109.0	(9, 2, 3)
(2, 0)	2016-10-20	(09:00, 09:20)	125	107.0	(4, 0, 0)
(2, 0)	2016-10-21	(09:00, 09:20)	105	100.0	(3, 0, 4)
(2, 0)	2016-10-22	(09:00, 09:20)	74	78.0	(9, 2, 0)
(2, 0)	2016-10-23	(09:00, 09:20)	66	66.0	(6, 2, 0)

(2, 0)	2016-10-24	(09:00, 09:20)	113	105.0	(3, 0, 0)
(2, 0)	2016-10-18	(09:20, 09:40)	90	97.0	(3, 0, 4)
(2, 0)	2016-10-19	(09:20, 09:40)	100	103.0	(9, 2, 3)
(2, 0)	2016-10-20	(09:20, 09:40)	108	102.0	(4, 0, 0)
(2, 0)	2016-10-21	(09:20, 09:40)	93	97.0	(3, 0, 4)
(2, 0)	2016-10-22	(09:20, 09:40)	80	84.0	(9, 2, 0)
(2, 0)	2016-10-23	(09:20, 09:40)	67	72.0	(6, 2, 0)
(2, 0)	2016-10-24	(09:20, 09:40)	101	100.0	(3, 0, 0)
(2, 0)	2016-10-18	(09:40, 10:00)	74	92.0	(3, 0, 4)
(2, 0)	2016-10-19	(09:40, 10:00)	91	98.0	(9, 2, 3)
(2, 0)	2016-10-20	(09:40, 10:00)	72	97.0	(4, 0, 0)
(2, 0)	2016-10-21	(09:40, 10:00)	83	92.0	(3, 0, 4)
(2, 0)	2016-10-22	(09:40, 10:00)	84	91.0	(9, 2, 0)

(2, 0)	2016-10-23	(09:40, 10:00)	78	78.0	(6, 2, 0)
(2, 0)	2016-10-24	(09:40, 10:00)	79	96.0	(3, 0, 0)
(2, 0)	2016-10-18	(17:00, 17:20)	69	78.0	(9, 2, 2)
(2, 0)	2016-10-19	(17:00, 17:20)	88	64.0	(7, 0, 0)
(2, 0)	2016-10-20	(17:00, 17:20)	77	72.0	(9, 2, 2)
(2, 0)	2016-10-21	(17:00, 17:20)	90	82.0	(5, 1, 4)
(2, 0)	2016-10-22	(17:00, 17:20)	68	67.0	(9, 2, 2)
(2, 0)	2016-10-23	(17:00, 17:20)	88	63.0	(9, 0, 0)
(2, 0)	2016-10-24	(17:00, 17:20)	74	79.0	(9, 2, 2)
(2, 0)	2016-10-18	(17:20, 17:40)	55	76.0	(9, 2, 2)
(2, 0)	2016-10-19	(17:20, 17:40)	56	62.0	(7, 0, 0)
(2, 0)	2016-10-20	(17:20, 17:40)	57	67.0	(9, 2, 2)
(2, 0)	2016-10-21	(17:20, 17:40)	75	76.0	(5, 1, 4)

(2, 0)	2016-10-22	(17:20, 17:40)	48	61.0	(9, 2, 2)
(2, 0)	2016-10-23	(17:20, 17:40)	56	58.0	(9, 0, 0)
(2, 0)	2016-10-24	(17:20, 17:40)	60	76.0	(9, 2, 2)
(2, 0)	2016-10-18	(17:40, 18:00)	67	70.0	(9, 2, 2)
(2, 0)	2016-10-19	(17:40, 18:00)	64	62.0	(7, 0, 0)
(2, 0)	2016-10-20	(17:40, 18:00)	66	59.0	(9, 2, 2)
(2, 0)	2016-10-21	(17:40, 18:00)	73	72.0	(5, 1, 4)
(2, 0)	2016-10-22	(17:40, 18:00)	48	53.0	(9, 2, 2)
(2, 0)	2016-10-23	(17:40, 18:00)	54	52.0	(9, 0, 0)
(2, 0)	2016-10-24	(17:40, 18:00)	61	69.0	(9, 2, 2)
(2, 0)	2016-10-18	(18:00, 18:20)	59	70.0	(9, 2, 2)
(2, 0)	2016-10-19	(18:00, 18:20)	65	62.0	(7, 0, 0)
(2, 0)	2016-10-20	(18:00, 18:20)	46	57.0	(9, 2, 2)

(2, 0)	2016-10-21	(18:00, 18:20)	67	66.0	(5, 1, 4)
(2, 0)	2016-10-22	(18:00, 18:20)	41	50.0	(9, 2, 2)
(2, 0)	2016-10-23	(18:00, 18:20)	49	51.0	(9, 0, 0)
(2, 0)	2016-10-24	(18:00, 18:20)	53	67.0	(9, 2, 2)
(2, 0)	2016-10-18	(18:20, 18:40)	53	71.0	(9, 2, 2)
(2, 0)	2016-10-19	(18:20, 18:40)	54	61.0	(7, 0, 0)
(2, 0)	2016-10-20	(18:20, 18:40)	35	56.0	(9, 2, 2)
(2, 0)	2016-10-21	(18:20, 18:40)	67	64.0	(5, 1, 4)
(2, 0)	2016-10-22	(18:20, 18:40)	39	48.0	(9, 2, 2)
(2, 0)	2016-10-23	(18:20, 18:40)	45	50.0	(9, 0, 0)
(2, 0)	2016-10-24	(18:20, 18:40)	45	65.0	(9, 2, 2)
(2, 0)	2016-10-18	(18:40, 19:00)	46	69.0	(9, 2, 2)
(2, 0)	2016-10-19	(18:40, 19:00)	56	60.0	(7, 0, 0)

(2, 0)	2016-10-20	(18:40, 19:00)	43	55.0	(9, 2, 2)
(2, 0)	2016-10-21	(18:40, 19:00)	54	61.0	(5, 1, 4)
(2, 0)	2016-10-22	(18:40, 19:00)	30	45.0	(9, 2, 2)
(2, 0)	2016-10-23	(18:40, 19:00)	50	49.0	(9, 0, 0)
(2, 0)	2016-10-24	(18:40, 19:00)	37	63.0	(9, 2, 2)
(3, 0)	2016-10-18	(08:00, 08:20)	198	160.0	(3, 0, 0)
(3, 0)	2016-10-19	(08:00, 08:20)	137	150.0	(3, 0, 0)
(3, 0)	2016-10-20	(08:00, 08:20)	163	130.0	(3, 2, 4)
(3, 0)	2016-10-21	(08:00, 08:20)	175	152.0	(3, 0, 0)
(3, 0)	2016-10-22	(08:00, 08:20)	112	93.0	(4, 2, 0)
(3, 0)	2016-10-23	(08:00, 08:20)	86	76.0	(6, 0, 4)
(3, 0)	2016-10-24	(08:00, 08:20)	164	166.0	(9, 0, 4)
(3, 0)	2016-10-18	(08:20, 08:40)	153	160.0	(3, 0, 0)

(3, 0)	2016-10-19	(08:20, 08:40)	152	148.0	(3, 0, 0)
(3, 0)	2016-10-20	(08:20, 08:40)	158	136.0	(3, 2, 4)
(3, 0)	2016-10-21	(08:20, 08:40)	152	151.0	(3, 0, 0)
(3, 0)	2016-10-22	(08:20, 08:40)	123	102.0	(4, 2, 0)
(3, 0)	2016-10-23	(08:20, 08:40)	87	82.0	(6, 0, 4)
(3, 0)	2016-10-24	(08:20, 08:40)	179	163.0	(9, 0, 4)
(3, 0)	2016-10-18	(08:40, 09:00)	125	155.0	(3, 0, 0)
(3, 0)	2016-10-19	(08:40, 09:00)	127	144.0	(3, 0, 0)
(3, 0)	2016-10-20	(08:40, 09:00)	143	137.0	(3, 2, 4)
(3, 0)	2016-10-21	(08:40, 09:00)	140	147.0	(3, 0, 0)
(3, 0)	2016-10-22	(08:40, 09:00)	108	110.0	(4, 2, 0)
(3, 0)	2016-10-23	(08:40, 09:00)	86	88.0	(6, 0, 4)
(3, 0)	2016-10-24	(08:40, 09:00)	169	155.0	(9, 0, 4)

(3, 0)	2016-10-18	(09:00, 09:20)	176	150.0	(3, 0, 0)
(3, 0)	2016-10-19	(09:00, 09:20)	145	140.0	(3, 0, 0)
(3, 0)	2016-10-20	(09:00, 09:20)	138	136.0	(3, 2, 4)
(3, 0)	2016-10-21	(09:00, 09:20)	140	143.0	(3, 0, 0)
(3, 0)	2016-10-22	(09:00, 09:20)	136	116.0	(4, 2, 0)
(3, 0)	2016-10-23	(09:00, 09:20)	94	92.0	(6, 0, 4)
(3, 0)	2016-10-24	(09:00, 09:20)	145	145.0	(9, 0, 4)
(3, 0)	2016-10-18	(09:20, 09:40)	130	146.0	(3, 0, 0)
(3, 0)	2016-10-19	(09:20, 09:40)	147	136.0	(3, 0, 0)
(3, 0)	2016-10-20	(09:20, 09:40)	132	134.0	(3, 2, 4)
(3, 0)	2016-10-21	(09:20, 09:40)	162	138.0	(3, 0, 0)
(3, 0)	2016-10-22	(09:20, 09:40)	116	124.0	(4, 2, 0)
(3, 0)	2016-10-23	(09:20, 09:40)	92	96.0	(6, 0, 4)

(3, 0)	2016-10-24	(09:20, 09:40)	160	133.0	(9, 0, 4)
(3, 0)	2016-10-18	(09:40, 10:00)	127	141.0	(3, 0, 0)
(3, 0)	2016-10-19	(09:40, 10:00)	140	132.0	(3, 0, 0)
(3, 0)	2016-10-20	(09:40, 10:00)	117	132.0	(3, 2, 4)
(3, 0)	2016-10-21	(09:40, 10:00)	116	135.0	(3, 0, 0)
(3, 0)	2016-10-22	(09:40, 10:00)	111	133.0	(4, 2, 0)
(3, 0)	2016-10-23	(09:40, 10:00)	103	98.0	(6, 0, 4)
(3, 0)	2016-10-24	(09:40, 10:00)	110	122.0	(9, 0, 4)
(3, 0)	2016-10-18	(17:00, 17:20)	109	117.0	(6, 1, 3)
(3, 0)	2016-10-19	(17:00, 17:20)	120	98.0	(8, 0, 4)
(3, 0)	2016-10-20	(17:00, 17:20)	129	133.0	(9, 0, 4)
(3, 0)	2016-10-21	(17:00, 17:20)	123	150.0	(9, 0, 4)
(3, 0)	2016-10-22	(17:00, 17:20)	97	103.0	(9, 1, 4)

(3, 0)	2016-10-23	(17:00, 17:20)	107	113.0	(9, 0, 4)
(3, 0)	2016-10-24	(17:00, 17:20)	133	116.0	(9, 0, 4)
(3, 0)	2016-10-18	(17:20, 17:40)	98	112.0	(6, 1, 3)
(3, 0)	2016-10-19	(17:20, 17:40)	105	91.0	(8, 0, 4)
(3, 0)	2016-10-20	(17:20, 17:40)	106	129.0	(9, 0, 4)
(3, 0)	2016-10-21	(17:20, 17:40)	118	138.0	(9, 0, 4)
(3, 0)	2016-10-22	(17:20, 17:40)	114	94.0	(9, 1, 4)
(3, 0)	2016-10-23	(17:20, 17:40)	124	106.0	(9, 0, 4)
(3, 0)	2016-10-24	(17:20, 17:40)	107	112.0	(9, 0, 4)
(3, 0)	2016-10-18	(17:40, 18:00)	88	107.0	(6, 1, 3)
(3, 0)	2016-10-19	(17:40, 18:00)	96	85.0	(8, 0, 4)
(3, 0)	2016-10-20	(17:40, 18:00)	87	122.0	(9, 0, 4)
(3, 0)	2016-10-21	(17:40, 18:00)	112	124.0	(9, 0, 4)

(3, 0)	2016-10-22	(17:40, 18:00)	81	87.0	(9, 1, 4)
(3, 0)	2016-10-23	(17:40, 18:00)	82	98.0	(9, 0, 4)
(3, 0)	2016-10-24	(17:40, 18:00)	110	105.0	(9, 0, 4)
(3, 0)	2016-10-18	(18:00, 18:20)	94	102.0	(6, 1, 3)
(3, 0)	2016-10-19	(18:00, 18:20)	79	80.0	(8, 0, 4)
(3, 0)	2016-10-20	(18:00, 18:20)	82	116.0	(9, 0, 4)
(3, 0)	2016-10-21	(18:00, 18:20)	91	113.0	(9, 0, 4)
(3, 0)	2016-10-22	(18:00, 18:20)	78	81.0	(9, 1, 4)
(3, 0)	2016-10-23	(18:00, 18:20)	65	93.0	(9, 0, 4)
(3, 0)	2016-10-24	(18:00, 18:20)	75	101.0	(9, 0, 4)
(3, 0)	2016-10-18	(18:20, 18:40)	66	98.0	(6, 1, 3)
(3, 0)	2016-10-19	(18:20, 18:40)	68	77.0	(8, 0, 4)
(3, 0)	2016-10-20	(18:20, 18:40)	74	109.0	(9, 0, 4)

(3, 0)	2016-10-21	(18:20, 18:40)	77	102.0	(9, 0, 4)
(3, 0)	2016-10-22	(18:20, 18:40)	71	76.0	(9, 1, 4)
(3, 0)	2016-10-23	(18:20, 18:40)	80	88.0	(9, 0, 4)
(3, 0)	2016-10-24	(18:20, 18:40)	62	96.0	(9, 0, 4)
(3, 0)	2016-10-18	(18:40, 19:00)	51	97.0	(6, 1, 3)
(3, 0)	2016-10-19	(18:40, 19:00)	68	76.0	(8, 0, 4)
(3, 0)	2016-10-20	(18:40, 19:00)	58	103.0	(9, 0, 4)
(3, 0)	2016-10-21	(18:40, 19:00)	75	93.0	(9, 0, 4)
(3, 0)	2016-10-22	(18:40, 19:00)	58	72.0	(9, 1, 4)
(3, 0)	2016-10-23	(18:40, 19:00)	74	85.0	(9, 0, 4)
(3, 0)	2016-10-24	(18:40, 19:00)	49	93.0	(9, 0, 4)
(3, 1)	2016-10-18	(08:00, 08:20)	119	99.0	(9, 2, 0)
(3, 1)	2016-10-19	(08:00, 08:20)	115	106.0	(9, 1, 4)

(3, 1)	2016-10-20	(08:00, 08:20)	87	106.0	(9, 1, 4)
(3, 1)	2016-10-21	(08:00, 08:20)	102	86.0	(9, 2, 0)
(3, 1)	2016-10-22	(08:00, 08:20)	69	69.0	(7, 1, 2)
(3, 1)	2016-10-23	(08:00, 08:20)	61	59.0	(5, 2, 0)
(3, 1)	2016-10-24	(08:00, 08:20)	174	109.0	(9, 2, 0)
(3, 1)	2016-10-18	(08:20, 08:40)	101	104.0	(9, 2, 0)
(3, 1)	2016-10-19	(08:20, 08:40)	115	102.0	(9, 1, 4)
(3, 1)	2016-10-20	(08:20, 08:40)	120	96.0	(9, 1, 4)
(3, 1)	2016-10-21	(08:20, 08:40)	114	89.0	(9, 2, 0)
(3, 1)	2016-10-22	(08:20, 08:40)	61	73.0	(7, 1, 2)
(3, 1)	2016-10-23	(08:20, 08:40)	81	66.0	(5, 2, 0)
(3, 1)	2016-10-24	(08:20, 08:40)	122	114.0	(9, 2, 0)
(3, 1)	2016-10-18	(08:40, 09:00)	136	113.0	(9, 2, 0)

(3, 1)	2016-10-19	(08:40, 09:00)	130	96.0	(9, 1, 4)
(3, 1)	2016-10-20	(08:40, 09:00)	134	88.0	(9, 1, 4)
(3, 1)	2016-10-21	(08:40, 09:00)	118	98.0	(9, 2, 0)
(3, 1)	2016-10-22	(08:40, 09:00)	96	73.0	(7, 1, 2)
(3, 1)	2016-10-23	(08:40, 09:00)	70	75.0	(5, 2, 0)
(3, 1)	2016-10-24	(08:40, 09:00)	164	128.0	(9, 2, 0)
(3, 1)	2016-10-18	(09:00, 09:20)	120	125.0	(9, 2, 0)
(3, 1)	2016-10-19	(09:00, 09:20)	139	97.0	(9, 1, 4)
(3, 1)	2016-10-20	(09:00, 09:20)	117	89.0	(9, 1, 4)
(3, 1)	2016-10-21	(09:00, 09:20)	123	109.0	(9, 2, 0)
(3, 1)	2016-10-22	(09:00, 09:20)	81	76.0	(7, 1, 2)
(3, 1)	2016-10-23	(09:00, 09:20)	94	82.0	(5, 2, 0)
(3, 1)	2016-10-24	(09:00, 09:20)	152	141.0	(9, 2, 0)

(3, 1)	2016-10-18	(09:20, 09:40)	136	137.0	(9, 2, 0)
(3, 1)	2016-10-19	(09:20, 09:40)	91	102.0	(9, 1, 4)
(3, 1)	2016-10-20	(09:20, 09:40)	110	91.0	(9, 1, 4)
(3, 1)	2016-10-21	(09:20, 09:40)	106	118.0	(9, 2, 0)
(3, 1)	2016-10-22	(09:20, 09:40)	84	79.0	(7, 1, 2)
(3, 1)	2016-10-23	(09:20, 09:40)	74	89.0	(5, 2, 0)
(3, 1)	2016-10-24	(09:20, 09:40)	113	153.0	(9, 2, 0)
(3, 1)	2016-10-18	(09:40, 10:00)	99	148.0	(9, 2, 0)
(3, 1)	2016-10-19	(09:40, 10:00)	98	96.0	(9, 1, 4)
(3, 1)	2016-10-20	(09:40, 10:00)	89	85.0	(9, 1, 4)
(3, 1)	2016-10-21	(09:40, 10:00)	101	128.0	(9, 2, 0)
(3, 1)	2016-10-22	(09:40, 10:00)	79	79.0	(7, 1, 2)
(3, 1)	2016-10-23	(09:40, 10:00)	87	95.0	(5, 2, 0)

(3, 1)	2016-10-24	(09:40, 10:00)	109	167.0	(9, 2, 0)
(3, 1)	2016-10-18	(17:00, 17:20)	78	82.0	(7, 0, 4)
(3, 1)	2016-10-19	(17:00, 17:20)	69	76.0	(3, 2, 1)
(3, 1)	2016-10-20	(17:00, 17:20)	81	82.0	(9, 1, 2)
(3, 1)	2016-10-21	(17:00, 17:20)	92	96.0	(9, 0, 2)
(3, 1)	2016-10-22	(17:00, 17:20)	62	79.0	(8, 0, 1)
(3, 1)	2016-10-23	(17:00, 17:20)	80	89.0	(8, 0, 1)
(3, 1)	2016-10-24	(17:00, 17:20)	80	85.0	(5, 2, 0)
(3, 1)	2016-10-18	(17:20, 17:40)	75	74.0	(7, 0, 4)
(3, 1)	2016-10-19	(17:20, 17:40)	75	75.0	(3, 2, 1)
(3, 1)	2016-10-20	(17:20, 17:40)	84	78.0	(9, 1, 2)
(3, 1)	2016-10-21	(17:20, 17:40)	83	88.0	(9, 0, 2)
(3, 1)	2016-10-22	(17:20, 17:40)	68	81.0	(8, 0, 1)

(3, 1)	2016-10-23	(17:20, 17:40)	81	87.0	(8, 0, 1)
(3, 1)	2016-10-24	(17:20, 17:40)	97	87.0	(5, 2, 0)
(3, 1)	2016-10-18	(17:40, 18:00)	86	76.0	(7, 0, 4)
(3, 1)	2016-10-19	(17:40, 18:00)	87	78.0	(3, 2, 1)
(3, 1)	2016-10-20	(17:40, 18:00)	80	81.0	(9, 1, 2)
(3, 1)	2016-10-21	(17:40, 18:00)	70	84.0	(9, 0, 2)
(3, 1)	2016-10-22	(17:40, 18:00)	65	75.0	(8, 0, 1)
(3, 1)	2016-10-23	(17:40, 18:00)	76	82.0	(8, 0, 1)
(3, 1)	2016-10-24	(17:40, 18:00)	80	87.0	(5, 2, 0)
(3, 1)	2016-10-18	(18:00, 18:20)	73	74.0	(7, 0, 4)
(3, 1)	2016-10-19	(18:00, 18:20)	71	78.0	(3, 2, 1)
(3, 1)	2016-10-20	(18:00, 18:20)	54	73.0	(9, 1, 2)
(3, 1)	2016-10-21	(18:00, 18:20)	62	80.0	(9, 0, 2)

(3, 1)	2016-10-22	(18:00, 18:20)	56	74.0	(8, 0, 1)
(3, 1)	2016-10-23	(18:00, 18:20)	78	80.0	(8, 0, 1)
(3, 1)	2016-10-24	(18:00, 18:20)	91	87.0	(5, 2, 0)
(3, 1)	2016-10-18	(18:20, 18:40)	65	70.0	(7, 0, 4)
(3, 1)	2016-10-19	(18:20, 18:40)	77	78.0	(3, 2, 1)
(3, 1)	2016-10-20	(18:20, 18:40)	78	77.0	(9, 1, 2)
(3, 1)	2016-10-21	(18:20, 18:40)	73	80.0	(9, 0, 2)
(3, 1)	2016-10-22	(18:20, 18:40)	55	73.0	(8, 0, 1)
(3, 1)	2016-10-23	(18:20, 18:40)	88	77.0	(8, 0, 1)
(3, 1)	2016-10-24	(18:20, 18:40)	94	85.0	(5, 2, 0)
(3, 1)	2016-10-18	(18:40, 19:00)	68	70.0	(7, 0, 4)
(3, 1)	2016-10-19	(18:40, 19:00)	80	78.0	(3, 2, 1)
(3, 1)	2016-10-20	(18:40, 19:00)	66	70.0	(9, 1, 2)

(3, 1)	2016-10-21	(18:40, 19:00)	58	73.0	(9, 0, 2)
(3, 1)	2016-10-22	(18:40, 19:00)	77	71.0	(8, 0, 1)
(3, 1)	2016-10-23	(18:40, 19:00)	79	73.0	(8, 0, 1)
(3, 1)	2016-10-24	(18:40, 19:00)	74	87.0	(5, 2, 0)

Tabla 8.22: Tabla de predicciones de la segunda aproximación de predicciones del volumen de tráfico (La fila con un '-' corresponde a una ruta, día e intervalo de tiempo de la que no se dispone dato real para comparar)

Bibliografía

- [1] Y. Yin and P. Shang, "Forecasting traffic time series with multivariate predicting method", Applied Mathematics and Computation, vol. 291 , pp. 266-278, 2016. [Online]. Disponible en: <https://goo.gl/5gkEfP>
- [2] P. Yuan and X. Lin , "How long will the traffic flow time series keep efficacious to forecast the future?", Physica A: Statistical Mechanics and its Applications, vol. 467 , pp. 419-437, 2017. [Online]. Disponible en: https://drive.google.com/open?id=1r4ZF2wughH4nZcRUsQqqn_hmrSSR6mw8
- [3] H. A. Sevilla, "Predicción de tráfico en las carreteras de la red de la Generalitat Valenciana", Trabajo fin de carrera, Dep. De Sistemas Informáticos y Computación, Escola Tècnica Superior d'Enginyeria Informàtica, Universitat Politècnica de València, Valencia, 2015. [Online]. Disponible en: <https://drive.google.com/open?id=1usrXMdc7c-J3-1Rt2CjzAHLvZII3eeEO>
- [4] M. Goletz, I. Feige and D. Heinrichs, "What Drives Mobility Trends: Results from Case Studies in Paris, Santiago de Chile, Singapore and Vienna", Transportation Research Procedia, vol. 13 , pp. 49-60, 2016. [Online]. Disponible en: <https://drive.google.com/open?id=1590qIrBgZvJjANFsTvHYZlLkWRz9vfFN>
- [5] C. Gloves, R. North, R. Johnston and G. Fletcher, "Short Term Traffic Prediction on the UK Motorway Network Using Neural Networks", Transportation Research Procedia, vol. 13 , pp. 184-195, 2016. [Online]. Disponible en: https://drive.google.com/open?id=1rDz5GfONXSf7ZFhLcgQv_FFdSnrmj55F
- [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=1GD1ZIpWDq7qMM7bvpTSTnIWqSnpDlSt>
- [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=1uK8IwRTe061NVbWQn4FQFxH7BUx6b6mq>
- [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?id=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-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>

<http://halweb.uc3m.es/esp/Personal/personas/amalonso/esp/seriestemporales.pdf>

http://www.est.uc3m.es/esp/nueva_docencia/leganes/ing_industrial/estadistica_industrial/doc_grupo2/archivos/tablasicosARyMA.pdf

