



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, 11 de abril de 2018

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

C E R T I F I C A

Que la presente memoria titulada:

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

ha sido realizada bajo su dirección por D. **Javier Ramos Fernández**,
con N.I.F. 45.865.421-V.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 11 de abril de 2018

Agradecimientos

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

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

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

Al resto de mi familia que, a pesar del alejamiento, me hagan pasar unos veranos inolvidables y su apoyo incondicional.

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

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

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

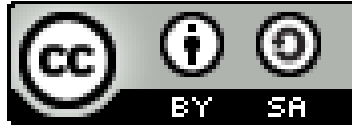
A la gente que ha dedicado su tiempo a escuchar mis problemas y me ha animado a seguir adelante.

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

Javier

Licencia

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



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

Resumen

*El objetivo de este trabajo ha sido realizar predicciones del tiempo promedio de viaje y el volumen de tráfico en la red de carreteras propuesta por la competición KDDCup 2017 (concretamente en **Hangzhou**, provincia de **Zhejiang**, en China). Para llevar a cabo esta tarea, nos hemos basado en la utilización de técnicas de aprendizaje automático para construir modelos de predicción que se adecuen de la forma más precisa posible a 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

Here should be the abstract in a foreing language...

Keywords: Keyword1, Keyword2, Keyword3, ...

Índice general

Capítulo 1 Introducción.....	1
1.1 Antecedentes. Problemática y estado del arte.....	2
1.2 Objetivos.....	6
1.3 Organización de la memoria.....	7
Capítulo 2 Tecnologías.....	8
2.1 Almacenamiento de datos.....	8
2.1.1 En el proyecto.....	8
2.1.1.1 PostgreSQL.....	8
2.1.2 Otras tecnologías posibles.....	9
2.1.2.1 MySQL.....	10
2.1.2.2 Oracle.....	11
2.1.2.3 Microsoft SQL Server.....	12
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.....	16

Capítulo 3 La minería de datos.....	17
3.1 Introducción.....	17
3.2 Técnicas de minería de datos.....	20
3.2.1 XGBOOST.....	20
3.2.1.1 Definición.....	20
3.2.1.2 Gradient boosting.....	21
3.2.1.3 Implementación del algoritmo.....	24
3.2.2 LightGBM.....	25
3.2.2.1 Definición.....	25
3.2.2.2 Ventajas.....	26
3.2.3 Perceptrón multicapa (Redes neuronales).....	27
3.2.3.1 Definición.....	27
3.2.3.2 Neuronas.....	28
3.2.3.3 Redes de neuronas.....	29
3.2.3.4 Entrenamiento de una red neuronal.....	31
3.2.4 Modelo ARIMA.....	32
3.2.4.1 Definición de una serie temporal.....	32
3.2.4.2 Características de las series temporales.....	33
3.3 Primera sección de este capítulo.....	35
3.4 Segundo apartado de este capítulo.....	35
3.5 Tercer apartado de este capítulo.....	35
Capítulo 4 Título del capítulo cuarto.....	36
Capítulo 5 Conclusiones y líneas futuras.....	37
Capítulo 6 Summary and Conclusions.....	38
Capítulo 7 Presupuesto.....	39
7.1 Sección Uno.....	39

Capítulo 8 Título del Apéndice 1.....	40
8.1 Algoritmo XXX.....	40
8.2 Algoritmo YYY.....	40
Capítulo 9 Título del Apéndice 2.....	42
9.1 Otro apéndice: Sección 1.....	42
9.2 Otro apéndice: Sección 2.....	42

Índice de figuras

Figura 2.1: Consola interactiva de PostgreSQL.....	9
Figura 2.2: Entorno gráfico de MySQL Workbench.....	11
Figura 2.3: Entorno gráfico del entorno de desarrollo Oracle JDeveloper.....	12
Figura 2.4: Entorno gráfico de Microsoft SQL Server Management Studio.....	13
Figura 2.5: Entorno gráfico de RapidMiner.....	15
Figura 2.6: Entorno gráfico de Weka.....	16
Figura 3.1: El proceso de descubrimiento de conocimiento en bases de datos (KDD).....	17
Figura 3.2: Cuatro de las tareas principales de minería de datos.....	19
Figura 3.3: Crecimiento level-wise del árbol en XGBoost.....	26
Figura 3.4: Crecimiento leaf-wise del árbol en LightGBM.....	26

Índice de tablas

Tabla 7.1: Resumen de tipos.....	8
----------------------------------	---

Capítulo 1

Introducción

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

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

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

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

En muchas situaciones, el método tradicional de convertir los datos en conocimiento consiste en un análisis e interpretación realizada de forma manual. El especialista en la materia analiza los datos y elabora un informe o hipótesis que refleja las tendencias o pautas de los mismos. Esta forma de actuar es pausado, costosa y muy subjetiva. En realidad, el análisis manual es impracticable en dominios donde el volumen de los datos crece exponencialmente. Consecuentemente, muchas decisiones importantes se realizan no sobre la base de los datos disponibles, sino siguiendo la propia intuición del usuario al no disponer de las herramientas necesarias. No obstante, actualmente existen herramientas de apoyo, como son las herramientas **OLAP** (*On-line Analytical Processing*, Procesamiento Analítico en Línea), que son técnicas de análisis descriptivo y de sumariaización que permite transformar los datos en otros datos agregados o cruzados de manera sofisticada.

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

1.1 Antecedentes. Problemática y estado del arte

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

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

Algunos ejemplos de estudio en este ámbito son los siguientes:

1. En el artículo [1] se propone utilizar series temporales multivariadas (método de predicción multivariante) para pronosticar variables de tráfico. Este método se plantea frente a las series temporales escalares que, aunque en teoría son genéricamente suficientes para reconstruir la dinámica de los sistemas subyacentes, resulta más enriquecedor utilizar todas las variables disponibles.
2. En el artículo [2] se estudia la cuestión acerca del tiempo de validez en que se mantiene eficaz una serie histórica de tiempos de flujo del tráfico para predecir el futuro. Es decir, se plantea el tiempo t que puede perdurar la eficacia de los datos de flujo de tráfico existentes en un tiempo t_0 para estimar tendencias de variación del flujo de tráfico en un tiempo t_0+t . Para ello, recopilan los datos de las series temporales de flujo de tráfico con diferentes granularidades y realizan una serie de análisis y métodos como analizar la propiedad de memoria larga de las series temporales del flujo de tráfico mediante el cálculo del exponente Hurst, además de un conjunto de comparaciones.
3. El artículo [3] es un Trabajo de Fin de Grado en el que se trata el tema de la predicción del tráfico en las carreteras de la red de la Generalitat Valenciana. El autor pretendía con este trabajo la posibilidad de mejorar la metodología empleada en algunas fases de la explotación de los datos de aforos a través del uso de métodos científicos, complementándola con herramientas estadísticas. Esto se hace puesto que los planes de aforo no pueden obtener el muestreo completo de toda la red durante todo el año dado el alto número de puntos de toma de datos y los recursos necesarios para abarcarlos todos de forma permanente. Para realizar la mejora, el autor considera la Intensidad

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

4. En el artículo [4] se estudian las tendencias de movilidad urbana en París, Santiago de Chile, Singapur y Viena con el objetivo de analizar la demanda de las diversas formas de transporte que existen en esas ciudades y establecer políticas adecuadas. No solo se examinan estas tendencias, sino también sus causas. Para ello, primero se identifican las tendencias específicas de cada una de las ciudades principalmente a través de indicadores de transporte, como los datos de viaje con respecto a los usuarios y estructuras espaciales, además de analizar el contexto de cada ciudad (infraestructura, desarrollo económico y desarrollo social de la ciudad) y realizar un modelo para explicar el comportamiento de los usuarios frente a unas tendencias u otras a partir de la diferenciación entre los motivos socio-emocionales y racionales de los mismos. A continuación, se consultan a expertos en el sistema de transporte de cada una de las ciudades para validar esas tendencias identificadas y preparar análisis cualitativos. Por último, se realizan análisis para comprender y describir las tendencias desde la perspectiva de los viajeros. El artículo examina una amplia gama de modos de transportes espacialmente y socialmente diferenciados.
5. El tema del artículo [5] consiste en realizar estimaciones a corto plazo (15 minutos en el futuro) con la información histórica del tráfico de la red de autopistas del Reino Unido utilizando redes neuronales, de tal forma que esto permita reducir la congestión del transporte mediante la mejora de sistemas inteligentes de transporte utilizados para controlar el tráfico para que realicen decisiones proactivas sobre la red de carreteras (anticiparse al inicio de la congestión del tráfico). Se plantean efectuar estas decisiones anticipadas a través de advertencias de la congestión

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

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

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

KNN (k-Nearest Neighbours).

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

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

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

1.2 Objetivos

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

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

tiempo solicitados por la competición.

1.3 Organización de la memoria

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

- En el capítulo 2 se introducen las tecnologías utilizadas para llevar a cabo la carga de las bases de datos que almacenan los datos de la competición y la aplicación de diferentes modelos de predicción sobre los mismos.
- En el capítulo 3 se explican las nociones fundamentales de la minería de datos, así como la descripción de distintas técnicas de aprendizaje automático empleadas para descubrir patrones y tendencias que existen en nuestros datos.
- En el capítulo 4 se abordan de forma detallada las distintas fases del proyecto, desde la creación de las bases de datos hasta la comparación de resultados de los distintos algoritmos de aprendizaje automático.

Capítulo 2

Tecnologías

La elección de las herramientas y tecnologías empleadas para el desarrollo del proyecto se ha realizado cautelosamente, de tal forma que nos ha permitido concentrarnos en el problema en cuestión en lugar de tener que instruirnos exhaustivamente en ellas y que aumentara la complejidad del trabajo. A continuación, se enumeran las principales tareas del proyecto junto con el software al que se ha recurrido para completarlas.

2.1 Almacenamiento de datos

2.1.1 En el proyecto

2.1.1.1 PostgreSQL

PostgreSQL es un potente sistema de base de datos objeto-relacional de código abierto basado en el paquete POSTGRES, desarrollado en el Departamento de Informática de la Universidad de California en Berkeley y liberado bajo la licencia BSD. Es un sistema que permite la creación, gestión y administración de bases de datos, así como la elección y manejo de las estructuras necesarias para el almacenamiento y búsqueda de la información del modo más eficiente posible. Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una sola compañía, sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo; dicha comunidad es denominada el *PGDG (PostgreSQL Global Development Group)*.

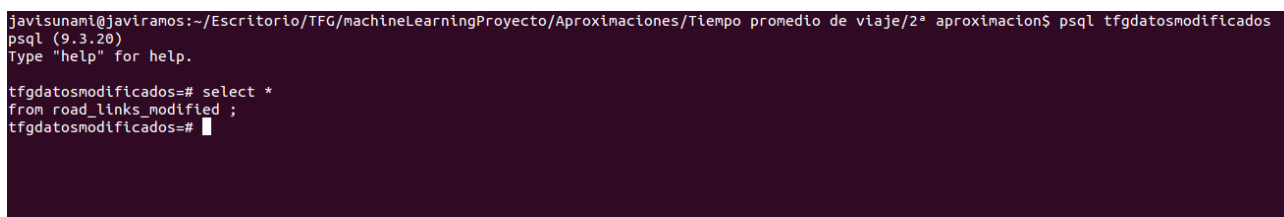
Este sistema de base de datos soporta una gran parte del estándar SQL y ofrece muchas características modernas incluyendo las siguientes:

- Consultas SQL complejas

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

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

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



```
javisunamijaviramos:~/Escritorio/TFG/machineLearningProyecto/Aproximaciones/Tiempo promedio de viaje/2ª 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.

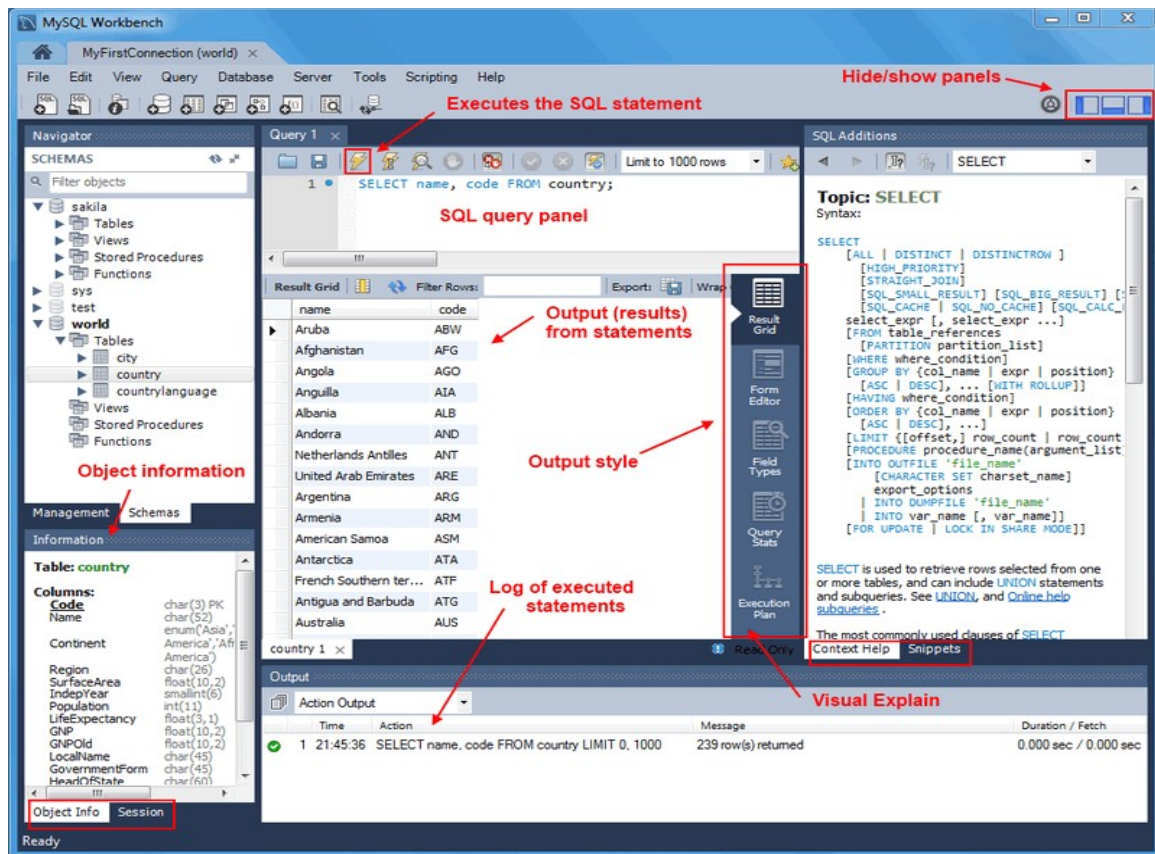


Figura 2.2: Entorno gráfico de MySQL Workbench

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.

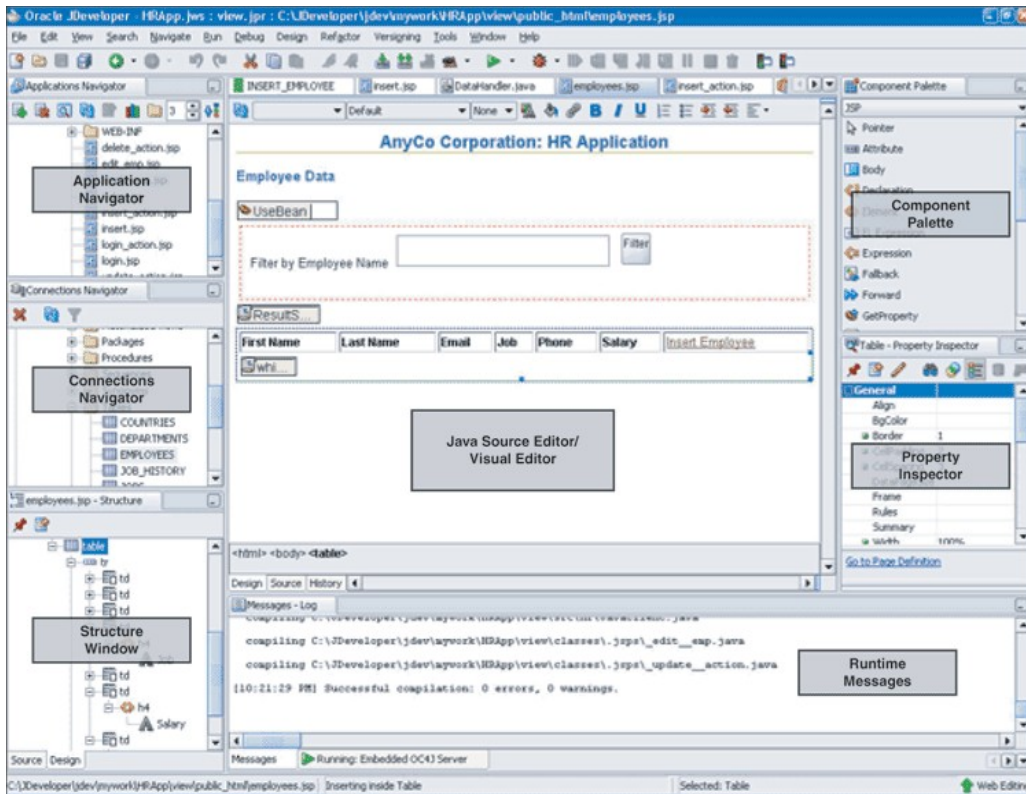


Figura 2.3: Entorno gráfico del entorno de desarrollo Oracle JDeveloper

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*.

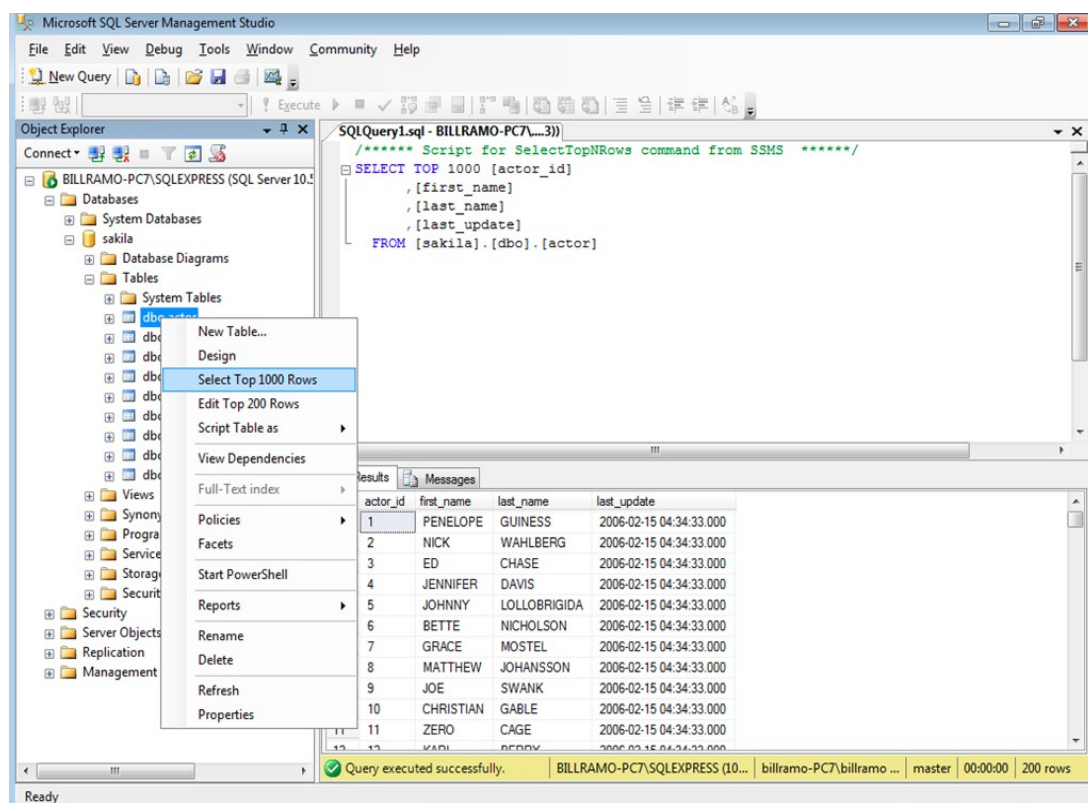


Figura 2.4: Entorno gráfico de Microsoft SQL Server Management Studio

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 archivos 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*.

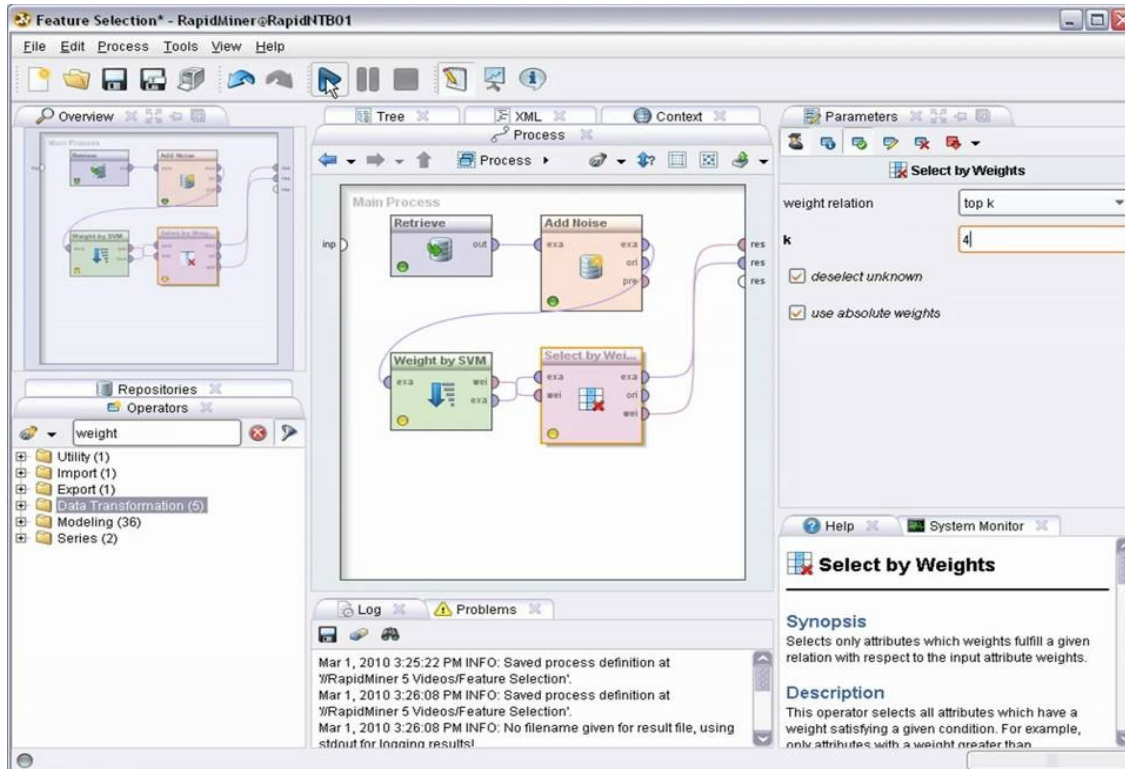


Figura 2.5: Entorno gráfico de RapidMiner

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

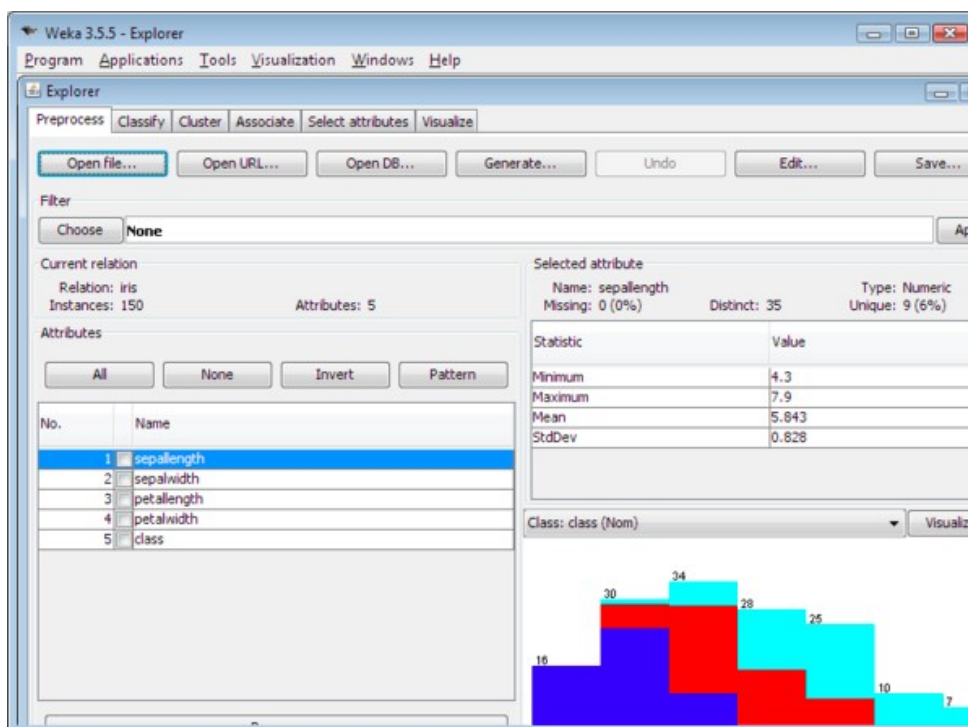


Figura 2.6: Entorno gráfico de Weka

Capítulo 3

La minería de datos

3.1 Introducción

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

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

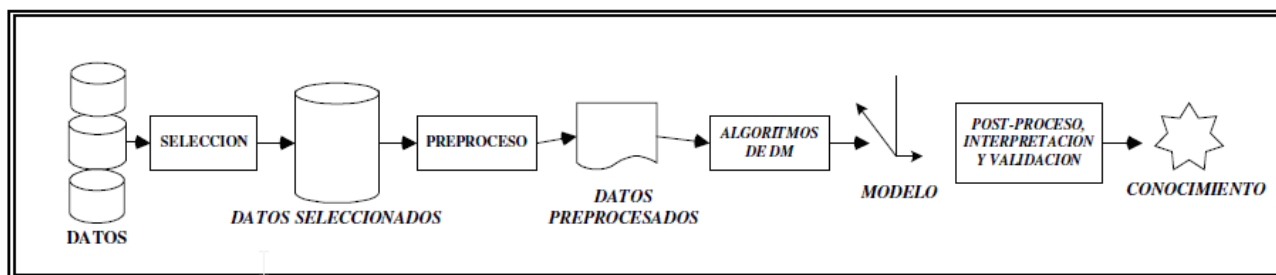


Figura 3.1: El proceso de descubrimiento de conocimiento en bases de datos (KDD)

- Los **datos de entrada** pueden almacenarse en una variedad de formatos (archivos planos, hojas de cálculo o tablas relacionales) y pueden residir en un repositorio de datos centralizado o distribuirse en varios sitios.
- El propósito del **preprocesamiento** es *transformar los datos* de entrada brutos en un formato apropiado para el análisis posterior. Los pasos involucrados en el preprocesamiento de datos incluyen la fusión de datos de múltiples fuentes, la limpieza de datos para eliminar el ruido y la duplicación de observaciones, y la selección de registros y características que son relevantes para la tarea de minería de datos a mano. Debido a las muchas maneras en que los datos pueden ser recolectados y almacenados, el *preprocesamiento* de datos es quizás el paso más laborioso y que consume más tiempo en el proceso general de descubrimiento de conocimiento.
- El objetivo del siguiente proceso es **integrar los resultados de la minería de datos** en los sistemas de apoyo a la toma de decisiones. Es la **fase de modelamiento** propiamente tal, en donde métodos inteligentes son aplicados con el objetivo de extraer patrones previamente desconocidos, válidos, nuevos, potencialmente útiles y comprensibles y que están contenidos u “ocultos” en los datos.
- La fase anterior (de integración de resultados) requiere una etapa de **tratamiento posterior** que garantice que en el sistema de apoyo a la adopción de decisiones sólo se incorporen resultados válidos y útiles. Es decir, se **identifican los patrones obtenidos** y que son realmente interesantes, basándose en algunas medidas y se realiza una **evaluación de los resultados obtenidos**.

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

- **Tareas predictivas.** El objetivo de estas tareas es predecir el valor de un atributo particular basado en los valores de otros atributos. El atributo a predecir se conoce comúnmente como la *variable objetivo* o *dependiente*, mientras que los atributos utilizados para hacer la predicción se conocen como las *variables explicativas* o *independientes*.
- **Tareas descriptivas.** El objetivo es derivar patrones (correlaciones, tendencias, clusters, trayectorias y anomalías) que resumen las relaciones subyacentes en los datos. Las tareas descriptivas de minería de datos son a menudo de naturaleza exploratoria y con frecuencia

requieren técnicas de postprocesamiento para validar y explicar los resultados

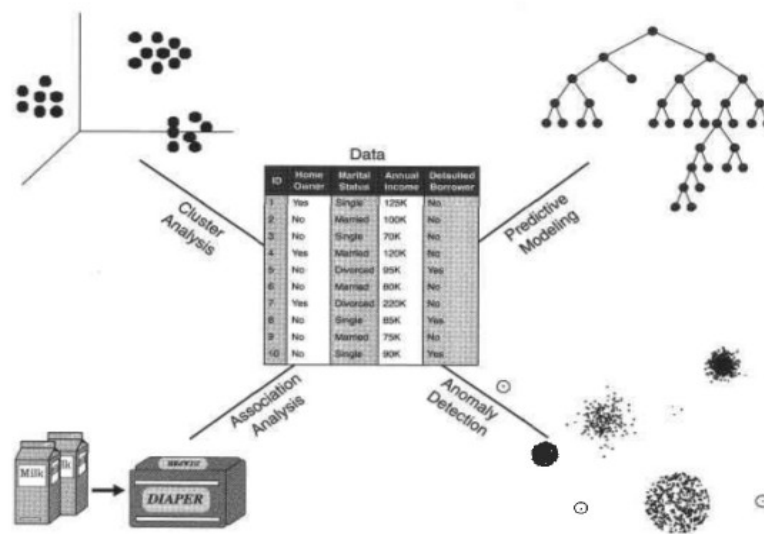


Figura 3.2: Cuatro de las tareas principales de minería de datos

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

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

identificación de páginas Web a las que se accede de forma conjunta o la comprensión de las relaciones entre los diferentes elementos del sistema climático de la Tierra.

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

3.2 Técnicas de minería de datos

Para realizar las predicciones del tiempo promedio de viaje y el volumen de tráfico oportunas propuestas por la competición *KDDCup 2017* ha sido imprescindible la utilización de técnicas de minería de datos apropiadas para dichas tareas. En los siguientes apartados se lleva a cabo una explicación detallada de las mismas.

3.2.1 XGBOOST

3.2.1.1 Definición

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

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

3.2.1.2 Gradient boosting

La **potenciación del gradiente** (o *gradient boosting*) es una de las técnicas más poderosas para construir modelos predictivos. Es una técnica en la que se crean nuevos modelos que predicen los residuos o errores de modelos anteriores y luego se suman para llevar a cabo la predicción final. Se denomina *potenciación del gradiente* porque utiliza un algoritmo de descenso de gradiente para minimizar la pérdida al añadir nuevos modelos. Es una técnica de aprendizaje automático utilizada para el análisis de regresión y los problemas de clasificación estadística, que produce un modelo predictivo en forma de un conjunto de modelos predictivos débiles, normalmente *árboles de decisión*. Construye el modelo de forma escalonada como hacen otros métodos de refuerzo, y los generaliza permitiendo la optimización arbitraria de una función de pérdida diferenciable.

Este algoritmo es un algoritmo eficiente para convertir hipótesis relativamente pobres en hipótesis muy buenas. Una *hipótesis débil* se define como aquella cuyo desempeño es al menos ligeramente mejor que el azar. La **potenciación de hipótesis** (*hypothesis boosting*) es la idea de filtrar las observaciones, dejando aquellas observaciones que el **weak learner** puede manejar y enfocándose en desarrollar nuevos aprendizajes débiles para manejar las observaciones difíciles restantes. La idea es utilizar el método de aprendizaje débil varias veces para obtener una sucesión de hipótesis, cada una reorientada hacia los ejemplos que los anteriores encontraban difíciles y mal clasificados.

La potenciación del gradiente implica tres elementos:

- Una **función coste** a optimizar. Esta función asigna un *evento* o valores de una o más variables en un número real que representa intuitivamente algún "coste" asociado al evento. Ésta debe ser *diferenciable* (una función diferenciable de una variable real es una función cuya derivada existe en cada punto de su dominio). Un problema de optimización busca minimizar una función de pérdida.
- Un **weak learner** para hacer predicciones. Los árboles de decisión se

utilizan como el **weak learner** en la potenciación del gradiente. Específicamente se utilizan árboles de regresión que emiten valores reales para las particiones y cuya salida puede sumarse, permitiendo que las salidas de los modelos subsiguientes se sumen y "corrijan" los residuos en las predicciones. Los árboles se construyen de una manera codiciosa, eligiendo los mejores puntos de división en función de las puntuaciones de pureza como *Gini* o para minimizar el error. No obstante, es común restringir los **weak learners** de manera específica (un número máximo de capas, nodos, divisiones o nodos hoja) para asegurar que permanezcan débiles pero que aún puedan ser construidos de una manera codiciosa, como veremos más adelante.

- 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 conlleva un mayor poder de predicción.

3.2.2 LightGBM

3.2.2.1 Definición

LightGBM es un framework de potenciación del gradiente que utiliza un *algoritmo de aprendizaje basado en árboles*. Es un framework rápido, distribuido y de alto rendimiento, utilizado para clasificar, priorizar y muchas otras tareas de aprendizaje automático.

LightGBM es similar a *XGBoost* pero varía en algunas formas específicas, especialmente en la forma en que crea los árboles. **LightGBM** realiza la construcción de los árboles **verticalmente**, mientras que el otro algoritmo lo hace **horizontalmente**, lo que significa que el primero genera los árboles **leaf-wise** (búsqueda primero el mejor) mientras que el otro los genera **level-wise** (búsqueda en anchura). Es decir, dado que **LightGBM** se basa en *algoritmos de árbol de decisión*, divide la hoja del árbol **leaf-wise** con el mejor ajuste, mientras que otros algoritmos de refuerzo dividen la profundidad del árbol **depth-wise** o **level-wise** en lugar de **leaf-wise**.

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

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

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

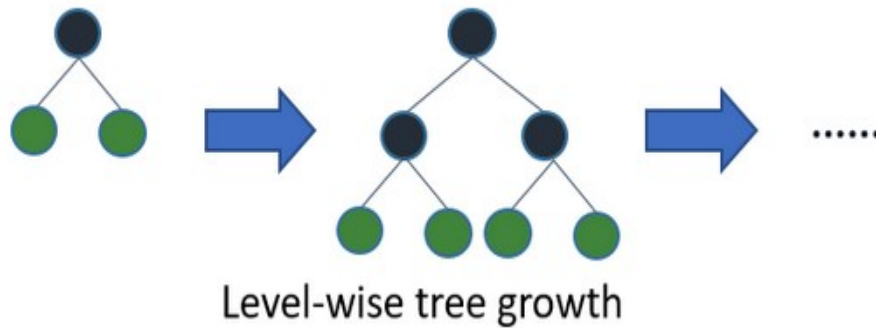


Figura 3.3: Crecimiento level-wise del árbol en XGBoost

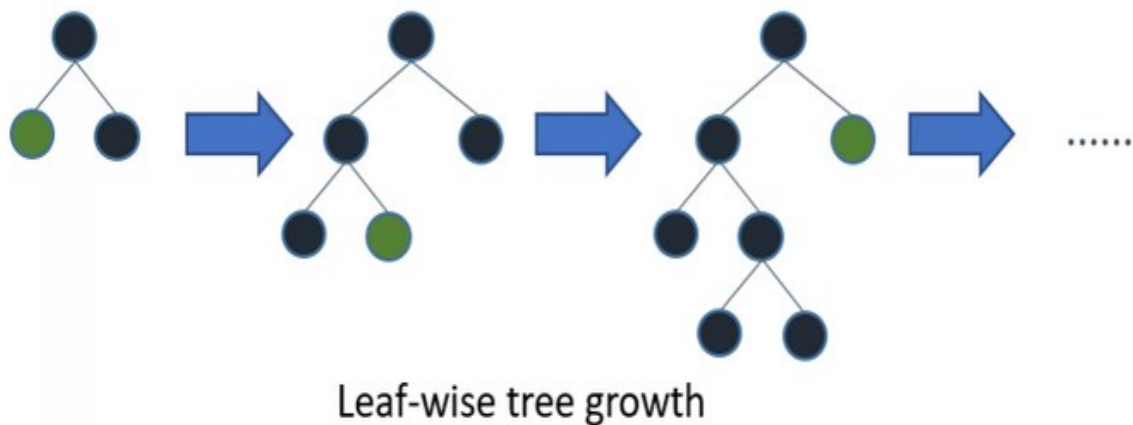


Figura 3.4: Crecimiento leaf-wise del árbol en LightGBM

3.2.2.2 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.

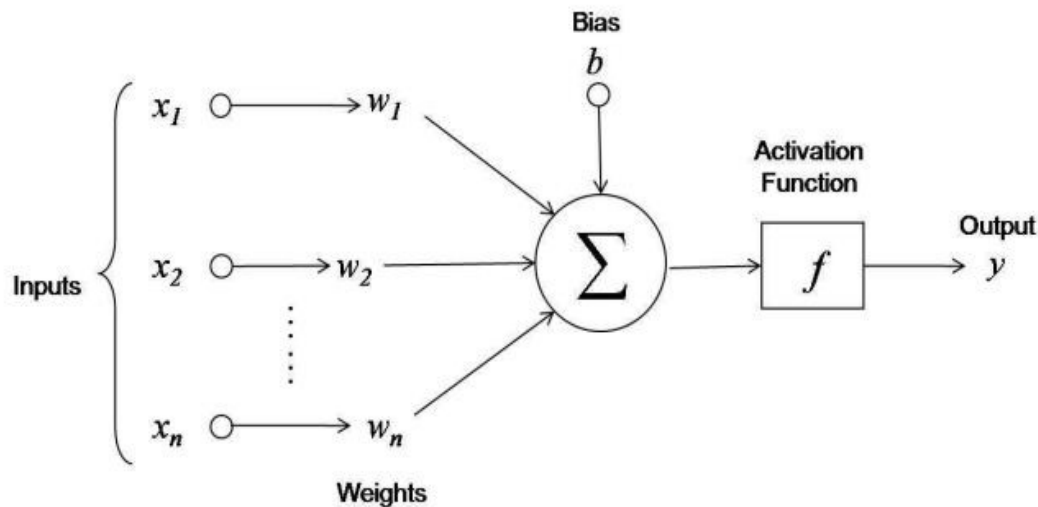


Figura 3.5: Estructura de una neurona artificial

De forma general, el funcionamiento de una neurona artificial es el siguiente:

1. Recibe una serie de **entradas**, que pueden ser características de un conjunto de entrenamiento o salidas de otras neuronas.
2. A continuación, se aplican unos **pesos** a las entradas. Estos pesos se inicializan a menudo a valores aleatorios pequeños, como valores en el rango de 0 a 0.3 , aunque se pueden utilizar esquemas de inicialización más complejos. Es deseable mantener pesos pequeños en la red y se pueden utilizar técnicas de regularización para ello.
3. Después, las entradas ponderadas se suman junto con un *sesgo* que tiene la neurona (se interpreta como una entrada que permite desplazar la función de activación a la izquierda o a la derecha, que siempre tiene el valor 1.0 y que también debe ser ponderada) y pasan

a través de una **función de activación**, obteniendo así las **salidas**. Esta *función de activación* es un simple mapeo de la entrada ponderada sumada a la salida de la neurona. Se llama función de activación porque gobierna el umbral a partir del cual se activa la neurona y la fuerza de la señal de salida. Es decir, Se utiliza para determinar la salida de la red neuronal como *si* o *no*: mapea los valores resultantes de 0 a 1 o de -1 a 1 , etc. (dependiendo de la función). Las distintas funciones de activación se engloban en dos tipos, *funciones de activación lineales* y *funciones de activación no lineales* y existen una gran variedad de ellas: *función sigmoide*, *Tanh*, *ReLU*, etc. Tradicionalmente se utilizan funciones de activación no lineales puesto que permiten a la red neuronal combinar las entradas de maneras más complejas y, a su vez, proporcionar una mejor capacidad en las funciones que pueden modelar.

3.2.3.3 Redes de neuronas

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

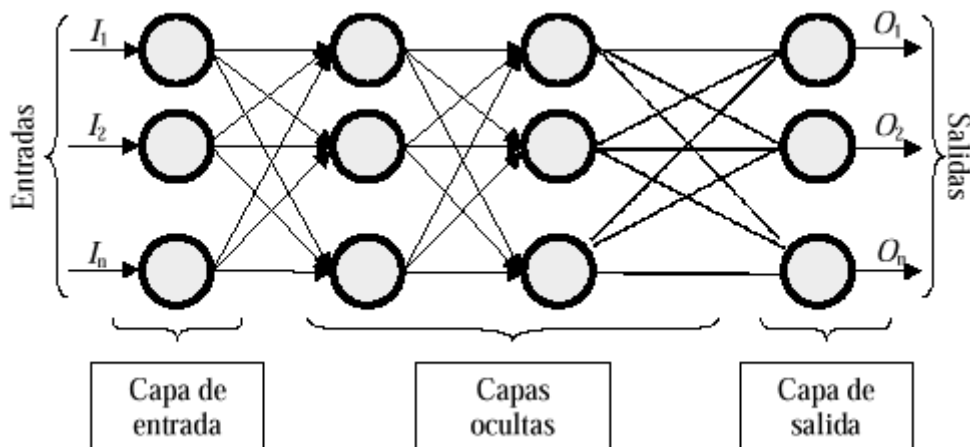


Figura 3.6: Estructura de una red neuronal

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

- **Capa de entrada:** La capa que toma la entrada del conjunto de datos se denomina *capa de entrada* o *capa visible*, ya que es la parte expuesta de la red. A menudo una red neuronal se representa con una capa visible con una neurona por valor de entrada o columna en el conjunto de datos. Éstas no son neuronas como se describió anteriormente, sino que simplemente pasan el valor de entrada a la siguiente capa.
- **Capas ocultas:** Las capas posteriores a la capa de entrada se denominan *capas ocultas* porque no están expuestas directamente a la entrada. La estructura de red neuronal más simple es tener una sola neurona en la capa oculta que produzca directamente el valor de salida. Dado el aumento de la potencia de cálculo y la eficiencia de las bibliotecas, se pueden construir redes neuronales muy profundas. El *aprendizaje profundo* se refiere a tener muchas capas ocultas en una red neuronal. Son profundas porque habrían sido inimaginablemente lentas para entrenar históricamente, pero pueden tardar segundos o minutos para entrenar dichas redes neuronales usando técnicas y hardware modernos.
- **Capa de salida:** La última capa oculta se denomina *capa de salida* y es responsable de emitir un valor o vector de valores que corresponden al formato requerido para el problema. La elección de la función de activación en la capa de salida está fuertemente limitada por el tipo de problema que se está modelando. Por ejemplo, un problema de *regresión* puede tener una neurona de salida única y la neurona puede no tener función de activación. Otro ejemplo sería un problema de *clasificación binaria*, que puede tener una neurona de salida única y utilizar una función de **activación sigmoide** para emitir un valor entre 0 y 1 para representar la probabilidad de predecir un valor para la clase 1. Esto se puede convertir en un valor de clase definido utilizando un umbral de 0.5 y ajustar valores inferiores al umbral a 0 y superiores a 1.

3.2.3.4 *Entrenamiento de una red neuronal*

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

Esta misma codificación en caliente se puede utilizar en la variable de salida en *problemas de clasificación* con más de una clase. Esto crearía un vector binario a partir de una sola columna que sería fácil de comparar directamente con la salida de la neurona en la capa de salida de la red neuronal que, como se describió anteriormente, produciría un valor para cada clase.

Las redes neuronales requieren que la entrada esté escalada de manera consistente. Se puede rescalar al rango entre 0 y 1 y esto se denomina **normalización**. Otra técnica bastante utilizada es **estandarizarla** para que la distribución de cada columna tenga la media de cero y la desviación estándar de 1, de modo que todas las entradas se expresan en rangos similares. Con la *normalización* y la *estandarización* el proceso de entrenamiento de la red neuronal se realiza con mucha mayor velocidad.

El clásico y aún preferido algoritmo de entrenamiento para redes neuronales se denomina **descenso por gradiente estocástico**, en el que una fila de datos se expone a la red neuronal a la vez como entrada. La red procesa la entrada hacia delante activando las neuronas a medida que se va avanzando a través de las capas ocultas hasta que finalmente se obtiene un valor de salida. Esto se denomina *propagación hacia delante* en la red neuronal. Es el tipo de propagación que también se utiliza después de que la red neuronal es entrenada para hacer predicciones sobre nuevos datos.

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

Los pesos en la red neuronal se pueden actualizar a partir de los errores

calculados para cada ejemplo de entrenamiento y esto se denomina ***aprendizaje en línea***. Puede resultar en cambios rápidos pero también caóticos en la red. De forma alternativa, los errores se pueden guardar en todos los ejemplos de entrenamiento y la red se puede actualizar al final. Esto se denomina ***aprendizaje por lotes*** y a menudo es más estable.

Típicamente, debido a que los conjuntos de datos son tan grandes y a las eficiencias computacionales, el *tamaño del lote* (el número de ejemplos que se le muestra a la red neuronal antes de una actualización), a menudo se reduce a un pequeño número, como decenas o cientos de ejemplos. El grado en el que se actualizan los pesos es controlado por un parámetro de configuración denominado ***velocidad de aprendizaje***. Este parámetro controla el cambio realizado en el peso de la red neuronal para un error determinado. A menudo se utilizan tamaños de peso pequeños tales como *0.1* o *0.01* o más pequeños.

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

3.2.4 Modelo ARIMA

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

3.2.4.1 Definición de una serie temporal

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

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.

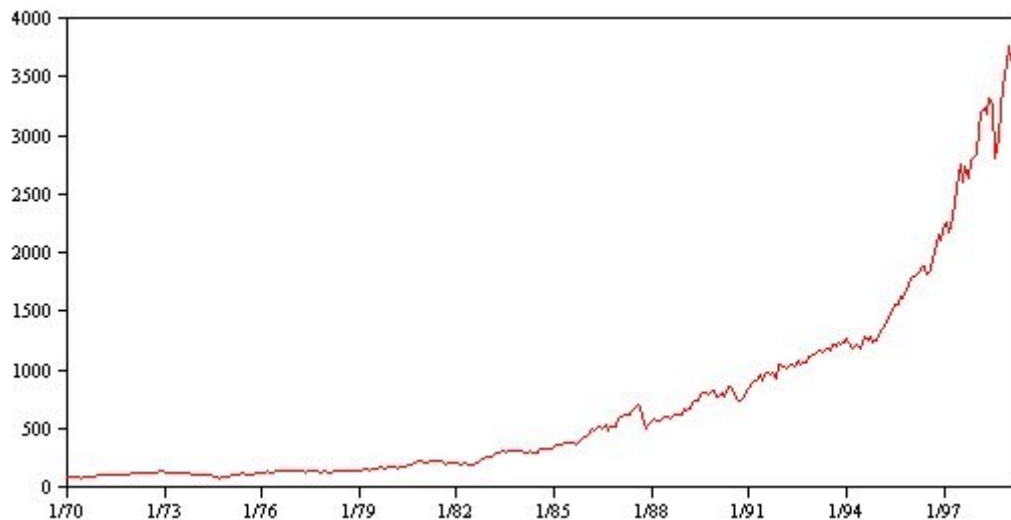


Figura 3.7: Ejemplo de una serie temporal con tendencia

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

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

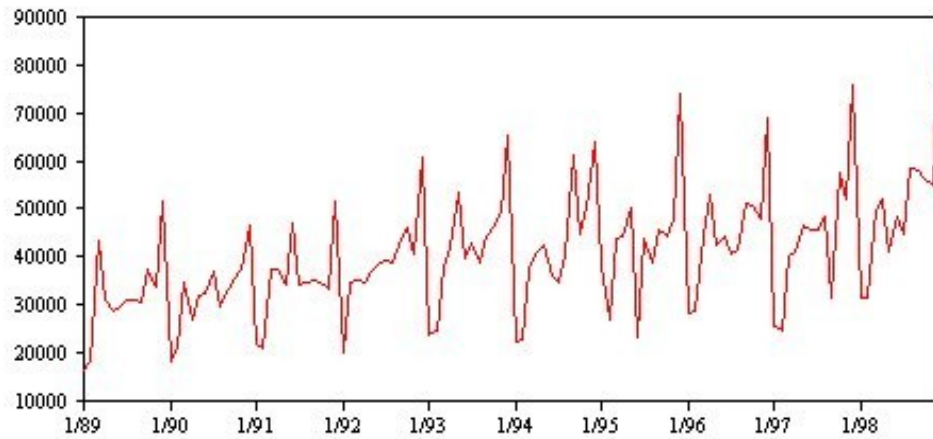


Figura 3.8: Ejemplo de una serie temporal con ciclos estacionales

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.

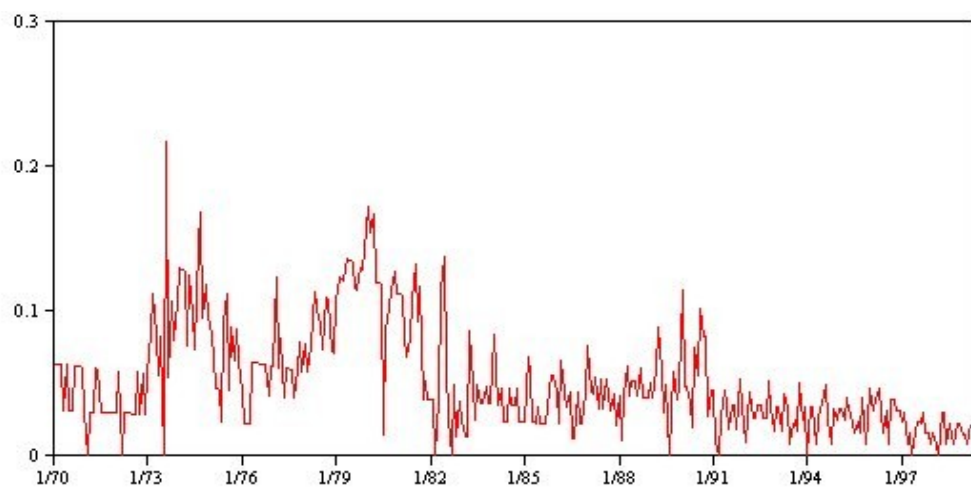


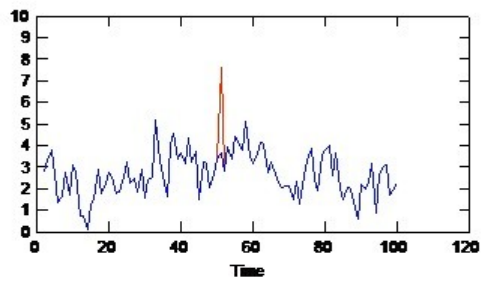
Figura 3.9: Serie temporal con pulsos

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.

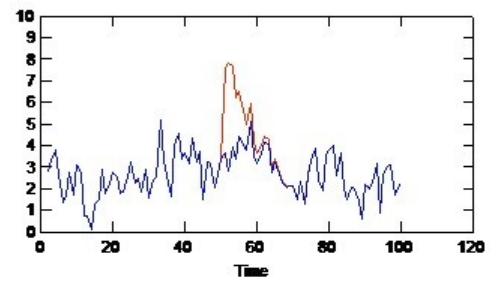
- **Valores atípicos:** Los desplazamientos en el nivel de una serie temporal que no se pueden explicar se denominan **valores atípicos**. Estas observaciones no coinciden con el resto de las series y pueden influir considerablemente en el análisis y, por lo tanto, afectar a la capacidad de previsión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

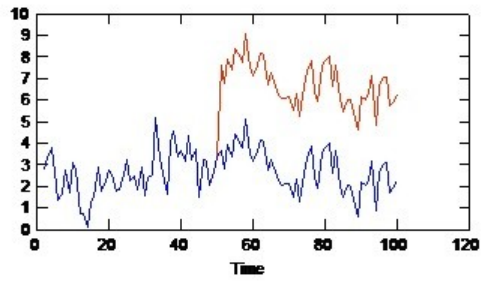
Additive Outlier



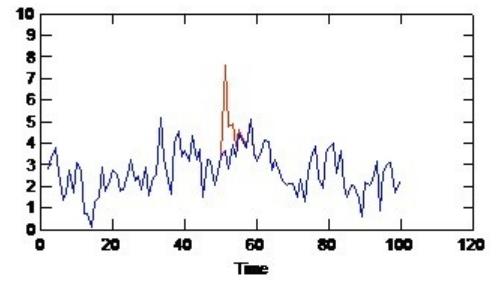
Innovational Outlier



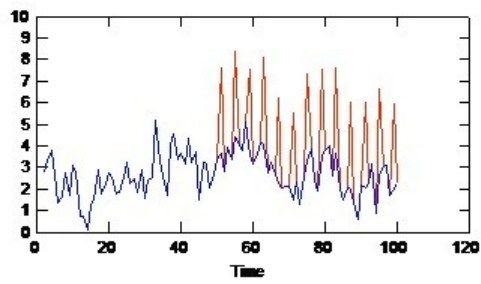
Level Shift Outlier



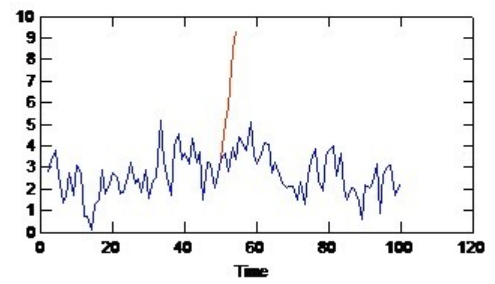
Transient Change Outlier



Seasonal Additive Outlier



Local Trend Outlier



— Outlier
— No Outlier

Capítulo 4

Fases del proyecto

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

- *Comprensión del problema planteado por la competición KDDCup 2017.* En primer lugar se ha llevado a cabo un estudio de las tareas encomendadas por la competición y de las reglas a aplicar en las predicciones que se realizaran.
- *Creación de una serie de bases de datos para almacenar los datos proporcionados por la competición.* Con el objetivo de poder acceder fácilmente y de forma flexible a la información suministrada por la competición KDDCup 2017, se ha planteado crear un conjunto de bases de datos que permita ordenar y clasificar los datos, de tal forma que se tenga una comprensión más acertada de ellos, se puedan realizar distintas combinaciones sobre los mismos y se genere nuevo conocimiento para llevar a cabo estimaciones.
- *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.

Capítulo 5

Conclusiones y líneas futuras

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

Capítulo 6

Summary and Conclusions

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

Capítulo 7

Presupuesto

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

7.1 Sección Uno

Tipos	Descripción
AAAA	BBBB
CCCC	DDDD
EEEE	FFFF
GGGG	HHHH

Tabla 7.1: Resumen de tipos

Capítulo 8

Título del Apéndice 1

8.1 Algoritmo XXX

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

8.2 Algoritmo YYY

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

* AUTORES

*

* FECHA

*

* DESCRIPCION

*

*

*****/

- *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.
1. *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.

Capítulo 9

Título del Apéndice 2

9.1 Otro apéndice: Sección 1

texto

9.2 Otro apéndice: Sección 2

texto

Bibliografía

- [1] Y. Yin and P. Shang. “Forecasting traffic time series with multivariate predicting method”. *Applied Mathematics and Computation* vol 291, pp. 266-278, 2016. [Online]. Disponible en: <https://goo.gl/5gkEfp>
- [2] P. Yuan and X. Lin. “How long will the traffic flow time series keep efficacious to forecast the future?”. *Physica A: Statistical Mechanics and its Applications* vol 467, pp. 419-431, 2017. [Online]. Disponible en: https://drive.google.com/open?id=1r4ZF2wughH4nZcRUsQqqn_hmrSSR6mw8
- [3] H. A. Sevilla, "Predicción de tráfico en las carreteras de la red de la Generalitat Valenciana", Trabajo fin de carrera, Dep. De Sistemas Informáticos y Computación, Escola Tècnica Superior d'Enginyeria Informàtica, Universitat Politècnica de València, Valencia, 2015. [Online]. Disponible en: <https://drive.google.com/open?id=1usrXMdc7c-J3-1Rt2CjzAHLvZII3eeEO>
- [4] M. Goletz, I. Feige and D. Heinrichs. “What Drives Mobility Trends: Results from Case Studies in Paris, Santiago de Chile, Singapore and Vienna,”. *Transportation Research Procedia* vol 13, pp. 49-60, 2016. [Online]. Disponible en: <https://drive.google.com/open?id=1590qIrBgZvJjANFsTvHYZILkWRz9vfFN>
- [5] C. Gloves, R. North, R. Johnston and G. Fletcher. “Short Term Traffic Prediction on the UK Motorway Network Using Neural Networks”. *Transportation Research Procedia* vol 13, pp. 184-195, 2016. [Online]. Disponible en: https://drive.google.com/open?id=1rDz5GfONXSf7ZFhLcgQv_FFfSnrmj55F

- [6] K. Hu, P. Huang, H. Chen and P. Yan, “KDD CUP 2017 Travel Time Prediction Predicting Travel Time – The Winning Solution of KDD Cup 2017”, presented at the KDD 2017, Halifax, Nova Scotia – Canada, 2017. [Online]. Disponible en: https://drive.google.com/open?id=12a4pBbxA6h_zy517ARtHTtn61w7uZ7jJ
- [7] Y. Huang, “Highway Tollgates Traffic Flow Prediction Task 1. Travel Time Prediction”, presented at the KDD 2017, Halifax, Nova Scotia – Canada, 2017. [Online]. Disponible en: https://drive.google.com/open?id=1GD1ZIpWDq7qMM7bvpTSTnIWqSnpDlS_t
- [8] H. Cai, R. Zhong, C.Wang et al., “KDD CUP 2017 Travel Time Prediction”, presented at the KDD 2017, Halifax, Nova Scotia – Canada, 2017. [Online]. Disponible en: <https://drive.google.com/open?id=1ew6oLOPHGoz8IMIt5g6XkZn67ADDtxJJ>
- [9] K. Hu, P. Huang, H. Chen and P. Yan, “KDDCUP 2017 Volume Prediction Step by step modeling for travel volume prediction”, presented at the KDD 2017, Halifax, Nova Scotia – Canada, 2017. [Online]. Disponible en: <https://drive.google.com/open?id=1uK8IwRTe061NVbWQn4FQFxFH7BUx6b6mq>
- [10] J. Zhou, Y. Guo, Y. Chen, J. Lin and H. Lin, “Learning and Prediction over Light-Weight Spatio-Temporal Data”, presented at the KDD 2017, Halifax, Nova Scotia – Canada, 2017. [Online]. Disponible en: https://drive.google.com/open?di=1Jbz0GNxYcjCghp0cfJia0omYKu1y5_aR
- [11] S. Luo, “KDD CUP 2017: Volume Prediction Task Solution”, presented at the KDD 2017, Halifax, Nova Scotia – Canada, 2017. [Online]. Disponible en: <https://drive.google.com/open?id=1-krIjoSatfoPNnzYRSOZKfxIKK0RHT59>

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

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

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

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

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

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

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

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

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

