Universidad Internacional de La Rioja (UNIR)

Escuela de Ingeniería

Máster en Análisis y Visualización de Datos Masivos

Modelos predictivos de accidentes de tráfico en Madrid

Trabajo Fin de Máster

presentado por: Cruz Bellas, Luis

Director/a: Madariaga Merino, Sara

Ciudad: Madrid
Fecha: 20-09-2017

Contenido

Índice de Imágenes

Índice de Tablas

Resumen

Nota: El objetivo de este trabajo es construir modelos predictivos que ayuden a estimar la probabilidad de ocurrencia de accidentes de tráfico en determinadas áreas de Madrid, España. Los modelos se construyen a partir de variables meteorológicas, festivos, franjas horarias y épocas del año entre otras. Para construir los modelos se aplican regresiones lineales y redes neuronales. Tras obtener los resultados, se puede observar que las variables predictoras elegidas no son capaces de explicar con exactitud el comportamiento de la variable dependiente. Por tanto, se deben explorar otras opciones tanto en las variables predictoras como en los métodos utilizados para la construcción de modelos.

Palabras Clave: Accidentes, tráfico, modelo predictivo, inteligencia artificial.

Abstract

Note: The objective of this publication is to build predictive models that help to estimate the probability of traffic accidents in some areas from Madrid, Spain. These models are built using meteorology variables, vacation days, time intervals, etc. Linear regression and neural networks are used to build the models. With the results, we can observe that predictor variables do not explain accurately the behaviour of the dependent variable. Thus, it is mandatory to explore other ways both in choosing the predictor variables and methods used to build models.

Keywords: Accidents, traffic, predictive models, artificial intelligence.

1. Introducción

Los accidentes de tráfico son una de las mayores causas externas de mortalidad en los países desarrollados. Este trabajo pretende construir modelos de predicción a partir de variables como el clima, la hora, la zona y la época del año que permitan estimar la probabilidad de ocurrencia de accidentes de tráfico. Para ello se han de capturar los datos, estudiarlos y procesarlos para que se puedan aplicar técnicas estadísticas y de inteligencia artificial. Después, se calcularán regresiones y redes neuronales sobre los datos y se analizarán los resultados para obtener conclusiones. Por último, se establecerá un plan de trabajo futuro para mejorar los resultados obtenidos.

1.1. Motivación

El ser humano ha conseguido vencer numerosas enfermedades y con ello ha logrado que la esperanza de vida sea cada vez mayor, especialmente en países desarrollados. Sin embargo, una de las mayores lacras en las sociedades desarrolladas son los accidentes de tráfico. En España, en 2016 murieron 1.160 personas en accidentes de tráfico (Dirección General de Tráfico, 2017), situándose como la quinta causa de muerte externa según el INE (Instituto Nacional de Estadística (INE), 2016). Por ello, los gobiernos de países como España invierten muchos recursos en prevenir estos accidentes mediante la mejora de las carreteras, la implantación de radares de velocidad o la creación del carnet por puntos. Estas medidas han conseguido reducir la mortalidad por accidentes de tráfico en los últimos años tal y como muestra el gráfico de la *Figura 1*. No obstante, el número de muertos en carreteras españolas sigue siendo elevado y se deben invertir recursos en disminuir la cifra lo máximo posible.

Consciente del problema que los accidentes de tráfico generan, el objetivo de este trabajo es aportar un grano de arena a la causa, realizando un estudio que pretende modelar la probabilidad de que se produzca un accidente de tráfico en puntos críticos de la ciudad donde reside el autor: Madrid, España.

Teniendo en cuenta la información disponible en el momento de realizar el trabajo, se pretende establecer patrones entre los accidentes de tráfico y variables como la lluvia, el viento, la temperatura o el día de la semana, mediante técnicas estadísticas y de inteligencia artificial.

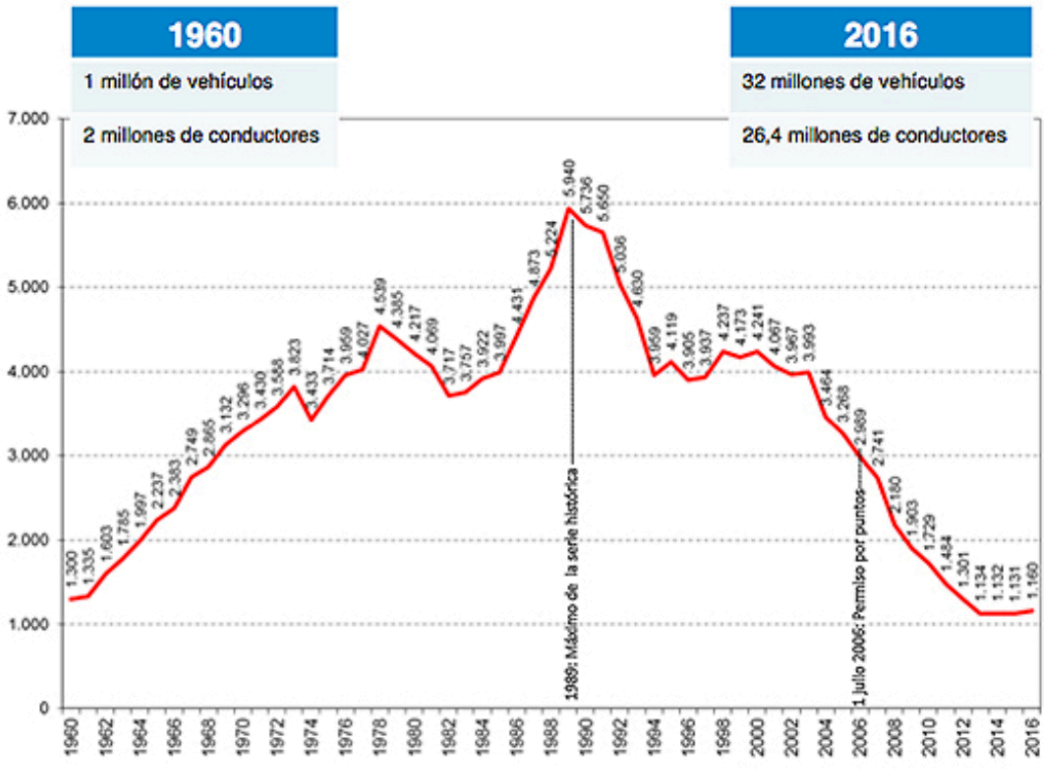


Figura 1. Evolución mortalidad accidentes de tráfico (DGT, 2017)

1.2. Planteamiento del trabajo

Los accidentes de tráfico podrían erradicarse evitando fallos en los vehículos y en la conducción. Sin embargo, estas opciones se antojan prácticamente imposibles a corto y medio plazo, por lo que una buena solución sería poder predecir cuándo es más probable que se produzca un accidente en un punto concreto y poder desplazar agentes de policía y servicios médicos al lugar para prevenir y actuar con rapidez en caso de que ocurra. Basado en este último enfoque, en este trabajo se pretende construir un modelo predictivo mediante minería de datos, técnicas estadísticas y algoritmos de inteligencia artificial que pueda predecir con cierto margen de error la probabilidad de que un accidente se produzca en una determinada zona de Madrid.

1.3. Estructura del trabajo

Tabla 1. Estructura de contenido de la memoria.

1	Introducción.
2	Contexto y estado del arte. Estudios realizados sobre accidentes de tráficos y modelos predictivos desarrollados.
3	Objetivo principal, objetivos específicos y metodología utilizada en el trabajo.
4	Desarrollo de la contribución. Captura de datos, estudio y procesamiento de

	los mismos. Cálculo de regresiones lineales. Cálculo de redes neuronales.
5	Conclusiones y trabajo futuro.
6	Bibliografía.
7	Anexos.

2. Contexto y estado del arte

Como una de las principales causas de muerte externa en los países desarrollados, los accidentes de tráfico son motivo de análisis y estudios desde hace muchos años.

2.1. Contexto

Los accidentes de tráfico son una de las principales causas de muerte externa en España (Dirección General de Tráfico, 2017). En los países desarrollados los gobiernos invierten gran cantidad de recursos para reducir la siniestralidad en las carreteras, consiguiendo que la cifra de accidentes haya disminuido en las últimas décadas.

La seguridad vial es un campo de estudio interesante dado la cantidad de variables que pueden influir en los accidentes de tráfico: climatología, estado de las vías, edad y sexo del conductor, densidad de tráfico, visibilidad, etc.

2.2. Estado del arte

En las últimas décadas se han realizado estudios sobre los accidentes de tráfico. En España concretamente, existen proyectos de diferentes propósitos y campos de estudio relacionados con los accidentes de tráfico.

Un ejemplo de ello es el estudio realizado por M. Ruiz Ramosa, R. Ocaña-Riolab y T. Hermosín Bonoc en su estudio “Evolución de la mortalidad por accidentes de tráfico en Andalucía desde 1975 hasta 2001 y predicción para el año 2004”, en el que se concluye que “la mortalidad por accidentes de tráfico presentó una evolución oscilante, con un importante descenso a partir de 1989. Las tasas de mortalidad aumentan con la edad. Según las predicciones realizadas, en los próximos años continuará la tendencia estable iniciada a partir de 1995”.

Otro ejemplo de la variedad de estudios realizados en España relacionados con los accidentes de tráfico es el trabajo realizado por Francisco J. Soler Flores, José María Pardillo Mayora y Rafael Jurado Piña llamado “Tratamiento de outliers en los modelos de predicción de accidentes de tráfico”, que estudia el tratamiento de outliers en modelos de predicción de accidentes de tráfico y presentan algunos métodos estadísticos para su tratamiento.

En Chile, Cecilia Montt, José Miguel Rubio y Silvana Lanata en su trabajo titulado “Análisis de accidentes de tránsito con inteligencia computacional”, desarrollan e investigan modelos de predicción que estimen el número de personas lesionadas y fallecidas en accidentes de

tráfico. En este proceso utilizan redes neuronales en combinación con otros algoritmos de inteligencia artificial.

Por otra parte, Dominique Lord y Bhagwant N. Persaud en su publicación “Accident Prediction Models With and Without Trend: Application of the Generalized Estimating Equations (GEE) Procedure” presentan la aplicación de ecuaciones generalizadas de estimación (GEE por sus siglas en inglés) para desarrollar un modelo de predicción de accidentes que incorpore tendencias en los datos utilizados. En la publicación se explica que los modelos de predicción de número de accidentes en intersecciones se realizan sobre un histórico de datos que abarca varios años. Sin embargo, hay que tener en cuenta la influencia de parámetros que cambian cada año, por lo que las tendencias hay que aplicarlas por separado para cada uno. El problema que se presenta con esta desagregación es que se crean correlaciones temporales que dificultan el calibrado de los modelos tradicionales.

Otros muchos trabajos publicados profundizan en distintos campos de estudio relacionados con accidentes de tráfico. Algunos ejemplos:

- Estudio de la mortalidad a 30 días por accidentes de tráfico (Catherine Pérez, Eva Cirera, Antoni Plasència).
- El ambiente de tráfico como generador de ansiedad en el conductor: Inventario de situaciones ansiógenas en el tráfico (Enrique Carbonell, Rosa Bañuls, Juan José Miguel-Tobal).
- Modelo de predicción de ocurrencia de accidentes en tramos de carretera mediante la medición continua de variables de influencia (Francisco Javier Páez Ayuso).

A pesar de la multitud de estudios realizados, aún queda mucho camino por recorrer.

3. Objetivos concretos y metodología de trabajo

3.1. Objetivo general

Con el fin de contribuir a disminuir la cifra de accidentes de tráfico en Madrid, este trabajo tiene como objetivo construir un modelo de predicción de accidentes en determinadas zonas de la ciudad utilizando técnicas estadísticas y redes neuronales.

3.2. Objetivos específicos

Como en cualquier trabajo de estas características, es necesario dividir el objetivo principal en objetivos más específicos con el fin de establecer metas reales y planificar el trabajo acorde a estas. Los objetivos específicos de este trabajo son los siguientes:

- Capturar datos sobre accidentes de tráfico en Madrid: los datos sobre accidentes en Madrid son el núcleo del trabajo. Sin estos datos no es posible avanzar hacia el objetivo final. Por tanto, es necesario buscar estos datos en fuentes fiables. Tanto la Dirección General de Tráfico (https://sedeapl.dgt.gob.es/WEB_IEST_CONSULTA/, 2015) como el Ayuntamiento de Madrid (Ayuntamiento de Madrid, 2017) disponen de este tipo de información.
- Capturar datos sobre meteorología: dentro de los datos que están al alcance de cualquier persona están los meteorológicos. Se puede pensar que el clima (precipitaciones, viento y temperatura) puede afectar en la ocurrencia de accidentes de tráfico. Por ello, es importante capturar información meteorológica y, en este caso, el mejor sitio donde obtenerla es la Agencia Estatal de Meteorología, en su sección de Open Data (Agencia Estatal de Meteorología, 2017).
- Capturar datos sobre calendario laboral: otra variable que puede influir en los accidentes de tráfico es el hecho de que el día en el que se produce es laboral o festivo. Desde la web www.calendarioslaborales.com (calendarioslaborales.com, 2012) se puede capturar dicha información.
- Capturar datos sobre el calendario escolar: es bien sabido que, lamentablemente, el calendario laboral es España consta de menos festivos que el calendario escolar. Por este motivo, otra de las posibles variables a utilizar en el modelo de predicción viene dado por el calendario escolar, el cual se obtiene de diferentes fuentes según el año (véase bibliografía).
- Estudiar y tratar los datos. Es necesario estudiar los datos disponibles, decidir cuáles utilizar y tratarlos con el fin de poder aplicar métodos estadísticos y algoritmos de inteligencia artificial.

- Aplicar regresiones. Cuando los datos están preparados para su utilización, lo primero que se hace es aplicar regresiones para poder realizar predicciones
- Aplicar redes neuronales. Tras haber aplicado regresiones, se utilizan redes neuronales, con el mismo objetivo.
- Evaluar los resultados. Con los resultados obtenidos, se evalúan y se comparan unos con otros.
- Establecer posibles mejoras a futuro. Se establece un plan de acción con posibles mejoras tanto en la utilización de variables, tratamiento de datos y elección de técnicas para construir el modelo de predicción.

3.3. Metodología del trabajo

El trabajo comienza con la búsqueda de información sobre accidentes de tráfico en Madrid. Para ello se busca en la sección de Open Data del Ayuntamiento de Madrid (Ayuntamiento de Madrid - Accidentes Tráfico, 2014) y se buscan conjuntos de datos de accidentes de tráficos. Estos datos son necesarios para disponer de un histórico con el que crear el modelo de predicción. En el sitio web del ayuntamiento los datos sobre accidentes de tráfico se encuentran agregados por lo que no son válidos para desarrollar el modelo.

Dado que el histórico de accidentes de tráfico en Madrid es un elemento clave en el desarrollo del trabajo, se contacta con el Ayuntamiento de Madrid para solicitar los datos disponibles en el sitio web, pero a nivel desagregado. Desde el área de “AGSSyE - DGPM / Unidad de Calidad y Evaluación”, facilitan un fichero con los datos sobre accidentes de tráfico con víctimas en Madrid a nivel desagregado.

El siguiente paso es capturar datos para enriquecer los facilitados desde el Ayuntamiento. Se comienza con los festivos y se sigue con datos meteorológicos.

Con los datos ya capturados, se procede a sus estudio y procesamiento (variables dummy, escalado de valores, etc.) para adaptarlos y poder calcular regresiones y redes neuronales a partir de estos. Para este proceso se utiliza Python (Python Software Foundation, 2017) ya que se trata de un lenguaje de programación de alto nivel, lo cual permite desarrollar rápidamente los scripts necesarios. Como editor de código se utiliza Sublime Text 3 (Sublime Text, 2017).

El primer modelo que se construye se calcula aplicando una regresión lineal sobre los datos, utilizando la función *lm()* de R (r-project.org, 2017). Para la edición de código se utiliza el IDE RStudio (rstudio.com, 2017).

A continuación, se procede a calcular redes neuronales artificiales, aplicando validación cruzada (cross validation en inglés) para validar los modelos, dividiendo los datos disponibles en dos conjuntos: conjunto de entrenamiento y conjunto de test. Se utiliza la raíz cuadrada del error cuadrático medio (RMSE por sus siglas en inglés) para comparar los modelos.

Finalmente se evalúan los resultados, se obtienen las conclusiones y se establece un plan de trabajo a futuro con el que mejorar los resultados obtenidos.

4. Desarrollo específico de la contribución

En esta sección se explicarán los pormenores de cada una de las fases necesarias para alcanzar los objetivos específicos descritos en el apartado anterior. Para ello se comenzará explicando de dónde y cómo se han obtenido los datos. A continuación, se describirá la tecnología utilizada en el trabajo. Después, se estudiarán y transformarán los datos para, posteriormente, aplicar regresiones lineales y redes neuronales sobre los mismos.

4.1. Captura de datos

En primer lugar, es necesario disponer de información acerca de los accidentes de tráfico que se han producido en Madrid. En la web del ayuntamiento de Madrid, hay una sección de "open data" (Ayuntamiento de Madrid, 2017) en la que se puede encontrar data sets de diferentes temáticas, tales como urbanismo (Urbanismo Ayuntamiento de Madrid, 2017), deportes (Deportes Ayuntamiento de Madrid, 2017), o circulación y tráfico , entre otros. Un conjunto de datos que, a priori, podría ser de utilidad para este estudio es el de "Accidentes de tráfico. Datos desde 2009" (Ayuntamiento de Madrid - Accidentes Tráfico, 2014), el cual contiene información sobre tipos de accidente, distrito en el que se ha producido, edad de los conductores implicados, horarios y día de la semana. Sin embargo, toda esta información se reporta a nivel agregado (ver *Figura 2*), por lo que no son de gran utilidad para construir un modelo predictivo.

Indicadores	Nº Accidentes											
Año	2016											Total
DISTRITO_ACCIDENTE	COLISIÓN DOBLE	COLISIÓN MÚLTIPLE	CHOQUE CON OBJETO FIJO	ATROPELLO	VUELCO	CAÍDA MOTOCICLETA	CAÍDA CICLOMOTOR	CAÍDA BICICLETA	CAÍDA VIAJERO BUS	OTRAS CAUSAS	CAÍDA VEHÍCULO 3 RUEDAS	
ARGANZUELA	328	62	86	57	1	46	12	10	6	2		610
BARAJAS	95	6	50	34	4	12	2	4	2	4		213
CARABANCHEL	346	55	121	106	5	53	18	7	8	6		725
CENTRO	444	23	172	144		97	11	33	5	3		932
CHAMARTIN	499	59	122	80	5	105	14	8	5	7		904
CHAMBERI	382	26	53	88		63	10	19	1	5		647
CIUDAD LINEAL	411	57	119	85	4	75	15	8	5	8		787
FUENCARRAL-EL PARDO	345	30	132	70	6	56	18	28	3	4		692
HORTALEZA	248	19	90	56	5	33	9	10	1	1		472
LATINA	288	40	119	104	3	36	12	11	7	2		622
MONCLOA-ARAVACA	294	61	182	54	12	68	8	27	2	1		709
MORATALAZ	150	31	53	32	5	23	5	2	4	2		307
PUENTE DE VALLECAS	386	81	140	118	8	41	11	11	10	5		811
RETIRO	344	69	63	56	1	67	9	33	1	3	1	647
SALAMANCA	467	65	111	112	1	142	14	22	14	3		951
SAN BLAS	299	20	57	68	3	34	6	9	5			501
TETUAN	330	43	70	91		73	25	13	5	3		653
USERA	206	27	74	75	2	16	2	5	4	1		412
VICALVARO	70	4	38	22	4	15	2	3	2	1		161
VILLA DE VALLECAS	149	7	42	54	5	26	4	4	1	3		295
VILLAVERDE	177	11	71	54	1	8	5	6	2	1		336
Total	6.258	796	1.965	1.560	75	1.089	212	273	93	65	1	12.387

Figura 2. Ejemplo datos agregados de accidentes de tráfico en Madrid.

Disponer de datos desagregados de accidentes de tráfico es muy importante para llevar a cabo el estudio y poder construir un modelo de predicción. Por ello, el autor se puso en contacto con el Ayuntamiento de Madrid que, desde el área de "AGSSyE - DGPM / Unidad de Calidad y Evaluación", facilitó información desagregada de los accidentes de tráfico con víctimas ocurridos en Madrid desde enero de 2012 hasta junio de 2017. El conjunto de datos

facilitado consta de *fecha* del accidente, *rango horario* en el que se produce el accidente, *número de víctimas*, *lugar*, *número* de la calle o punto kilométrico de la vía en el que tiene lugar el accidente y *tipo de accidente* (ver *Figura 3*).

```
Fecha,RangoHorario,NumVictimas,Lugar,Numero,TipoAccidente
Domingo 01 Enero 2012,DE 2:00 A 2:59,1,AVENIDA DE SAN DIEGO NUM,96,OTRAS CAUSAS
Domingo 01 Enero 2012,DE 2:00 A 2:59,3,CALLE DE BRAVO MURILLO - CALLE DE LAS CAROLINAS,0,COLISIÓN DOBLE
Domingo 01 Enero 2012,DE 2:00 A 2:59,2,CALLE DE LA PRINCESA NUM,10,ATROPELLO
Domingo 01 Enero 2012,DE 5:00 A 5:59,1,CALLE DE ALCALA - AUTOVIA M-30 CALZADA 1,0,COLISIÓN MÚLTIPLE
Domingo 01 Enero 2012,DE 7:00 A 7:59,1,AVENIDA DE LOS POBLADOS - CALLE DEL PARQUE EUGENIA DE MONTIJO,0,CAÍDA VIAJERO BUS
Domingo 01 Enero 2012,DE 7:00 A 7:59,1,CALLE DE O'DONNELL NUM,53,ATROPELLO
Domingo 01 Enero 2012,DE 18:00 A 18:59,1,AVENIDA DE AMERICA - CALLE DE CARTAGENA,0,ATROPELLO
Domingo 01 Enero 2012,DE 18:00 A 18:59,2,CALLE DE JOAQUIN COSTA NUM,49,ATROPELLO
Domingo 01 Enero 2012,DE 19:00 A 19:59,1,CALLE DEL GENERAL RICARDOS NUM,33,COLISIÓN DOBLE
Domingo 01 Enero 2012,DE 20:00 A 20:59,1,CARRETERA DE BOADILLA DEL MONTE NUM,1,COLISIÓN DOBLE
Domingo 01 Enero 2012,DE 22:00 A 22:59,1,CALLE DE LA VILLA DE MARIN NUM,34,COLISIÓN DOBLE
Domingo 01 Enero 2012,DE 23:00 A 23:59,1,PASEO DE SANTA MARIA DE LA CABEZA - CALLE DE PALOS DE LA FRONTERA,0,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 4:00 A 4:59,1,CALLE DE BALEARES NUM,33,CHOQUE CON OBJETO FIJO
Lunes 02 Enero 2012,DE 10:00 A 10:59,1,AVENIDA DEL SANTUARIO DE VALVERDE NUM,99,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 10:00 A 10:59,1,PLAZA DE LA INDEPENDENCIA - CALLE DE ALFONSO XII,0,CAÍDA MOTOCICLETA
Lunes 02 Enero 2012,DE 12:00 A 12:59,1,CALLE DE LA ESFINGE - CALLE DEL DISCOBOLO,0,CHOQUE CON OBJETO FIJO
Lunes 02 Enero 2012,DE 13:00 A 13:59,2,AVENIDA DE LA PRINCESA JUANA DE AUSTRIA KM.,4700,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 13:00 A 13:59,1,PASEO DE LOS OLMOS NUM,13,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 13:00 A 13:59,1,PLAZA DE LA BEATA MARIA ANA DE JESUS NUM,6,CAÍDA CICLOMOTOR
Lunes 02 Enero 2012,DE 14:00 A 14:59,2,PLAZA DEL CONDE DE CASAL NUM,5,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 14:00 A 14:59,1,CALLE DEL GENERAL ALVAREZ DE CASTRO NUM,24,CAÍDA MOTOCICLETA
Lunes 02 Enero 2012,DE 16:00 A 16:59,1,CALLE DE EUGENIA DE MONTIJO - AVENIDA DE NUESTRA SEÑORA DE FATIMA,0,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 16:00 A 16:59,1,CALLE DE ALCALA - AVENIDA DE CANILLEJAS A VICALVARO,0,CHOQUE CON OBJETO FIJO
Lunes 02 Enero 2012,DE 16:00 A 16:59,1,CUESTA DEL SAGRADO CORAZON - CALLE 30 08RE,0,COLISIÓN MÚLTIPLE
Lunes 02 Enero 2012,DE 16:00 A 16:59,1,CALLE DEL GENERAL RICARDOS - CALLE DE RIO DE ORO,0,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 18:00 A 18:59,3,CALLE DEL PUERTO DE ARLABAN - CALLE DE LA SIERRA DE ALQUIFE,0,COLISIÓN MÚLTIPLE
Lunes 02 Enero 2012,DE 18:00 A 18:59,1,CALLE DEL PRINCIPE DE VERGARA - PLAZA DE LA REPUBLICA DEL ECUADOR,0,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 19:00 A 19:59,1,CALLE DE ARIADNA - AUTOVIA M-40,0,COLISIÓN MÚLTIPLE
Lunes 02 Enero 2012,DE 19:00 A 19:59,1,CALLE DE ALCALA NUM,728,COLISIÓN MÚLTIPLE
Lunes 02 Enero 2012,DE 20:00 A 20:59,1,AVENIDA DE LA ALBUFERA - CALLE DE BUSTOS,0,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 20:00 A 20:59,1,CALLE DEL ALTO DEL RETIRO NUM,33,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 22:00 A 22:59,1,CALLE DE SIERRA TOLEDANA NUM,41,CAÍDA MOTOCICLETA
Lunes 02 Enero 2012,DE 22:00 A 22:59,1,CALLE DE LA CONDESA DE VENADITO NUM,14,COLISIÓN DOBLE
Martes 03 Enero 2012,DE 00:00 A 00:59,1,CALLE DEL LAGO CONSTANZA NUM,22,COLISIÓN DOBLE
Martes 03 Enero 2012,DE 7:00 A 7:59,1,CALLE DE LA FUENTE CARRANTONA NUM,57,ATROPELLO
Martes 03 Enero 2012,DE 8:00 A 8:59,1,CALLE DEL PRINCIPE DE VERGARA - CALLE DE PADILLA,0,COLISIÓN DOBLE
Martes 03 Enero 2012,DE 8:00 A 8:59,1,CALLE DEL COCHERON DE LA VILLA NUM,2,COLISIÓN DOBLE
```

Figura 3. Ejemplo de información desagregada de accidentes de tráfico.

Contando ya con los datos básicos sobre los que construir el modelo, el siguiente objetivo es disponer de datos sobre meteorología. Para ello se accede al sitio web de la Agencia Estatal de Meteorología, concretamente al centro de descargas de su sección de Open Data (Agencia Estatal de Meteorología, 2017). Aquí se puede descargar información relacionada con redes de rayos, información de satélites, predicción climática, etc. La información que interesa para este trabajo es la referente a los datos climatológicos históricos desde enero de 2012 hasta junio de 2017. Como se puede observar en la *Figura 4*, el formulario de la web permite seleccionar para climatologías diarias la provincia (Madrid), la estación meteorológica (Ciudad Universitaria) que recoge los datos y un intervalo de fechas (enero 2012 – junio 2017).

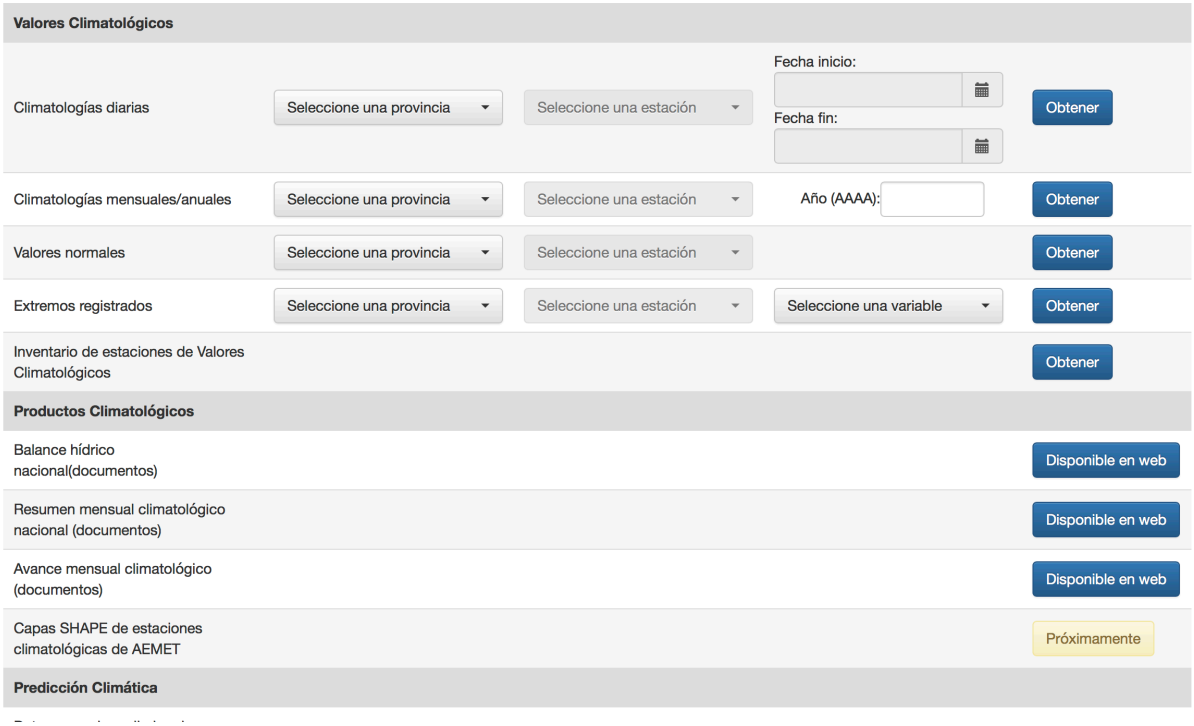


Figura 4. Ejemplo de formulario de extracción de datos de la web de AEMET.

Los datos extraídos de este sitio web se encuentran en formato JSON, por lo que se pueden tratar con cualquier tecnología de análisis de datos, y están compuestos por los siguientes campos (ver *Figura 5*): fecha, identificador, nombre de la estación meteorológica, provincia, altitud, temperatura media del día, temperatura máxima del día, temperatura mínima del día, hora en la que se ha recogido la temperatura máxima, hora correspondiente a la temperatura mínima, dirección de la racha máxima de viento, velocidad media del viento, racha máxima del viento, hora en a la que se ha producido la racha máxima.

Tras obtener los datos de accidentes de tráfico y meteorológicos, se procede a capturar datos del calendario laboral. Para ello se accede a la web http://www.calendarioslaborales.com/ donde se puede seleccionar la provincia y el año para obtener el calendario laboral correspondiente (ver *Figura 5*). En este caso, se ha elaborado a mano un fichero CSV con los días festivos de cada año (ver *Figura 6*). Lo mismo se hace con los festivos del calendario escolar (ver *Figura 6*).

Figura 5. Sitio web calendario laboral (calendarioslaborales.com, 2012)

```
Year,Month,Day,Date,WeekDayNum,WeekDay
2012,1,6,06/01/2012,6,Viernes
2012,3,19,19/03/2012,2,Lunes
2012,4,5,05/04/2012,5,Jueves
2012,4,6,06/04/2012,6,Viernes
2012,5,1,01/05/2012,3,Martes
2012,5,2,02/05/2012,4,Miércoles
2012,5,15,15/05/2012,3,Martes
2012,8,15,15/08/2012,4,Miércoles
2012,10,12,12/10/2012,6,Viernes
2012,11,1,01/11/2012,5,Jueves
2012,11,9,09/11/2012,6,Viernes
2012,12,6,06/12/2012,5,Jueves
2012,12,8,08/12/2012,7,Sábado
2012,12,25,25/12/2012,3,Martes
2013,1,1,01/01/2013,3,Martes
2013,1,7,07/01/2013,2,Lunes
2013,3,18,18/03/2013,2,Lunes
2013,3,28,28/03/2013,5,Jueves
2013,3,29,29/03/2013,6,Viernes
2013,5,1,01/05/2013,4,Miércoles
2013,5,2,02/05/2013,5,Jueves
2013,5,15,15/05/2013,4,Miércoles
2013,8,15,15/08/2013,5,Jueves
2013,10,12,12/10/2013,7,Sábado
2013,11,1,01/11/2013,6,Viernes
2013,11,9,09/11/2013,7,Sábado
2013,12,6,06/12/2013,6,Viernes
2013,12,25,25/12/2013,4,Miércoles
2014,1,1,01/01/2014,4,Miércoles
2014,1,6,06/01/2014,2,Lunes
2014,4,17,17/04/2014,5,Jueves
2014,4,18,18/04/2014,6,Viernes
2014,5,1,01/05/2014,5,Jueves
```

```
1  Year,Month,Day,Date,WeekDayNum,WeekDay
2  2012,1,2,02/01/2012,2,Lunes
3  2012,1,3,03/01/2012,3,Martes
4  2012,1,4,04/01/2012,4,Miércoles
5  2012,1,5,05/01/2012,5,Jueves
6  2012,1,6,06/01/2012,6,Viernes
7  2012,3,19,19/03/2012,2,Lunes
8  2012,3,30,30/03/2012,6,Viernes
9  2012,4,2,02/04/2012,2,Lunes
10 2012,4,3,03/04/2012,3,Martes
11 2012,4,4,04/04/2012,4,Miércoles
12 2012,4,5,05/04/2012,5,Jueves
13 2012,4,6,06/04/2012,6,Viernes
14 2012,4,9,09/04/2012,2,Lunes
15 2012,4,30,30/04/2012,2,Lunes
16 2012,5,1,01/05/2012,3,Martes
17 2012,5,2,02/05/2012,4,Miércoles
18 2012,6,27,27/06/2012,4,Miércoles
19 2012,6,28,28/06/2012,5,Jueves
20 2012,6,29,29/06/2012,6,Viernes
21 2012,7,1,01/07/2012,1,Domingo
22 2012,7,2,02/07/2012,2,Lunes
23 2012,7,3,03/07/2012,3,Martes
24 2012,7,4,04/07/2012,4,Miércoles
25 2012,7,5,05/07/2012,5,Jueves
26 2012,7,6,06/07/2012,6,Viernes
27 2012,7,7,07/07/2012,7,Sábado
28 2012,7,8,08/07/2012,1,Domingo
29 2012,7,9,09/07/2012,2,Lunes
30 2012,7,10,10/07/2012,3,Martes
31 2012,7,11,11/07/2012,4,Miércoles
32 2012,7,12,12/07/2012,5,Jueves
33 2012,7,13,13/07/2012,6,Viernes
34 2012,7,14,14/07/2012,7,Sábado
```

Figura 6. Formato CSV de festivos (izquierda) y festivo escolares (derecha)

4.2. Tecnología

En este trabajo se han utilizado dos lenguajes de programación: Python y R. Para trabajar con Python se ha utilizado SublimeText, mientras que para trabajar con R se ha utilizado RStudio. A continuación, se profundiza en estas tecnologías.

Lenguaje Python

Python es, tal como se explica en el libro "Python 3: los fundamentos del lenguaje" (Chazallet, 2016), un lenguaje de programación interpretado, multiplataforma y de alto nivel. Es libre y gratuito, funciona en todas las plataformas. Está diseñado para que sea simple y permite al programador abstraerse de problemas de bajo nivel, produciendo una sensación confortable ya que Python resolverá dichos problemas de la mejor manera posible, lo que mejora considerablemente la productividad.

Python fue creado en 1990 por Guido van Rossum como sucesor de un lenguaje llamado ABC. En 2001 se creó Python Software Foundation (una organización sin ánimo de lucro) específicamente para poseer la propiedad intelectual de Python. Todas las versiones de Python se distribuyen bajo licencias compatibles con GPL (Python Software Foundation, 2017).

Figura 7. Logo de Python (Python Software Foundation, 2017).

Se ha elegido Python para este trabajo por ser gratuito, multiplataforma y de alto nivel. Este último atributo permite desarrollar en poco tiempo prototipos software.

Lo primero que hay que hacer para comenzar a utilizar Python es descargar una versión. Para ello se accede al sitio web de Python Software Foundation (Python Software Foundation, 2017) y se descarga una versión compatible con la plataforma que se vaya a usar (ver *Figura 8*).

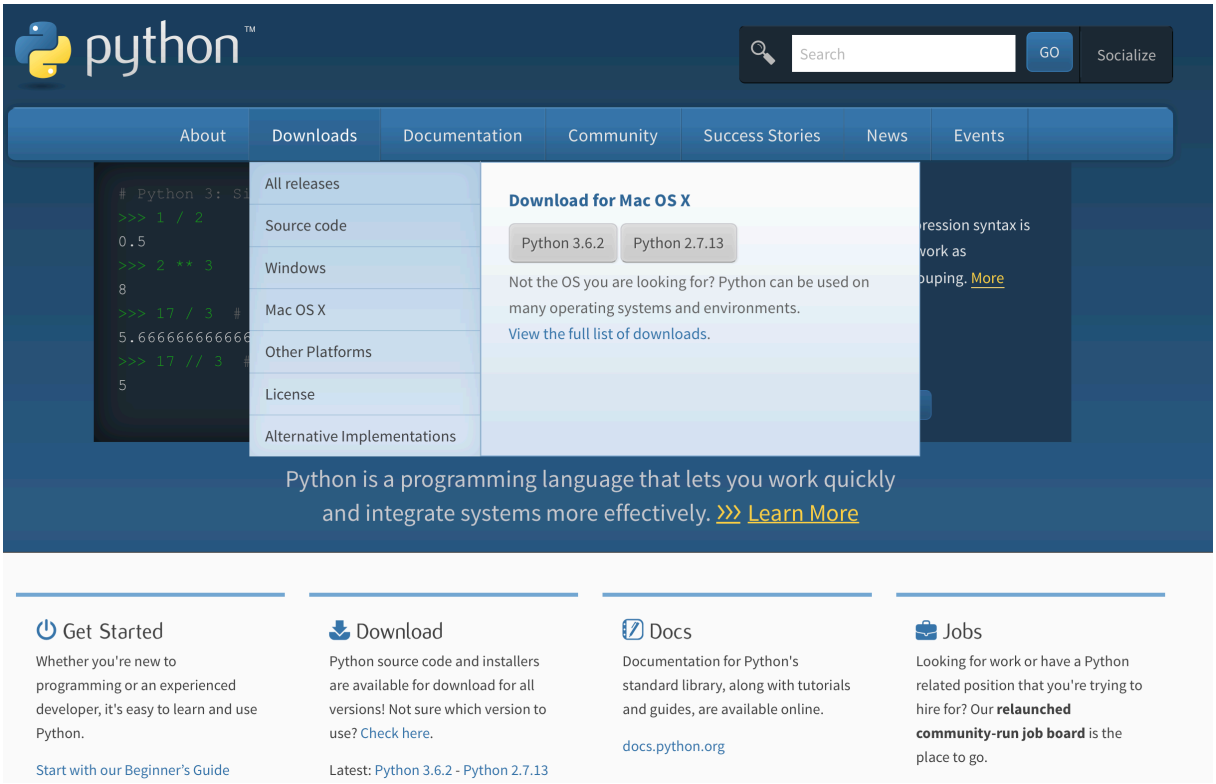


Figura 8. Sitio web Python Software Foundation – Downloads

Se descarga un fichero que, al ejecutarse, utiliza un asistente para la instalación del lenguaje. En caso de querer instalar Python en alguna plataforma que no sea Windows o Mac OS X, en el siguiente enlace se explica cómo proceder: https://tutorial.djangogirls.org/es/python_installation/ (djangogirls.org, 2017).

Sublime Text como editor de Python

Sublime Text es un editor de código avanzado. Se puede descargar de su sitio web (Sublime Text, 2017) y se puede probar de manera gratuita. Permite una visualización del texto con colores (ver *Figura 9*), lo cual ayuda al desarrollador. Soporta multitud de lenguajes y, en el caso de Python, permite ejecutar el código directamente desde el editor.

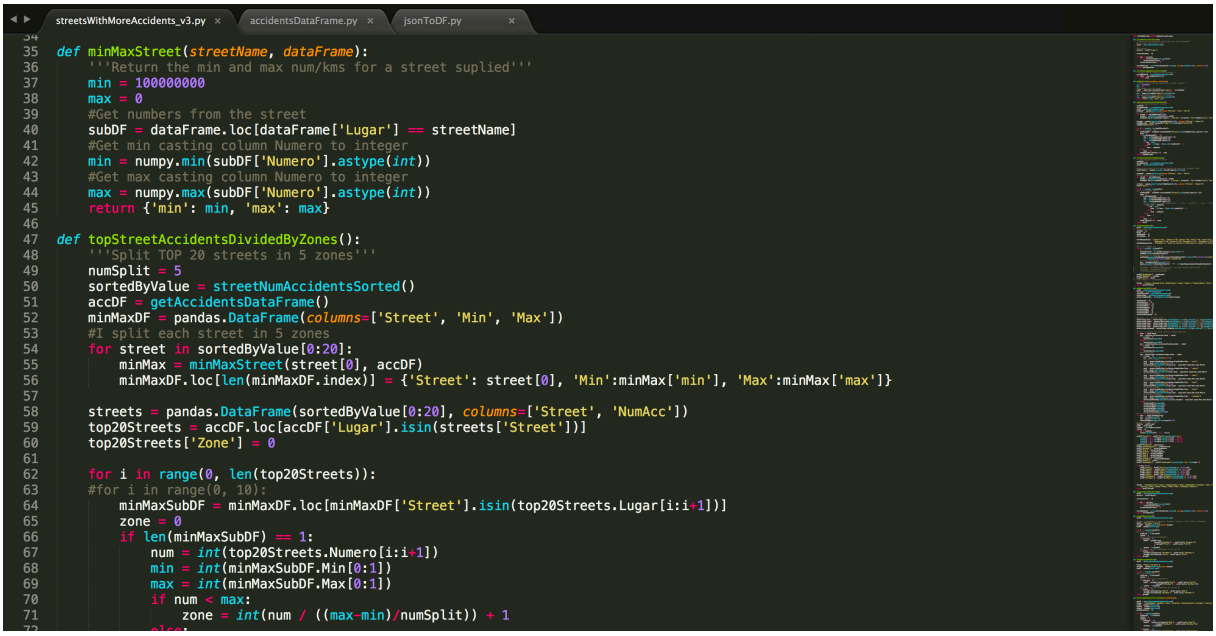


Figura 9. Editor de código Sublime Text

La instalación de Sublime Text es sencilla. Primero se tiene que descargar del sitio web https://www.sublimetext.com/3 (ver *Figura* 10) eligiendo la plataforma. A continuación, únicamente hay que ejecutar el fichero seleccionado y seguir los pasos del instalador (Windows y Mac OS X).

Figura 10. Sitio web para descargar Sublime Text (sublimetext.com, 2017) .

Lenguaje R

R es una lenguaje y entorno orientados al análisis estadístico y visualización de datos. Es un proyecto GNU similar al lenguaje S, desarrollado por John Chambers. R ofrece una amplia variedad de técnicas estadísticas y de visualización de datos y está disponible como software libre bajo los términos de *Free Software Foundation's GNU General Public License* (GNU, 2017) y para la mayoría de sistemas UNIX, Windows y Mac OS (r-project.org, 2017).

Figura 11. Logo de R (r-project.org, 2017)

Para instalar R debemos dirigirnos a la página oficial (r-project.org, 2017) y pinchar sobre el enlace *download R* en el primer párrafo de la sección “Getting Started” (ver *Figura 12*).

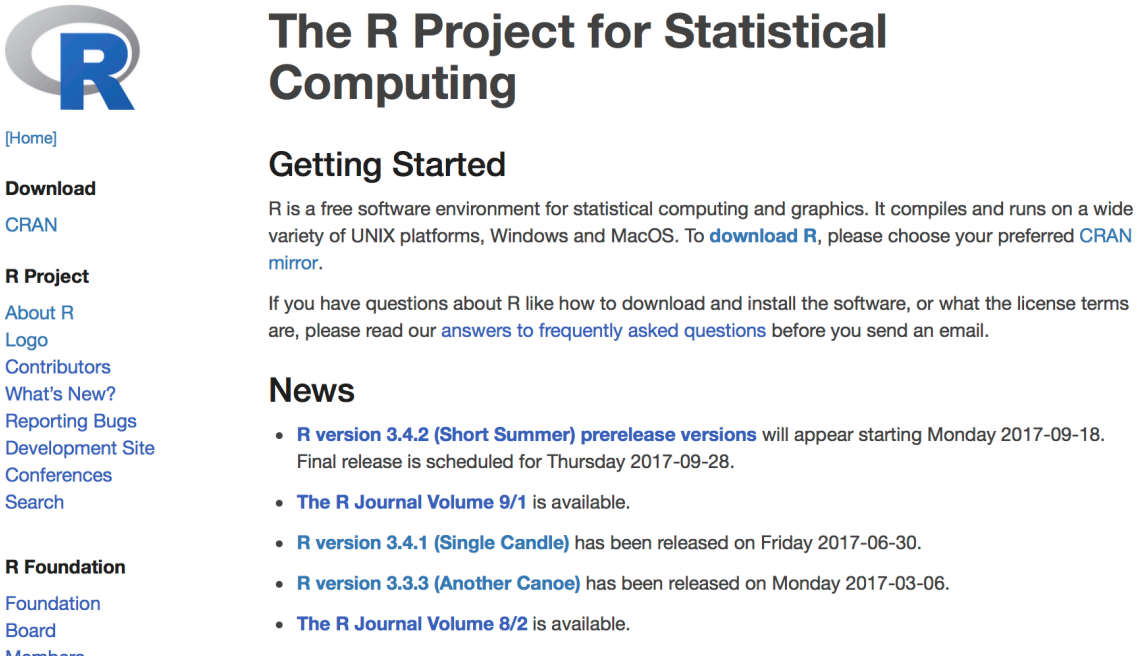


Figura 12. Instalando R – Descarga (1)

A continuación, se debe elegir el servidor desde el que queramos realizar la descarga. En este caso hemos seleccionado el correspondiente a la Oficina de software libre (CIXUG) de España (ver *Figura 13*).

South Africa	
http://r.adu.org.za/	University of Cape Town
http://cran.mirror.ac.za/	TENET, Johannesburg
Spain	
https://ftp.cixug.es/CRAN/	Oficina de software libre (CIXUG)
http://ftp.cixug.es/CRAN/	Oficina de software libre (CIXUG)
https://cran.rediris.es/	Spanish National Research Network, Madrid
http://cran.rediris.es/	Spanish National Research Network, Madrid
Sweden	
https://ftp.acc.umu.se/mirror/CRAN/	Academic Computer Club, Umeå University
http://ftp.acc.umu.se/mirror/CRAN/	Academic Computer Club, Umeå University
Switzerland	
https://stat.ethz.ch/CRAN/	ETH Zürich
http://stat.ethz.ch/CRAN/	ETH Zürich
Taiwan	
https://ftp.yzu.edu.tw/CRAN/	Department of Computer Science and Engineering, Yuan Ze University
http://ftp.yzu.edu.tw/CRAN/	Department of Computer Science and Engineering, Yuan Ze University
http://cran.csie.ntu.edu.tw/	National Taiwan University, Taipei
Thailand	
http://mirrors.psu.ac.th/pub/cran/	Prince of Songkla University, Hatyai
Turkey	
https://cran.pau.edu.tr/	Pamukkale University, Denizli
http://cran.pau.edu.tr/	Pamukkale University, Denizli

Figura 13. Instalando R – Descarga (2)

En este paso se ha de elegir la plataforma sobre la que deseamos instalar R (ver *Figura 14*). Para finalizar la instalación, tan solo es necesario seguir los pasos indicados para la plataforma correspondiente.

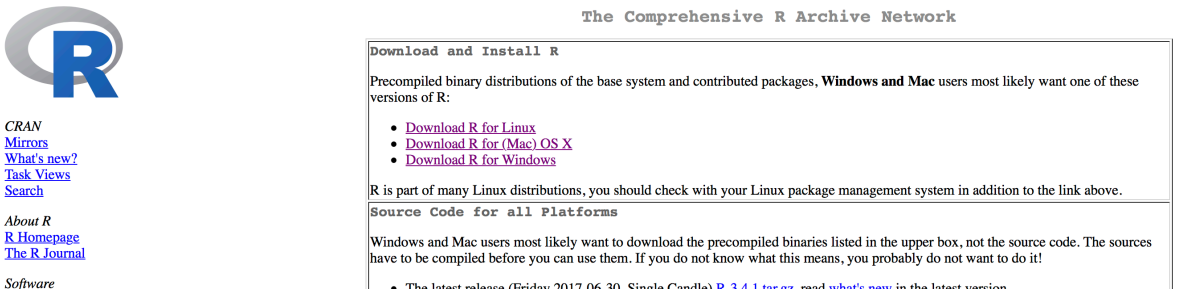


Figure 14. Instalando R – Descarga (3)

RStudio

Como se explica en el libro “Getting Started with RStudio” (Verzani, 2011), RStudio es un entorno de desarrollo integrado (IDE por sus siglas en inglés) para R. Dispone de cuatro paneles (ver *Figura 16*):

- Panel superior izquierdo: editor de código con coloreado de palabras clave. Este editor consta de diferentes pestañas con las que puedes acceder a cada uno de los scripts que se estén editando.
- Panel inferior izquierdo: consola para sesiones interactivas de R.
- Panel superior derecho: gestor de variables en el que aparece un listado de las variables que están siendo usadas en la sesión. En este panel también se puede acceder al histórico de comandos utilizados en la sesión. Además, permite exportar e importar el espacio de trabajo (conjunto de variables).

- Panel inferior derecho: visualizador de ficheros, paquetes y gráficos.

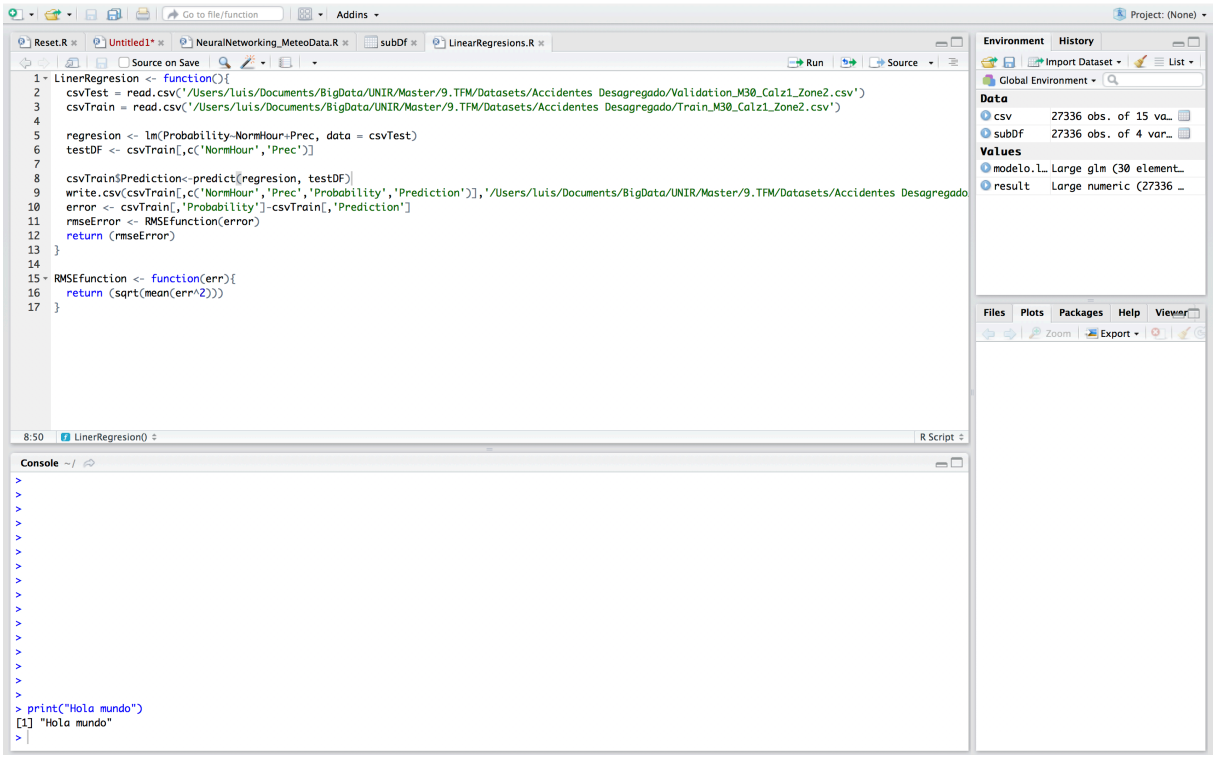


Figura 15. IDE RStudio

Tanto la consola como el editor de código están enlazados con el sistema de ayuda interno de R, ofreciendo la opción de autocompletado. RStudio puede ser utilizado como aplicación de escritorio en los tres sistemas operativos principales, así como en un navegador web para accesos remotos. Posee una licencia Open Source para fines no comerciales. Se puede descargar un instalador en el siguiente enlace https://www.rstudio.com/products/rstudio/download/ (ver *Figura 16*).

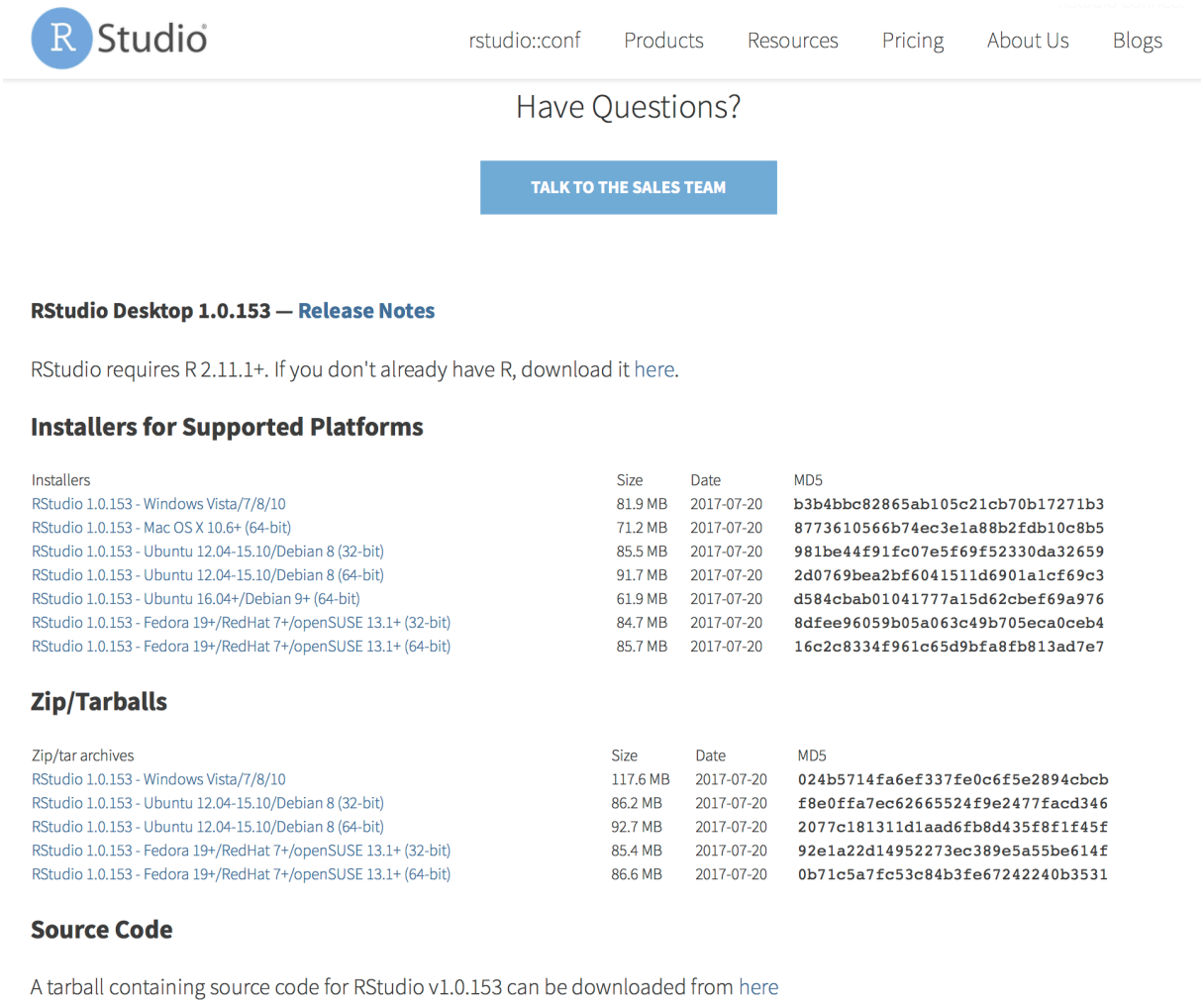


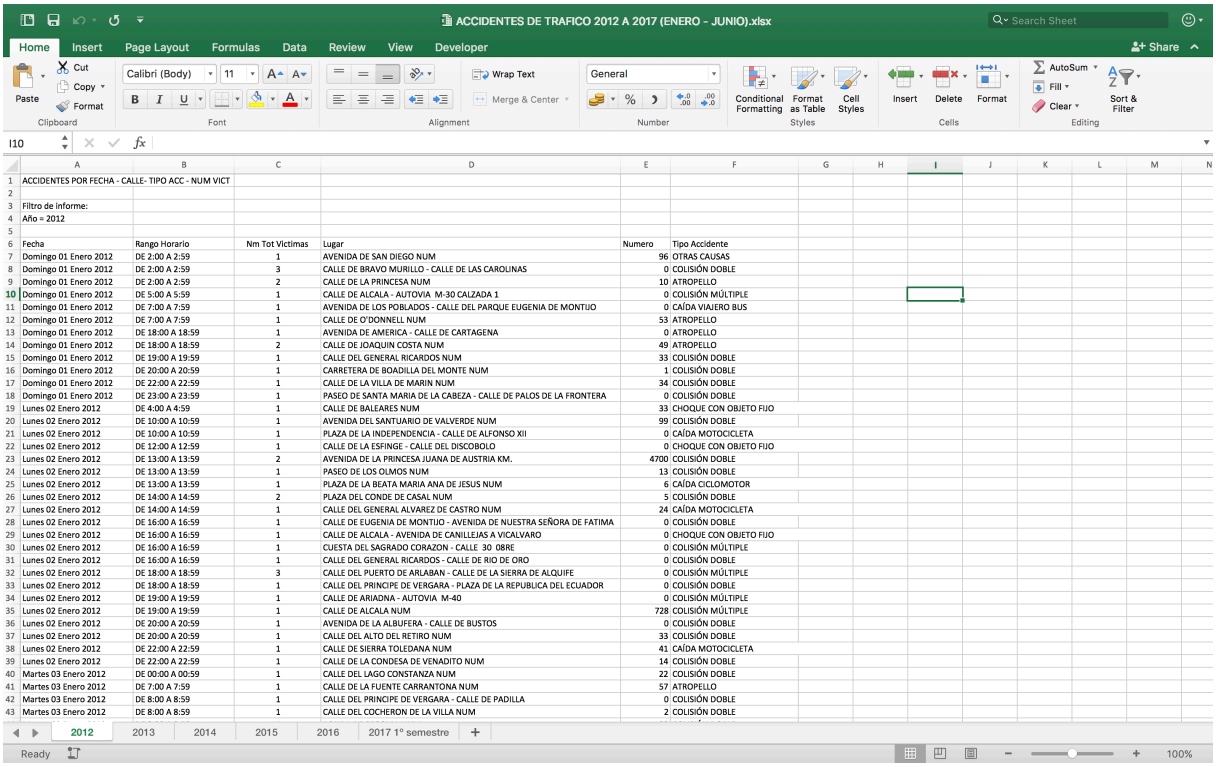
Figura 16. Sitio web de RStudio – descarga (rstudio.com, 2017).

4.3. Estudio y transformación de los datos

En las anteriores secciones se han explicado las fuentes de información y la tecnología utilizada para realizar el trabajo. El siguiente paso es el estudio de la información de la que se dispone para, posteriormente, transformarla en datos que sean de utilidad.

Datos de accidentes de tráfico

El primer conjunto de datos que se va a estudiar es el de los accidentes de tráfico proporcionado por el área de "AGSSyE - DGPM / Unidad de Calidad y Evaluación" del Ayuntamiento de Madrid. El fichero facilitado es un archivo Excel con seis pestañas, una por cada año (ver *Figura 17*). Cada pestaña contiene información sobre la fecha, rango horario, calle o vía y número o punto kilométrico de la vía en la que se produce el accidente. Además, indica el número de víctimas y el tipo de accidente.



ACCIDENTES DE TRAFICO 2012 A 2017 (ENERO - JUNIO).xlsx

ACCIDENTES POR FECHA - CALLE- TIPO ACC - NUM VICT

Filtro de informe:
Año = 2012

Fecha	Rango Horario	Nm Tot Victimas	Lugar	Numero	Tipo Accidente
Domingo 01 Enero 2012	DE 2:00 A 2:59	1	AVENIDA DE SAN DIEGO NUM	96	OTRAS CAUSAS
Domingo 01 Enero 2012	DE 2:00 A 2:59	3	CALLE DE BRAVO MURILLO - CALLE DE LAS CAROLINAS	0	COLISIÓN DOBLE
Domingo 01 Enero 2012	DE 2:00 A 2:59	2	CALLE DE LA PRINCESA NUM	10	ATROPELLO
Domingo 01 Enero 2012	DE 5:00 A 5:59	1	CALLE DE ALCALA - AUTOVIA M-30 CALZADA 1	0	COLISIÓN MÚLTIPLE
Domingo 01 Enero 2012	DE 7:00 A 7:59	1	AVENIDA DE LOS POBLADOS - CALLE DEL PARQUE EUGENIA DE MONTIJO	0	CAÍDA VIAJERO BUS
Domingo 01 Enero 2012	DE 7:00 A 7:59	1	CALLE DE O'DONNELL NUM	53	ATROPELLO
Domingo 01 Enero 2012	DE 18:00 A 18:59	1	AVENIDA DE AMERICA - CALLE DE CARTAGENA	0	ATROPELLO
Domingo 01 Enero 2012	DE 18:00 A 18:59	2	CALLE DE JOAQUIN COSTA NUM	49	ATROPELLO
Domingo 01 Enero 2012	DE 19:00 A 19:59	1	CALLE DEL GENERAL RICARDOS NUM	33	COLISIÓN DOBLE
Domingo 01 Enero 2012	DE 20:00 A 20:59	1	CARRETERA DE BOADILLA DEL MONTE NUM	1	COLISIÓN DOBLE
Domingo 01 Enero 2012	DE 22:00 A 22:59	1	CALLE DE LA VILLA DE MARIN NUM	34	COLISIÓN DOBLE
Domingo 01 Enero 2012	DE 23:00 A 23:59	1	PASEO DE SANTA MARIA DE LA CABEZA - CALLE DE PALOS DE LA FRONTERA	0	COLISIÓN DOBLE
Lunes 02 Enero 2012	DE 4:00 A 4:59	1	CALLE DE BALEARES NUM	33	CHOQUE CON OBJETO FIJO
Lunes 02 Enero 2012	DE 10:00 A 10:59	1	AVENIDA DEL SANTUARIO DE VALVERDE NUM	99	COLISIÓN DOBLE
Lunes 02 Enero 2012	DE 10:00 A 10:59	1	PLAZA DE LA INDEPENDENCIA - CALLE DE ALFONSO XII	0	CAÍDA MOTOCICLETA
Lunes 02 Enero 2012	DE 12:00 A 12:59	1	CALLE DE LA ESFINGE - CALLE DEL DISCOBOLO	0	CHOQUE CON OBJETO FIJO
Lunes 02 Enero 2012	DE 13:00 A 13:59	2	AVENIDA DE LA PRINCESA JUANA DE AUSTRIA KM.	4700	COLISIÓN DOBLE
Lunes 02 Enero 2012	DE 13:00 A 13:59	1	PASEO DE LOS OLMOS NUM	13	COLISIÓN DOBLE
Lunes 02 Enero 2012	DE 13:00 A 13:59	1	PLAZA DE LA BEATA MARIA ANA DE JESUS NUM	6	CAÍDA CICLOMOTOR
Lunes 02 Enero 2012	DE 14:00 A 14:59	2	PLAZA DEL CONDE DE CASAL NUM	5	COLISIÓN DOBLE
Lunes 02 Enero 2012	DE 14:00 A 14:59	1	CALLE DEL GENERAL ALVAREZ DE CASTRO NUM	24	CAÍDA MOTOCICLETA
Lunes 02 Enero 2012	DE 16:00 A 16:59	1	CALLE DE EUGENIA DE MONTIJO - AVENIDA DE NUESTRA SEÑORA DE FATIMA	0	COLISIÓN DOBLE
Lunes 02 Enero 2012	DE 16:00 A 16:59	1	CALLE DE ALCALA - AVENIDA DE CANILLEJAS A VICALVARO	0	CHOQUE CON OBJETO FIJO
Lunes 02 Enero 2012	DE 16:00 A 16:59	1	CUESTA DEL SAGRADO CORAZON - CALLE 30 08RE	0	COLISIÓN MÚLTIPLE
Lunes 02 Enero 2012	DE 16:00 A 16:59	1	CALLE DEL GENERAL RICARDOS - CALLE DE RIO DE ORO	0	COLISIÓN DOBLE
Lunes 02 Enero 2012	DE 18:00 A 18:59	3	CALLE DEL PUERTO DE ARLABAN - CALLE DE LA SIERRA DE ALQUIFE	0	COLISIÓN MÚLTIPLE
Lunes 02 Enero 2012	DE 18:00 A 18:59	1	CALLE DEL PRINCIPE DE VERGARA - PLAZA DE LA REPUBLICA DEL ECUADOR	0	COLISIÓN DOBLE
Lunes 02 Enero 2012	DE 19:00 A 19:59	1	CALLE DE ARIADNA - AUTOVIA M-40	0	COLISIÓN MÚLTIPLE
Lunes 02 Enero 2012	DE 19:00 A 19:59	1	CALLE DE ALCALA NUM	728	COLISIÓN MÚLTIPLE
Lunes 02 Enero 2012	DE 20:00 A 20:59	1	AVENIDA DE LA ALBUFERA - CALLE DE BUSTOS	0	COLISIÓN DOBLE
Lunes 02 Enero 2012	DE 20:00 A 20:59	1	CALLE DEL ALTO DEL RETIRO NUM	33	COLISIÓN DOBLE
Lunes 02 Enero 2012	DE 22:00 A 22:59	1	CALLE DE SIERRA TOLEDANA NUM	41	CAÍDA MOTOCICLETA
Lunes 02 Enero 2012	DE 22:00 A 22:59	1	CALLE DE LA CONDESA DE VENADITO NUM	14	COLISIÓN DOBLE
Martes 03 Enero 2012	DE 00:00 A 00:59	1	CALLE DEL LAGO CONSTANZA NUM	22	COLISIÓN DOBLE
Martes 03 Enero 2012	DE 7:00 A 7:59	1	CALLE DE LA FUENTE CARRANTONA NUM	57	ATROPELLO
Martes 03 Enero 2012	DE 8:00 A 8:59	1	CALLE DEL PRINCIPE DE VERGARA - CALLE DE PADILLA	0	COLISIÓN DOBLE
Martes 03 Enero 2012	DE 8:00 A 8:59	1	CALLE DEL COCHERON DE LA VILLA NUM	2	COLISIÓN DOBLE

Figura 17. Fichero facilitado por "AGSSyE - DGPM / Unidad de Calidad y Evaluación".

Si se analiza cada campo con detalle, se obtienen las siguientes conclusiones:

- Fecha: es un campo de texto que contiene la fecha en la que se produce el accidente. El formato es el siguiente: "Lunes 02 Enero 2012", indicando día de la semana, día del mes, mes y año. El rango de fechas utilizados comienza con "Domingo 01 Enero 2012" y finaliza con "Viernes 30 Junio 2017".
- Rango Horario: es un campo de texto que contiene el rango horario en el que tiene lugar el accidente. Cada rango horario es de una hora, comenzando con una hora y cero minutos y finalizando con la misma hora y cincuenta y nueve minutos. El valor mínimo que toma es "DE 0:00 A 0:59" y el máximo es "DE 23:00 A 23:59".
- Número de víctimas: es un campo numérico y contiene el número de víctimas que se ven envueltas en el accidente. El valor mínimo que toma es 1 (dado que el conjunto contiene los accidentes de tráfico con víctimas ocurridos en Madrid) y el valor máximo es 11.
- Lugar: contiene la calle o vía en la que se produce el accidente. Este campo puede contener el nombre de una única calle – por ejemplo, "CALLE DE TOLEDO NUM" – o el nombre de dos calles separados por un guion – por ejemplo, "CALLE DE PONZANO - CALLE DE RIOS ROSAS" – lo que indica que se trata de una intersección. En caso de no tratarse de una intersección, es decir, cuando

únicamente se informa del nombre de una calle, dicho nombre va a acompañado al final de la palabra “NUM” o “KM.“.

- Número: es un campo numérico e indica el número de la calle o el punto kilométrico de la vía en la que tiene lugar el accidente. Para saber si se trata de un número o de un punto kilométrico, hay que fijarse en la terminación del campo anterior, es decir, si el campo Lugar acaba con “NUM”, entonces el campo Número se corresponde con el número de la calle, pero si acaba con “KM.” se trata del punto kilométrico de la vía.
- Tipo de accidente: es un campo de texto. Se utiliza para clasificar los accidentes en: atropello, caída bicicleta, caída ciclomotor, caída motocicleta, caída vehículo de tres ruedas, caída viajero bus, choque con objeto fijo, colisión doble, colisión múltiple, vuelco, y otras causas.

El primer paso es convertir el fichero Excel con los datos sobre accidentes de tráfico en un formato que se pueda cargar en las herramientas que se van a utilizar. Para ello, se eliminan las pestañas dejando la información de todos los años en una sola. Los datos se formatean como CSV (Comma Separated Values – Valores Separados por Comas). Este formato tiene las siguientes características (Universidad Internacional de La Rioja, 2016):

- Cada registro se delimita por un cambio de línea.
- Los valores de cada registro se separan por comas. Es necesario que el número de valores sea constante para todos los registros disponibles en el fichero.
- Los valores pueden estar encapsulados con comillas dobles. Esto es obligatorio en aquellos casos en los que el valor incluye un cambio de línea, una coma o comillas dobles.
- Si un valor contiene comillas dobles, estas deben escaparse precediéndolas con otro carácter de comillas dobles.
- Opcionalmente, se puede incluir una primera línea con el nombre de cada campo.

El resultado de la transformación se puede apreciar en la *Figura 18*.

```
Fecha,RangoHorario,NumVictimas,Lugar,Numero,TipoAccidente
Domingo 01 Enero 2012,DE 2:00 A 2:59,1,AVENIDA DE SAN DIEGO NUM,96,OTRAS CAUSAS
Domingo 01 Enero 2012,DE 2:00 A 2:59,3,CALLE DE BRAVO MURILLO - CALLE DE LAS CAROLINAS,0,COLISIÓN DOBLE
Domingo 01 Enero 2012,DE 2:00 A 2:59,2,CALLE DE LA PRINCESA NUM,10,ATROPELLO
Domingo 01 Enero 2012,DE 5:00 A 5:59,1,CALLE DE ALCALA - AUTOVIA M-30 CALZADA 1,0,COLISIÓN MÚLTIPLE
Domingo 01 Enero 2012,DE 7:00 A 7:59,1,AVENIDA DE LOS POBLADOS - CALLE DEL PARQUE EUGENIA DE MONTIJO,0,CAÍDA VIAJERO BUS
Domingo 01 Enero 2012,DE 7:00 A 7:59,1,CALLE DE O'DONNELL NUM,53,ATROPELLO
Domingo 01 Enero 2012,DE 18:00 A 18:59,1,AVENIDA DE AMERICA - CALLE DE CARTAGENA,0,ATROPELLO
Domingo 01 Enero 2012,DE 18:00 A 18:59,2,CALLE DE JOAQUIN COSTA NUM,49,ATROPELLO
Domingo 01 Enero 2012,DE 19:00 A 19:59,1,CALLE DEL GENERAL RICARDOS NUM,33,COLISIÓN DOBLE
Domingo 01 Enero 2012,DE 20:00 A 20:59,1,CARRETERA DE BOADILLA DEL MONTE NUM,1,COLISIÓN DOBLE
Domingo 01 Enero 2012,DE 22:00 A 22:59,1,CALLE DE LA VILLA DE MARIN NUM,34,COLISIÓN DOBLE
Domingo 01 Enero 2012,DE 23:00 A 23:59,1,PASEO DE SANTA MARIA DE LA CABEZA - CALLE DE PALOS DE LA FRONTERA,0,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 4:00 A 4:59,1,CALLE DE BALEARES NUM,33,CHOQUE CON OBJETO FIJO
Lunes 02 Enero 2012,DE 10:00 A 10:59,1,AVENIDA DEL SANTUARIO DE VALVERDE NUM,99,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 10:00 A 10:59,1,PLAZA DE LA INDEPENDENCIA - CALLE DE ALFONSO XII,0,CAÍDA MOTOCICLETA
Lunes 02 Enero 2012,DE 12:00 A 12:59,1,CALLE DE LA ESFINGE - CALLE DEL DISCOBOLO,0,CHOQUE CON OBJETO FIJO
Lunes 02 Enero 2012,DE 13:00 A 13:59,2,AVENIDA DE LA PRINCESA JUANA DE AUSTRIA KM.,4700,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 13:00 A 13:59,1,PASEO DE LOS OLMOS NUM,13,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 13:00 A 13:59,1,PLAZA DE LA BEATA MARIA ANA DE JESUS NUM,6,CAÍDA CICLOMOTOR
Lunes 02 Enero 2012,DE 14:00 A 14:59,2,PLAZA DEL CONDE DE CASAL NUM,5,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 14:00 A 14:59,1,CALLE DEL GENERAL ALVAREZ DE CASTRO NUM,24,CAÍDA MOTOCICLETA
Lunes 02 Enero 2012,DE 16:00 A 16:59,1,CALLE DE EUGENIA DE MONTIJO - AVENIDA DE NUESTRA SEÑORA DE FATIMA,0,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 16:00 A 16:59,1,CALLE DE ALCALA - AVENIDA DE CANILLEJAS A VICALVARO,0,CHOQUE CON OBJETO FIJO
Lunes 02 Enero 2012,DE 16:00 A 16:59,1,CUESTA DEL SAGRADO CORAZON - CALLE 30 08RE,0,COLISIÓN MÚLTIPLE
Lunes 02 Enero 2012,DE 16:00 A 16:59,1,CALLE DEL GENERAL RICARDOS - CALLE DE RIO DE ORO,0,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 18:00 A 18:59,3,CALLE DEL PUERTO DE ARLABAN - CALLE DE LA SIERRA DE ALQUIFE,0,COLISIÓN MÚLTIPLE
Lunes 02 Enero 2012,DE 18:00 A 18:59,1,CALLE DEL PRINCIPE DE VERGARA - PLAZA DE LA REPUBLICA DEL ECUADOR,0,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 19:00 A 19:59,1,CALLE DE ARIADNA - AUTOVIA M-40,0,COLISIÓN MÚLTIPLE
Lunes 02 Enero 2012,DE 19:00 A 19:59,1,CALLE DE ALCALA NUM,728,COLISIÓN MÚLTIPLE
Lunes 02 Enero 2012,DE 20:00 A 20:59,1,AVENIDA DE LA ALBUFERA - CALLE DE BUSTOS,0,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 20:00 A 20:59,1,CALLE DEL ALTO DEL RETIRO NUM,33,COLISIÓN DOBLE
Lunes 02 Enero 2012,DE 22:00 A 22:59,1,CALLE DE SIERRA TOLEDANA NUM,41,CAÍDA MOTOCICLETA
Lunes 02 Enero 2012,DE 22:00 A 22:59,1,CALLE DE LA CONDESA DE VENADITO NUM,14,COLISIÓN DOBLE
Martes 03 Enero 2012,DE 00:00 A 00:59,1,CALLE DEL LAGO CONSTANZA NUM,22,COLISIÓN DOBLE
Martes 03 Enero 2012,DE 7:00 A 7:59,1,CALLE DE LA FUENTE CARRANTONA NUM,57,ATROPELLO
Martes 03 Enero 2012,DE 8:00 A 8:59,1,CALLE DEL PRINCIPE DE VERGARA - CALLE DE PADILLA,0,COLISIÓN DOBLE
Martes 03 Enero 2012,DE 8:00 A 8:59,1,CALLE DEL COCHERON DE LA VILLA NUM,2,COLISIÓN DOBLE
```

Figura 18. Extracto del fichero datos de accidentes de tráfico.

Los datos ya están preparados para ser manipulados. Para esta fase se utiliza Python que, tal y como se ha visto anteriormente, es un lenguaje de programación de alto nivel.

Para cargar el fichero en Python se importa el módulo “csv” con el comando “import”. Con el objetivo de manipular los datos se va a convertir el fichero csv en un “data frame”, estructura de datos con forma de matriz de elementos de diferentes tipos – ser podrían comparar con tablas. Para poder trabajar con los “data frames” es necesario importar el módulo “pandas”. El código que se muestra a continuación, es una función que, dada la ruta de un fichero csv, devuelve un “data frame” con los datos contenidos en este.

```
import csv
import pandas

def getDataFrameFromCsv(pathFile):
        with open(pathFile) as auxCsv:
                #Se indica que el delimitador es una coma y las
                #comillas dobles se usan para delimitar un campo
                #de texto
                dataCsv = csv.reader(auxCsv, delimiter=',', quotechar='"')
                #Se convierte en una lista
                dataList = list(dataCsv)
                #Se crea un data frame con dicha lista
                dataDF = pandas.DataFrame(dataList[1:],columns =
dataList[0])
                return dataDF
```

Ya están disponibles los datos en Python. Ahora, se puede ver qué calles son las que más accidentes tienen. Ejecutando la siguiente función, obtenemos el resultado de la *Figura 19*.

```
def streetNumAccidentsSorted():
        '''Count num of accidets by street and sorts them descending'''
        #Se obtiene el data frame a partir del CSV creado
        #con los datos de los accidentes de trafico
        accDF = getAccidentsDataFrame()

        #Lista con las calles o vías
        accList = accDF['Lugar']

        streetsCounter = {}

        #Para cada elemento de la lista de calles
        #Se cuenta el número de veces que aparece en
        #el data frame, que será igual al número de
        #accidentes que se dan en esa calle
        for acc in accList:
                if not streetsCounter.has_key(acc):
                        streetsCounter[acc] = 0
                streetsCounter[acc] += 1

        #Se ordena el resultado obtenido de mayor a menor
        sortedByValue = sorted(streetsCounter.items(),
key=op.itemgetter(1), reverse=True)
        return sortedByValue
```

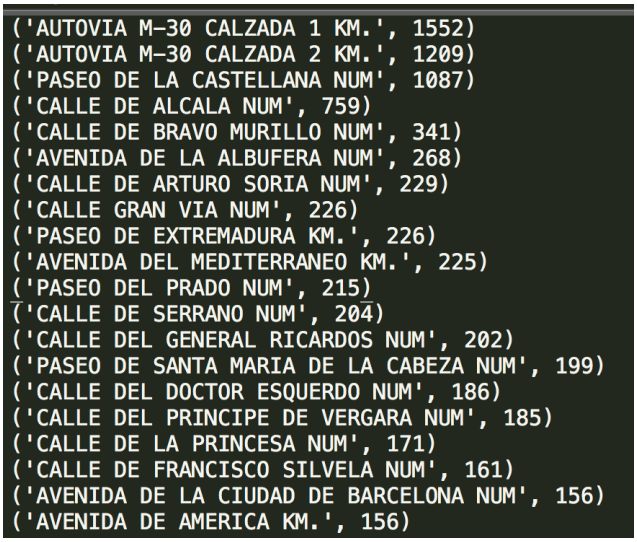


Figura 19. Top 20 calles de Madrid con más accidentes de tráfico

Como se puede observar, el output obtenido ofrece un listado ordenado en el que cada elemento consta del nombre de una calle y el número de accidentes que se han producido en esa calle desde enero de 2012 hasta junio de 2017. Se muestran únicamente los veinte primeros elementos de la lista. El resultado tiene sentido ya que la M-30 es la calle más concurrida y con más kilómetros de la ciudad de Madrid ya que se trata de una circunvalación. Le siguen en la lista el Paseo de La Castellana y la Calle Alcalá, dos de las calles más largas de la ciudad.

Teniendo en cuenta el resultado obtenido en el listado de la *Figura 19*, el estudio se va a centrar en las calles con más accidentes para acotar el alcance. Dado que las calles que vamos a tratar son largas, las vamos a dividir en cinco zonas cada una con el objetivo de determinar qué áreas son las más conflictivas. Para ello, se obtienen el máximo y el mínimo del campo “Número” para cada calle y se calcula a qué zona pertenece cada número. La siguiente función desarrollada en Python realiza la división de las calles en zonas y calcula la zona a la que pertenece cada uno:

```
def streetAccidentsDividedByZones():
        '''Split TOP 20 streets in 5 zones'''

        #Numero de zonas en las que se va a dividir cada calle
        numSplit = 5
        sortedByValue = streetNumAccidentsSorted()
        accDF = getAccidentsDataFrame()

        #Se reemplaza los valores NA por ceros para poder obtener
        #correctamente el maximo y el minimo del campo Numero
        accDF.Numero = pandas.to_numeric(accDF.Numero).fillna(0)

        #Se crea un data frame vacio con 3 columnas en el que
        #se va a almacenar la calle, su maximo y su minimo
        minMaxDF = pandas.DataFrame(columns=['Street', 'Min', 'Max'])
        #Para cada valle
        for street in sortedByValue:
                #Se obtiene una lista con el maximo y el minimo para la calle
dada
                minMax = minMaxStreet(street[0], accDF)
                #Se añade al resultado al data frame final
                minMaxDF.loc[len(minMaxDF.index)] = {'Street': street[0],
'Min':minMax['min'], 'Max':minMax['max']}

        #Se crea un nuevo data frame vacío con dos columnas
        streets = pandas.DataFrame(sortedByValue[0:20], columns=['Street',
'NumAcc'])
        #Se inicializa la columna nueva Zone a 0 en todos los elementos del
data frame original
        accDF['Zone'] = 0

        for i in range(0, len(accDF)):
                #Se busca el máximo el mínimo para una calle dada
                minMaxSubDF =
minMaxDF.loc[minMaxDF['Street'].isin(accDF.Lugar[i:i+1])]
                zone = 0
                #Si se ha encontrado la calle
                if len(minMaxSubDF) == 1:
                        num = int(accDF.Numero[i:i+1])
                        min = int(minMaxSubDF.Min[0:1])
                        max = int(minMaxSubDF.Max[0:1])

                        #Si la diferencia entre el maximo y el minimo es mayor
que el numero de zonas
                        if (max-min) > numSplit:
                                if num < max:
                                        zone = int(num / ((max-min)/numSplit))
+ 1
                                else:
```

```
                                        zone = numSplit
                        else:
                                zone = 1
                else:
                        zone = -1
                #Se modifica el valor de la columna Zone para ese
elemento de data frame
                accDF.Zone[i:i+1] = zone
        return accDF
```

El resultado obtenido es el “data frame” creado anteriormente a partir de fichero csv con una columna adicional llamada Zone en la que se almacena la zona de la calle a la que pertenece el número. Este “data frame” se guarda en un fichero CSV con las siguientes instrucciones:

```
#Se obtiene el data frame dividido por zonas
df = streetAccidentsDividedByZones()
#Utilizando la función to_csv del modulo pandas, se guarda el data
frame como fichero csv
#especificando la ruta donde se va a guardar el fichero y la
codificacion
df.to_csv(path_or_buf='/Users/luis/Documents/BigData/UNIR/Master/9.TFM
/Code/Output.csv', encoding='latin-1')
```

La *Figura 20* muestra un extracto del fichero CSV guardado con las instrucciones anteriores en la que se puede ver la nueva columna Zone.

```
Id,Fecha,RangoHorario,NumVictimas,Lugar,Numero,TipoAccidente,Zone
0,Domingo 01 Enero 2012,DE 2:00 A 2:59,1,AVENIDA DE SAN DIEGO NUM,96.0,OTRAS CAUSAS,4
1,Domingo 01 Enero 2012,DE 2:00 A 2:59,3,CALLE DE BRAVO MURILLO - CALLE DE LAS CAROLINAS,0.0,COLISIÓN DOBLE,1
2,Domingo 01 Enero 2012,DE 2:00 A 2:59,2,CALLE DE LA PRINCESA NUM,10.0,ATROPELLO,1
3,Domingo 01 Enero 2012,DE 5:00 A 5:59,1,CALLE DE ALCALA - AUTOVIA M-30 CALZADA 1,0.0,COLISIÓN MÚLTIPLE,1
4,Domingo 01 Enero 2012,DE 7:00 A 7:59,1,AVENIDA DE LOS POBLADOS - CALLE DEL PARQUE EUGENIA DE MONTIJO,0.0,CAÍDA VIAJERO BUS,1
5,Domingo 01 Enero 2012,DE 7:00 A 7:59,1,CALLE DE O'DONNELL NUM,53.0,ATROPELLO,4
6,Domingo 01 Enero 2012,DE 18:00 A 18:59,1,AVENIDA DE AMERICA - CALLE DE CARTAGENA,0.0,ATROPELLO,1
7,Domingo 01 Enero 2012,DE 18:00 A 18:59,2,CALLE DE JOAQUIN COSTA NUM,49.0,ATROPELLO,5
8,Domingo 01 Enero 2012,DE 19:00 A 19:59,1,CALLE DEL GENERAL RICARDOS NUM,33.0,COLISIÓN DOBLE,1
9,Domingo 01 Enero 2012,DE 20:00 A 20:59,1,CARRETERA DE BOADILLA DEL MONTE NUM,1.0,COLISIÓN DOBLE,1
10,Domingo 01 Enero 2012,DE 22:00 A 22:59,1,CALLE DE LA VILLA DE MARIN NUM,34.0,COLISIÓN DOBLE,5
11,Domingo 01 Enero 2012,DE 23:00 A 23:59,1,PASEO DE SANTA MARIA DE LA CABEZA - CALLE DE PALOS DE LA FRONTERA,0.0,COLISIÓN DOBLE,1
12,Lunes 02 Enero 2012,DE 4:00 A 4:59,1,CALLE DE BALEARES NUM,33.0,CHOQUE CON OBJETO FIJO,7
13,Lunes 02 Enero 2012,DE 10:00 A 10:59,1,AVENIDA DEL SANTUARIO DE VALVERDE NUM,99.0,COLISIÓN DOBLE,5
14,Lunes 02 Enero 2012,DE 10:00 A 10:59,1,PLAZA DE LA INDEPENDENCIA - CALLE DE ALFONSO XII,0.0,CAÍDA MOTOCICLETA,1
15,Lunes 02 Enero 2012,DE 12:00 A 12:59,1,CALLE DE LA ESFINGE - CALLE DEL DISCOBOLO,0.0,CHOQUE CON OBJETO FIJO,1
16,Lunes 02 Enero 2012,DE 13:00 A 13:59,2,AVENIDA DE LA PRINCESA JUANA DE AUSTRIA KM.,4700.0,COLISIÓN DOBLE,4
17,Lunes 02 Enero 2012,DE 13:00 A 13:59,1,PASEO DE LOS OLMOS NUM,13.0,COLISIÓN DOBLE,5
18,Lunes 02 Enero 2012,DE 13:00 A 13:59,1,PLAZA DE LA BEATA MARIA ANA DE JESUS NUM,6.0,CAÍDA CICLOMOTOR,4
19,Lunes 02 Enero 2012,DE 14:00 A 14:59,2,PLAZA DEL CONDE DE CASAL NUM,5.0,COLISIÓN DOBLE,6
20,Lunes 02 Enero 2012,DE 14:00 A 14:59,1,CALLE DEL GENERAL ALVAREZ DE CASTRO NUM,24.0,CAÍDA MOTOCICLETA,4
21,Lunes 02 Enero 2012,DE 16:00 A 16:59,1,CALLE DE EUGENIA DE MONTIJO - AVENIDA DE NUESTRA SEÑORA DE FATIMA,0.0,COLISIÓN DOBLE,1
22,Lunes 02 Enero 2012,DE 16:00 A 16:59,1,CALLE DE ALCALA - AVENIDA DE CANILLEJAS A VICALVARO,0.0,CHOQUE CON OBJETO FIJO,1
23,Lunes 02 Enero 2012,DE 16:00 A 16:59,1,CUESTA DEL SAGRADO CORAZON - CALLE 30 08RE,0.0,COLISIÓN MÚLTIPLE,1
24,Lunes 02 Enero 2012,DE 16:00 A 16:59,1,CALLE DEL GENERAL RICARDOS - CALLE DE RIO DE ORO,0.0,COLISIÓN DOBLE,1
25,Lunes 02 Enero 2012,DE 18:00 A 18:59,3,CALLE DEL PUERTO DE ARLABAN - CALLE DE LA SIERRA DE ALQUIFE,0.0,COLISIÓN MÚLTIPLE,1
26,Lunes 02 Enero 2012,DE 18:00 A 18:59,1,CALLE DEL PRINCIPE DE VERGARA - PLAZA DE LA REPUBLICA DEL ECUADOR,0.0,COLISIÓN DOBLE,1
27,Lunes 02 Enero 2012,DE 19:00 A 19:59,1,CALLE DE ARIADNA - AUTOVIA M-40,0.0,COLISIÓN MÚLTIPLE,1
28,Lunes 02 Enero 2012,DE 19:00 A 19:59,1,CALLE DE ALCALA NUM,728.0,COLISIÓN MÚLTIPLE,5
```

Figura 20. Extracto del fichero CSV con columna Zone

Como se ha podido observar anteriormente en el análisis realizado sobre los distintos campos que componen los datos de los accidentes de tráfico, el campo Fecha es un campo descriptivo del que se puede obtener información importante. No obstante, para poder utilizar dicha información, es necesario realizar alguna transformación. Se crean tres campos que se calculan a partir del campo Fecha: día de la semana (escrito), número del día de la semana, fecha con formato "DD/MM/YYYY". La función que se explica a continuación realiza dicha transformación:

```
def splitDateAndHour():
        #Se recupera el data frame guardado anteriormente con la zona
de cada calle
        accDF = getAccidentsZoneDataFrame()
        weekDay = []
        dates = []
        weekDayNum = []
        streetZone = []
        #Se crea un listado con equivalencias.
        #Dado el nombre de un mes, se le asigna el numero de mes
correspondiente
        monthEquivalence = {'Enero':'01',
'Febrero':'02','Marzo':'03','Abril':'04','Mayo':'05','Junio':'06','Jul
io':'07','Agosto':'08',

        'Septiembre':'09','Octubre':'10','Noviembre':'11',
'Diciembre':'12'}
        #Listado de equivalencias de dias de la semana
        weekNumEquivalence = {'Domingo':1,
'Lunes':2,'Martes':3,'Miercoles':4,'Jueves':5,'Viernes':6,'Sabado':7}
```

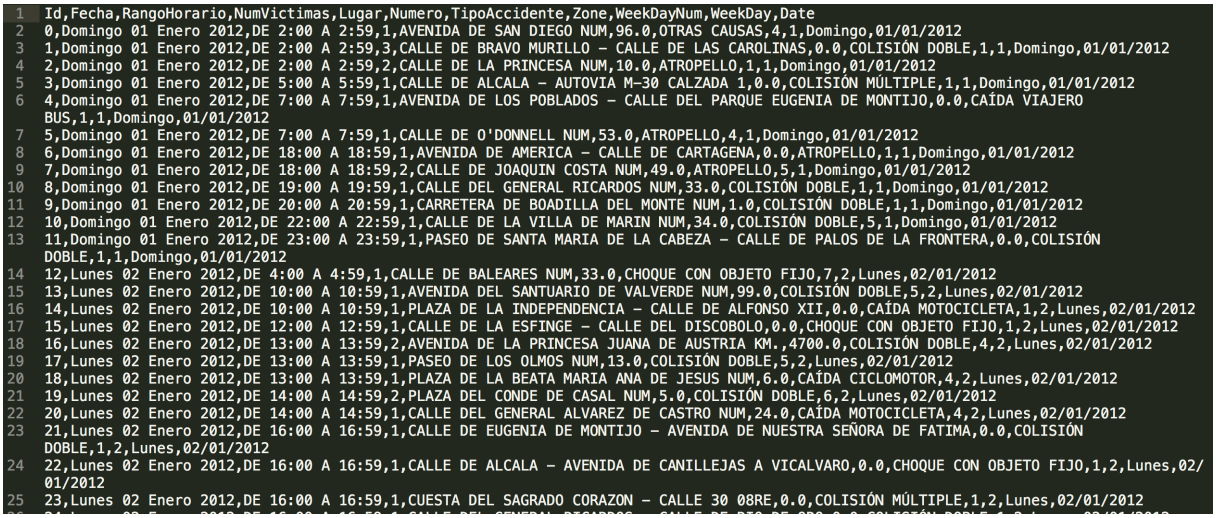
```
#Para cada fila del fata frame
        for i in range(0, len(accDF)):
                #------------- Dia de la semana
                fechaSplitted = str(accDF.Fecha[i:i+1]).split(' ')
                weekDay.append(fechaSplitted[4])
                #------------- Numero del dia de la semana

        weekDayNum.append(weekNumEquivalence[fechaSplitted[4].replace('
Mi\xef\xbf\xbdrcoles','Miercoles').
                                replace('S\xef\xbf\xbdbado','Sabado')])
                #------------- Fechas
                aux = fechaSplitted[7].split('\n')
                dates.append(str(fechaSplitted[5]) + "/" +
str(monthEquivalence[fechaSplitted[6]]) + "/" + str(aux[0]))
        #Se anaden al data frame origen los elementos calculados
        accDF['WeekDayNum'] = weekDayNum
        accDF['WeekDay'] = weekDay
        accDF['Date'] = dates
        #Listado de los campos que se van a devolver del data frame
        #A los ya existentes se han anadido numero del dia de la
semana,
        #dia de la semana escrito y fecha
        fields = ['Fecha', 'RangoHorario', 'NumVictimas', 'Lugar',
'Numero', 'TipoAccidente', 'Zone', 'WeekDayNum', 'WeekDay', 'Date',]
        return accDF[fields]
```

Dado que el volumen de datos es relativamente grande, estos procesos duran unos minutos y, con el fin de no repetirlos cada vez que se quiera realizar alguna modificación, se guarda cada “data frame” enriquecido como se hizo en el paso anterior:

```
#Se obtiene el data frame enriquecido con los datos de la fecha
df = splitDateAndHour()
#Se guarda como fichero CSV
df.to_csv(path_or_buf='/Users/luis/Documents/BigData/UNIR/Master/9.TFM
/Code/Output_Dates.csv')
```

El resultado obtenido se puede apreciar en la *Figura 21*, en la que se muestra un extracto del fichero CSV generado con los tres campos nuevos obtenidos del campo Fecha inicial:



```
Id,Fecha,RangoHorario,NumVictimas,Lugar,Numero,TipoAccidente,Zone,WeekDayNum,WeekDay,Date
0,Domingo 01 Enero 2012,DE 2:00 A 2:59,1,AVENIDA DE SAN DIEGO NUM,96.0,OTRAS CAUSAS,4,1,Domingo,01/01/2012
1,Domingo 01 Enero 2012,DE 2:00 A 2:59,3,CALLE DE BRAVO MURILLO - CALLE DE LAS CAROLINAS,0.0,COLISIÓN DOBLE,1,1,Domingo,01/01/2012
2,Domingo 01 Enero 2012,DE 2:00 A 2:59,2,CALLE DE LA PRINCESA NUM,10.0,ATROPELLO,1,1,Domingo,01/01/2012
3,Domingo 01 Enero 2012,DE 5:00 A 5:59,1,CALLE DE ALCALA - AUTOVIA M-30 CALZADA 1,0.0,COLISIÓN MÚLTIPLE,1,1,Domingo,01/01/2012
4,Domingo 01 Enero 2012,DE 7:00 A 7:59,1,AVENIDA DE LOS POBLADOS - CALLE DEL PARQUE EUGENIA DE MONTIJO,0.0,CAÍDA VIAJERO
BUS,1,1,Domingo,01/01/2012
5,Domingo 01 Enero 2012,DE 7:00 A 7:59,1,CALLE DE O'DONNELL NUM,53.0,ATROPELLO,4,1,Domingo,01/01/2012
6,Domingo 01 Enero 2012,DE 18:00 A 18:59,1,AVENIDA DE AMERICA - CALLE DE CARTAGENA,0.0,ATROPELLO,1,1,Domingo,01/01/2012
7,Domingo 01 Enero 2012,DE 18:00 A 18:59,2,CALLE DE JOAQUIN COSTA NUM,49.0,ATROPELLO,5,1,Domingo,01/01/2012
8,Domingo 01 Enero 2012,DE 19:00 A 19:59,1,CALLE DEL GENERAL RICARDOS NUM,33.0,COLISIÓN DOBLE,1,1,Domingo,01/01/2012
9,Domingo 01 Enero 2012,DE 20:00 A 20:59,1,CARRETERA DE BOADILLA DEL MONTE NUM,1.0,COLISIÓN DOBLE,1,1,Domingo,01/01/2012
10,Domingo 01 Enero 2012,DE 22:00 A 22:59,1,CALLE DE LA VILLA DE MARIN NUM,34.0,COLISIÓN DOBLE,5,1,Domingo,01/01/2012
11,Domingo 01 Enero 2012,DE 23:00 A 23:59,1,PASEO DE SANTA MARIA DE LA CABEZA - CALLE DE PALOS DE LA FRONTERA,0.0,COLISIÓN
DOBLE,1,1,Domingo,01/01/2012
12,Lunes 02 Enero 2012,DE 4:00 A 4:59,1,CALLE DE BALEARES NUM,33.0,CHOQUE CON OBJETO FIJO,7,2,Lunes,02/01/2012
13,Lunes 02 Enero 2012,DE 10:00 A 10:59,1,AVENIDA DEL SANTUARIO DE VALVERDE NUM,99.0,COLISIÓN DOBLE,5,2,Lunes,02/01/2012
14,Lunes 02 Enero 2012,DE 10:00 A 10:59,1,PLAZA DE LA INDEPENDENCIA - CALLE DE ALFONSO XII,0.0,CAÍDA MOTOCICLETA,1,2,Lunes,02/01/2012
15,Lunes 02 Enero 2012,DE 12:00 A 12:59,1,CALLE DE LA ESFINGE - CALLE DEL DISCOBOLO,0.0,CHOQUE CON OBJETO FIJO,1,2,Lunes,02/01/2012
16,Lunes 02 Enero 2012,DE 13:00 A 13:59,2,AVENIDA DE LA PRINCESA JUANA DE AUSTRIA KM.,4700.0,COLISIÓN DOBLE,4,2,Lunes,02/01/2012
17,Lunes 02 Enero 2012,DE 13:00 A 13:59,1,PASEO DE LOS OLMOS NUM,13.0,COLISIÓN DOBLE,5,2,Lunes,02/01/2012
18,Lunes 02 Enero 2012,DE 13:00 A 13:59,1,PLAZA DE LA BEATA MARIA ANA DE JESUS NUM,6.0,CAÍDA CICLOMOTOR,4,2,Lunes,02/01/2012
19,Lunes 02 Enero 2012,DE 14:00 A 14:59,2,PLAZA DEL CONDE DE CASAL NUM,5.0,COLISIÓN DOBLE,6,2,Lunes,02/01/2012
20,Lunes 02 Enero 2012,DE 14:00 A 14:59,1,CALLE DEL GENERAL ALVAREZ DE CASTRO NUM,24.0,CAÍDA MOTOCICLETA,4,2,Lunes,02/01/2012
21,Lunes 02 Enero 2012,DE 16:00 A 16:59,1,CALLE DE EUGENIA DE MONTIJO - AVENIDA DE NUESTRA SEÑORA DE FATIMA,0.0,COLISIÓN
DOBLE,1,2,Lunes,02/01/2012
22,Lunes 02 Enero 2012,DE 16:00 A 16:59,1,CALLE DE ALCALA - AVENIDA DE CANILLEJAS A VICALVARO,0.0,CHOQUE CON OBJETO FIJO,1,2,Lunes,02/
01/2012
23,Lunes 02 Enero 2012,DE 16:00 A 16:59,1,CUESTA DEL SAGRADO CORAZON - CALLE 30 08RE,0.0,COLISIÓN MÚLTIPLE,1,2,Lunes,02/01/2012
```

Figura 21. Extracto del fichero CSV con tres nuevas columnas.

Datos de festivos

Una vez estudiados y transformados los datos relativos a los accidentes de tráfico con víctimas ocurridos en Madrid desde enero de 2012 a junio de 2017, se procede a estudiar los datos sobre días festivos obtenidos del calendario laboral y del calendario escolar.

Tal y como se ha explicado anteriormente, se han creado sendos ficheros CSV con información sobre festivos del calendario laboral y del calendario escolar respectivamente. Como recordatorio, los ficheros estaban formados por seis campos: año, número, día, fecha completa con formato “DD/MM/YYYY”, número del día de la semana y día de la semana escrito. Cada fila se corresponde con un festivo, es decir, si un día del año no está presente en uno de los ficheros, significa que ese día no es festivo.

Estos datos se quieren utilizar para enriquecer los que ya se tienen sobre accidentes de tráfico. Para ello se van a cruzar por el campo Fecha, ya que tanto los ficheros de los festivos como el de los accidentes disponen del mismo formato. Se va a trabajar con variables “dummy” para poder aplicar regresiones sobre los datos. Las variables “dummy” son variables numéricas que representan subgrupos de una muestra (Trochim, 2006); actúan como interruptores de una ecuación. En este caso, si el día en el que ocurre el accidente es festivo se utiliza el valor 1, mientas que si no lo es se utiliza el valor 0.

Datos meteorológicos

Los datos meteorológicos son capturados del sitio web de la Agencia Estatal de Meteorología (Agencia Estatal de Meteorología, 2017). Los datos obtenidos se han almacenado en un fichero JSON. Dicho fichero contiene los siguientes campos: día en el que se recogen los datos, estación meteorológica que los recoge, precipitaciones, temperatura media del día, temperatura máxima del día, hora en la que se da esa

temperatura máxima, temperatura mínima, hora en la que se da la temperatura mínima, la racha máxima de viento, hora en la que se da la racha máxima, y velocidad media del viento durante el día.

A la hora de capturar los datos, el sitio web de la Agencia Estatal de Meteorología tiene ciertas limitaciones. Para rescatar datos históricos diarios, el formulario únicamente permite seleccionar un rango de fecha no superior a treinta y un días. Por este motivo, se han descargado sesenta y cinco ficheros JSON, uno por cada mes que se va a utilizar. Por tanto, el siguiente paso es cargar esos ficheros en un único "data frame" para así poder tratarlos en Python. La siguiente función realiza esta tarea, iterando sobre los ficheros y unificando todos los datos en un solo "data frame":

```
def getMeteorologyDataFrame():
        #Con la funcion read_json del modulo pandas cargamos el primer
fichero JSON
        df =
pandas.read_json('/Users/luis/Documents/BigData/UNIR/Master/9.TFM/Datasets/
Meteorologia/201201.json')
        #Se inicializa las variables year y month
        year = 2012
        month = 1
        #Para cada fichero
        for i in range(65):
                if month == 12 :
                        month = 1
                        year += 1
                else:
                        month += 1
                fileName = str(year * 100 + month)
                jsonData =
pandas.read_json('/Users/luis/Documents/BigData/UNIR/Master/9.TFM/Datasets/
Meteorologia/' + fileName + '.json')
                #Se crea una lista con el data frame que contiene la
informacion
                #de los ficheros cargados hasta el momento y el nuevo
fichero
                dfsToConcat = [df,jsonData]
                #Se unen los dos elementos de la lista
                df = pandas.concat(dfsToConcat)
        #Se rellenan los valores nulos con ceros
        df = df.fillna(0)
        #Se formatea la fecha con el patron "DD/MM/YYYY"
        df.fecha = df.fecha.apply(lambda date: date.split('-')[2] + '/' +
date.split('-')[1] + '/' + date.split('-')[0])
        return df
```

Transformación final

Con los datos ya analizados y tratados, es el momento de cruzarlos para obtener un único "data frame" con información de los accidentes de tráfico, festivos y meteorología. La siguiente función desarrollada en Python realiza esta tarea:

```
def addVacationToDF(trunc):
        #Datos de accidentes de trafico
        accDF = getAccidentsZoneDateDataFrame()
        #Datos de vacaciones obtenidos del calendario laboral
        vacation = getVacationDays()
        #Datos de vacaciones obtenidos del calendario escolar
        schoolVacation = getSchoolVacationDays()
        #Datos meteorologicos
        meteorology = getMeteorologyDataFrame()
        #Esta funcion calcula los valores mínimo y máximo de cada campo
        #para la informacion meteorolgica.
        meteorologyMinMax = minMaxMeteorology(meteorology)

        #Lista en la que se almacenan los datos de las vacaciones
laborales
        vactionList = []
        #Lista en la que se almacenan los datos de las vacaciones
escolares
        schoolVacList = []
        #Lista en la que se almacenan las horas normalizadas/eslacadas
        normalizedHours = []
        #Lista en la que se almacenan los datos de las precipitaciones
        normalizedPrec = []
        #Lista en la que se almacenan los datos de las rachas de viento
        normalizedRacha = []
        #Lista en la que se almacenan los datos de la temperatura
maxima
        normalizedTMax = []
        #Lista en la que se almacenan los datos de la temperatura media
        normalizedTMed = []
        #Lista en la que se almacenan los datos de la temperatura
minima
        normalizedTMin = []
        #Lista en la que se almacenan los datos de la velocidad media
        normalizedVelmedia = []

        #Se reemplaza las comas por puntos para cada columna
        meteorology.prec = meteorology.prec.apply(lambda x:
str(x).replace(',','.')).astype(float)
        meteorology.racha = meteorology.racha.apply(lambda x:
str(x).replace(',','.')).astype(float)
        meteorology.tmax = meteorology.tmax.apply(lambda x:
str(x).replace(',','.')).astype(float)
        meteorology.tmed = meteorology.tmed.apply(lambda x:
str(x).replace(',','.')).astype(float)
        meteorology.tmin = meteorology.tmin.apply(lambda x:
str(x).replace(',','.')).astype(float)
        meteorology.velmedia = meteorology.velmedia.apply(lambda x:
str(x).replace(',','.')).astype(float)
```

```
#----------Vacaciones y datos meteorologicos
        #Para cada fecha presente en el data frame de accidentes de
trafico
        for date in accDF.Date:
                #Contiene un data frame de vacaciones laborales
filtrado por la fecha
                aux = vacation.loc[vacation.Date == date]
                #Si el data frame contiene algún elemento implica que
ese dia es festivo laboral
                if len(aux) > 0:
                        vactionList.append(1)
                else:
                        vactionList.append(0)
                #Contiene un data frame de vacaciones escolares
filtrado por la fecha
                aux = schoolVacation.loc[schoolVacation.Date == date]
                #Si el data frame contiene algún elemento implica que
ese dia es festivo escolar
                if len(aux) > 0:
                        schoolVacList.append(1)
                else:
                        schoolVacList.append(0)
                #Data frame meteorologico filtrado por la fecha
                aux = meteorology.loc[meteorology.fecha == date]
                if len(aux) > 0:
                        #Se resetan los indices del data frame
                        aux = aux.reset_index(drop=True)
                        #-------------Prec --> se escalan los valores
                        aux2 =
meteorologyMinMax.loc[meteorologyMinMax.Item == 'prec']
                        aux2 = aux2.reset_index(drop=True)
                        normalizedPrec.append(float((aux.prec -
aux2.Min)/(aux2.Max-aux2.Min)))
                        #-------------Racha --> se escalan los valores
                        aux2 =
meteorologyMinMax.loc[meteorologyMinMax.Item == 'racha']
                        aux2 = aux2.reset_index(drop=True)
                        normalizedRacha.append(float((aux.racha -
aux2.Min)/(aux2.Max-aux2.Min)))
                        #-------------Tmax --> se escalan los valores
                        aux2 =
meteorologyMinMax.loc[meteorologyMinMax.Item == 'tmax']
                        aux2 = aux2.reset_index(drop=True)
                        normalizedTMax.append(float((aux.tmax -
aux2.Min)/(aux2.Max-aux2.Min)))
```

```
#-------------Tmed --> se escalan los valores
                        aux2 =
meteorologyMinMax.loc[meteorologyMinMax.Item == 'tmed']
                        aux2 = aux2.reset_index(drop=True)
                        normalizedTMed.append(float((aux.tmed -
aux2.Min)/(aux2.Max-aux2.Min)))
                        #-------------Tmin --> se escalan los valores
                        aux2 =
meteorologyMinMax.loc[meteorologyMinMax.Item == 'tmin']
                        aux2 = aux2.reset_index(drop=True)
                        normalizedTMin.append(float((aux.tmin -
aux2.Min)/(aux2.Max-aux2.Min)))
                        #-------------Velmed --> se escalan los valores
                        aux2 =
meteorologyMinMax.loc[meteorologyMinMax.Item == 'velmedia']
                        aux2 = aux2.reset_index(drop=True)
                        normalizedVelmedia.append(float((aux.velmedia -
aux2.Min)/(aux2.Max-aux2.Min)))
                else:
                        normalizedPrec.append(0)
                        normalizedRacha.append(0)
                        normalizedTMax.append(0)
                        normalizedTMed.append(0)
                        normalizedTMin.append(0)
                        normalizedVelmedia.append(0)
        #-----------Se escala el campo Rango Horario
        for hour in accDF.RangoHorario:
                aux = str(hour).split(' ')
                aux = aux[1].split(':')
                normalizedHours.append(float(aux[0])/24)
        #-----------Se concatenan el nombre de la calle y la zona
        lugares = accDF.Lugar
        zones = accDF.Zone
        lugZone = zip(lugares,zones)
        result = []
        for lz in lugZone:
                result.append(lz[0] + '_' + lz[1])
        #Se crea un nuevo campo Quarter que indica el trimestre
        #al que pertenece la fecha
        accDF['Quarter'] = accDF['Date'].apply(lambda date:
                float(1) / 4 if int(date.split('/')[1]) <= 3 else
                float(2) / 4 if int(date.split('/')[1]) <= 6 else
                float(3) / 4 if int(date.split('/')[1]) <= 9 else
                float(4) / 4)
```

```
#Se asignan los valores a los nuevos campos del data frame principal
        accDF['Vacation'] = vactionList
        accDF['VacationSchool'] = schoolVacList
        accDF['NormHour'] = normalizedHours
        accDF['Prec'] = normalizedPrec
        accDF['Racha'] = normalizedRacha
        accDF['TMax'] = normalizedTMax
        accDF['TMed'] = normalizedTMed
        accDF['TMin'] = normalizedTMin
        accDF['VelMedia'] = normalizedVelmedia
        accDF['Result'] = result
        accDF['WeekDayNum'] = accDF['WeekDayNum'].apply(lambda day:
float(day)/7)

        #Si se quieren truncar los valores
        if trunc==True:
                accDF['Prec'] = accDF['Prec'].apply(lambda x:
'%.1f'%(x))
                accDF['Racha'] = accDF['Racha'].apply(lambda x:
'%.1f'%(x))
                accDF['TMax'] = accDF['TMax'].apply(lambda x:
'%.1f'%(x))
                accDF['TMed'] = accDF['TMed'].apply(lambda x:
'%.1f'%(x))
                accDF['TMin'] = accDF['TMin'].apply(lambda x:
'%.1f'%(x))
                accDF['VelMedia'] = accDF['VelMedia'].apply(lambda x:
'%.1f'%(x))
                accDF['NormHour'] = accDF['NormHour'].apply(lambda x:
'%.1f'%(x))


        #Se seleccionan los campos del data frame que se quieren
devolver
        fields =
['RangoHorario','Lugar','TipoAccidente','Zone','WeekDayNum','WeekDay',
'Date','Vacation','VacationSchool','NormHour',

        'Quarter','Prec','Racha','TMax','TMed','TMin','VelMedia','Resul
t']
        return accDF[fields]
```

En primer lugar, se recuperan los datos de accidentes, vacaciones laborales, vacaciones escolares y datos meteorológicos y se almacenan en diferentes “data frames”. A continuación, se formatean correctamente lo datos meteorológicos, sustituyendo las comas por puntos para que cuando se vayan a escalar los valores no se produzcan problemas con

los decimales. Después, para cada fecha del fichero de accidentes se comprueba si es día festivo laboral y festivo escolar. Los datos meteorológicos se recuperan para la fecha dada y se escalan en función de sus valores mínimo y máximo obteniendo un número comprendido entre 0 y 1. También se escalan los valores del campo Rango Horario y se concatenan el nombre de la calle y la zona, valor que se almacena en un nuevo campo llamado "Result". Adicionalmente, se crea un nuevo campo llamado "Quarter" que almacena el trimestre del año al que pertenece la fecha.

La función anterior genera un "data frame" que se guarda como fichero CSV con las siguientes instrucciones:

```
df = addVacationToDF(True)
df.to_csv(path_or_buf='/Users/luis/Documents/BigData/UNIR/Master/9.TFM
/Code/Output_Vacation.csv')
```

La Figura 22 muestra un extracto del fichero CSV generado con la información de accidentes de tráfico enriquecida con datos meteorológicos y de vacaciones.

```
Id,RangoHorario,Lugar,TipoAccidente,Zone,WeekDayNum,WeekDay,Date,Vacation,VacationSchool,NormHour,Quarter,Prec,Racha,TMax,TMed,TMin,VelMedia,Result
0,DE 2:00 A 2:59,AVENIDA DE SAN DIEGO NUM,OTRAS CAUSAS,4,0.142857142857,Domingo,01/01/2012,0,0,0.1,0.25,0.0,0.2,0.3,0.2,0.2,0.1,AVENIDA DE SAN DIEGO
NUM_4
1,DE 2:00 A 2:59,CALLE DE BRAVO MURILLO - CALLE DE LAS CAROLINAS,COLISIÓN DOBLE,1,0.142857142857,Domingo,01/
01/2012,0,0,0.1,0.25,0.0,0.2,0.3,0.2,0.2,0.1,CALLE DE BRAVO MURILLO - CALLE DE LAS CAROLINAS_1
2,DE 2:00 A 2:59,CALLE DE LA PRINCESA NUM,ATROPELLO,1,0.142857142857,Domingo,01/01/2012,0,0,0.1,0.25,0.0,0.2,0.3,0.2,0.2,0.1,CALLE DE LA PRINCESA NUM_1
3,DE 5:00 A 5:59,CALLE DE ALCALA - AUTOVIA M-30 CALZADA 1,COLISIÓN MÚLTIPLE,1,0.142857142857,Domingo,01/
01/2012,0,0,0.2,0.25,0.0,0.2,0.3,0.2,0.2,0.1,CALLE DE ALCALA - AUTOVIA M-30 CALZADA 1_1
4,DE 7:00 A 7:59,AVENIDA DE LOS POBLADOS - CALLE DEL PARQUE EUGENIA DE MONTIJO,CAÍDA VIAJERO BUS,1,0.142857142857,Domingo,01/
01/2012,0,0,0.3,0.25,0.0,0.2,0.3,0.2,0.2,0.1,AVENIDA DE LOS POBLADOS - CALLE DEL PARQUE EUGENIA DE MONTIJO_1
5,DE 7:00 A 7:59,CALLE DE O'DONNELL NUM,ATROPELLO,4,0.142857142857,Domingo,01/01/2012,0,0,0.3,0.25,0.0,0.2,0.3,0.2,0.2,0.1,CALLE DE O'DONNELL NUM_4
6,DE 18:00 A 18:59,AVENIDA DE AMERICA - CALLE DE CARTAGENA,ATROPELLO,1,0.142857142857,Domingo,01/01/2012,0,0,0.8,0.25,0.0,0.2,0.3,0.2,0.2,0.1,AVENIDA
DE AMERICA - CALLE DE CARTAGENA_1
7,DE 18:00 A 18:59,CALLE DE JOAQUIN COSTA NUM,ATROPELLO,5,0.142857142857,Domingo,01/01/2012,0,0,0.8,0.25,0.0,0.2,0.3,0.2,0.2,0.1,CALLE DE JOAQUIN COSTA
NUM_5
8,DE 19:00 A 19:59,CALLE DEL GENERAL RICARDOS NUM,COLISIÓN DOBLE,1,0.142857142857,Domingo,01/01/2012,0,0,0.8,0.25,0.0,0.2,0.3,0.2,0.2,0.1,CALLE DEL
GENERAL RICARDOS NUM_1
9,DE 20:00 A 20:59,CARRETERA DE BOADILLA DEL MONTE NUM,COLISIÓN DOBLE,1,0.142857142857,Domingo,01/
01/2012,0,0,0.8,0.25,0.0,0.2,0.3,0.2,0.2,0.1,CARRETERA DE BOADILLA DEL MONTE NUM_1
10,DE 22:00 A 22:59,CALLE DE LA VILLA DE MARIN NUM,COLISIÓN DOBLE,5,0.142857142857,Domingo,01/01/2012,0,0,0.9,0.25,0.0,0.2,0.3,0.2,0.2,0.1,CALLE DE LA
VILLA DE MARIN NUM_5
11,DE 23:00 A 23:59,PASEO DE SANTA MARIA DE LA CABEZA - CALLE DE PALOS DE LA FRONTERA,COLISIÓN DOBLE,1,0.142857142857,Domingo,01/
01/2012,0,0,1.0,0.25,0.0,0.2,0.3,0.2,0.2,0.1,PASEO DE SANTA MARIA DE LA CABEZA - CALLE DE PALOS DE LA FRONTERA_1
12,DE 4:00 A 4:59,CALLE DE BALEARES NUM,CHOQUE CON OBJETO FIJO,7,0.285714285714,Lunes,02/01/2012,0,1,0.2,0.25,0.0,0.7,0.3,0.3,0.4,0.6,CALLE DE BALEARES
NUM_7
13,DE 10:00 A 10:59,AVENIDA DEL SANTUARIO DE VALVERDE NUM,COLISIÓN DOBLE,5,0.285714285714,Lunes,02/01/2012,0,1,0.4,0.25,0.0,0.7,0.3,0.3,0.4,0.6,AVENIDA
DEL SANTUARIO DE VALVERDE NUM_5
14,DE 10:00 A 10:59,PLAZA DE LA INDEPENDENCIA - CALLE DE ALFONSO XII,CAÍDA MOTOCICLETA,1,0.285714285714,Lunes,02/
01/2012,0,1,0.4,0.25,0.0,0.7,0.3,0.3,0.4,0.6,PLAZA DE LA INDEPENDENCIA - CALLE DE ALFONSO XII_1
15,DE 12:00 A 12:59,CALLE DE LA ESFINGE - CALLE DEL DISCOBOLO,CHOQUE CON OBJETO FIJO,1,0.285714285714,Lunes,02/
01/2012,0,1,0.5,0.25,0.0,0.7,0.3,0.3,0.4,0.6,CALLE DE LA ESFINGE - CALLE DEL DISCOBOLO_1
16,DE 13:00 A 13:59,AVENIDA DE LA PRINCESA JUANA DE AUSTRIA KM.,COLISIÓN DOBLE,4,0.285714285714,Lunes,02/
01/2012,0,1,0.5,0.25,0.0,0.7,0.3,0.3,0.4,0.6,AVENIDA DE LA PRINCESA JUANA DE AUSTRIA KM._4
17,DE 13:00 A 13:59,PASEO DE LOS OLMOS NUM,COLISIÓN DOBLE,5,0.285714285714,Lunes,02/01/2012,0,1,0.5,0.25,0.0,0.7,0.3,0.3,0.4,0.6,PASEO DE LOS OLMOS
NUM_5
18,DE 13:00 A 13:59,PLAZA DE LA BEATA MARIA ANA DE JESUS NUM,CAÍDA CICLOMOTOR,4,0.285714285714,Lunes,02/
```

Figura 22. Extracto del fichero CSV con todos los datos.

En el fichero CSV que se ha obtenido a partir de las transformaciones anteriores existen determinados campos útiles para construir el modelo y otros que son meramente informativos. Además, hasta ahora se tiene una fila por cada accidente producido, con toda la información enriquecida, pero es necesario realizar algunas modificaciones para poder aplicar regresiones y redes neuronales. Un primer paso es agrupar los registros del fichero por los campos que se van a utilizar para determinar si se produce un accidente o no. Estos

campos son los siguientes: WeekDayNum, Vacation, NormHour, Quarter, Prec, TMed, VelMedia. La función utilizada para realizar la agrupación es la siguiente:

```
def groupDFByVariables():
        #Se obtiene el dataframe guardado previamente
        #con datos de accidentes enriquecidos con vacaciones
        #y meteorologia
        accDF = getAccidentsZoneDateVacDataFrame()
        #Lista de campos por los que se realiza la agrupacion
        header
=['WeekDayNum','Vacation','NormHour','Quarter','Prec','TMed','VelMedia
']
        #Data frame donde se almacena la agrupacion
        groupDF = pandas.DataFrame(columns=header)
        auxDF = pandas.DataFrame()
        #Para cada registro
        for i in range(len(accDF)):
                lenGroup = len(groupDF)
                lenAux = 0
                if lenGroup > 0:
                        #Se obtiene los registros que cumplan con el
filtro aplicado
                        auxDF = groupDF.loc[
                                (groupDF['WeekDayNum'] ==
accDF.loc[i,'WeekDayNum'])
                                &(groupDF['Vacation'] ==
accDF.loc[i,'Vacation'])
                                &(groupDF['NormHour'] ==
accDF.loc[i,'NormHour'])
                                & (groupDF['Quarter'] ==
accDF.loc[i,'Quarter'])
                                & (groupDF['Prec'] ==
accDF.loc[i,'Prec'])
                                & (groupDF['TMed'] ==
accDF.loc[i,'TMed'])
                                & (groupDF['VelMedia'] ==
accDF.loc[i,'VelMedia'])
                                ]
                        lenAux = len(auxDF)
                if lenAux == 0:
```

```
#Si no existe dicha combinacion, se crea en el nuevo data frame
                        groupDF.loc[lenGroup,'WeekDayNum'] =
accDF.loc[i,'WeekDayNum']
                        groupDF.loc[lenGroup,'Vacation'] =
accDF.loc[i,'Vacation']
                        groupDF.loc[lenGroup,'NormHour'] =
accDF.loc[i,'NormHour']
                        groupDF.loc[lenGroup,'Quarter'] =
accDF.loc[i,'Quarter']
                        groupDF.loc[lenGroup,'Prec'] =
accDF.loc[i,'Prec']
                        groupDF.loc[lenGroup,'TMed'] =
accDF.loc[i,'TMed']
                        groupDF.loc[lenGroup,'VelMedia'] =
accDF.loc[i,'VelMedia']
        return groupDF
```

El resultado obtenido es un “data frame” que contiene todas las combinaciones posibles de los campos vistos anteriormente que contiene el fichero CSV con los datos de los accidentes de tráfico enriquecidos con vacaciones y datos meteorológicos.

Como se indicó al comienzo de esta sección, el estudio se va a centrar en aquellas zonas que más accidentes acumulen. Se va a comenzar con “AUTOVIA M-30 CALZADA 1 KM._2”, es decir, la zona 2 de la calle Autovía M-30 Calzada 1. Para esto, se va a recorrer cada fila del “data frame” generado con las variables agrupadas y se va a buscar esta combinación de variables en el “data frame” de accidentes, añadiendo al filtro aplicado que el campo Result sea igual a “AUTOVIA M-30 CALZADA 1 KM._2”. El número de resultados positivos se dividirá entre el número registros que se obtengan de filtrar el “data frame” de los accidentes con la combinación de variables anteriores (sin filtrar el campo Result). De esta manera, el resultado obtenido es la probabilidad de ocurrencia del accidente para esa combinación de variables. La siguiente función realiza esta tarea:

```
def getProbabilityDF(streetZoneName):
        #Se obtiene el data frame de las variables agrupadas
        groupVariables = getVariablesGroupDataFrame()
        #Se obtiene el data frame de los accidentes enriquecidos
        accDF = getAccidentsZoneDateVacDataFrame()
        #Lista donde se almacenan las probabilidades
        probability = []
        #Sub data frame con los registros que coinciden
        #con la combinación de variables
        totalCases = pandas.DataFrame()
        #Sub data frame con los registros que coinciden
        #con la combinación de variables y con la zona
        trueCases = pandas.DataFrame()
        #Para cada combinacion de variables
        for i in range(len(groupVariables)):
                #Filtro sin la calle y zona
                totalCases = accDF.loc[
                        (accDF['WeekDayNum'] ==
groupVariables.loc[i,'WeekDayNum'])
                        & (accDF['Vacation'] ==
groupVariables.loc[i,'Vacation'])
                        & (accDF['NormHour'] ==
groupVariables.loc[i,'NormHour'])
                        & (accDF['Quarter'] ==
groupVariables.loc[i,'Quarter'])
                        & (accDF['Prec'] ==
groupVariables.loc[i,'Prec'])
                        & (accDF['TMed'] ==
groupVariables.loc[i,'TMed'])
                        &(accDF['NormHour'] ==
groupVariables.loc[i,'NormHour'])
                        & (accDF['VelMedia'] ==
groupVariables.loc[i,'VelMedia'])]
```

```
#Filtro con la calle y zona
                trueCases = accDF.loc[
                          (accDF['WeekDayNum'] ==
groupVariables.loc[i,'WeekDayNum'])
                        & (accDF['Vacation'] ==
groupVariables.loc[i,'Vacation'])
                        & (accDF['NormHour'] ==
groupVariables.loc[i,'NormHour'])
                        & (accDF['Quarter'] ==
groupVariables.loc[i,'Quarter'])
                        & (accDF['Prec'] ==
groupVariables.loc[i,'Prec'])
                        & (accDF['TMed'] ==
groupVariables.loc[i,'TMed'])
                        &(accDF['NormHour'] ==
groupVariables.loc[i,'NormHour'])
                        & (accDF['VelMedia'] ==
groupVariables.loc[i,'VelMedia'])
                        & (accDF['Result'] == streetZoneName)]

        probability.append(float(len(trueCases))/float(len(totalCases))
)

        #Se crea un nuevo campo con la probabilidad de que se produzca
un accidente
        groupVariables['Probability'] = probability
        return groupVariables
```

La *Figura 23* muestra un extracto del “data frame” guardado como fichero CSV con la probabilidad de ocurrencia de un accidente en la zona “AUTOVIA M-30 CALZADA 1 KM._2”.

```
Id2,Id1,WeekDayNum,Vacation,NormHour,Quarter,Prec,TMed,VelMedia,Probability
0,0,0.142857142857,0,0.1,0.25,0.0,0.2,0.1,0.0
1,1,0.142857142857,0,0.2,0.25,0.0,0.2,0.1,0.0
2,2,0.142857142857,0,0.3,0.25,0.0,0.2,0.1,0.0
3,3,0.142857142857,0,0.8,0.25,0.0,0.2,0.1,0.0
4,4,0.142857142857,0,0.9,0.25,0.0,0.2,0.1,0.0
5,5,0.142857142857,0,1.0,0.25,0.0,0.2,0.1,0.0
6,6,0.285714285714,0,0.2,0.25,0.0,0.3,0.6,0.0
7,7,0.285714285714,0,0.4,0.25,0.0,0.3,0.6,0.0
8,8,0.285714285714,0,0.5,0.25,0.0,0.3,0.6,0.0
9,9,0.285714285714,0,0.6,0.25,0.0,0.3,0.6,0.0
10,10,0.285714285714,0,0.7,0.25,0.0,0.3,0.6,0.0
11,11,0.285714285714,0,0.8,0.25,0.0,0.3,0.6,0.0
12,12,0.285714285714,0,0.9,0.25,0.0,0.3,0.6,0.0
13,13,0.428571428571,0,0.0,0.25,0.0,0.2,0.1,0.0
14,14,0.428571428571,0,0.3,0.25,0.0,0.2,0.1,0.0
15,15,0.428571428571,0,0.4,0.25,0.0,0.2,0.1,0.0
16,16,0.428571428571,0,0.5,0.25,0.0,0.2,0.1,0.0
17,17,0.428571428571,0,0.6,0.25,0.0,0.2,0.1,0.0
18,18,0.428571428571,0,0.7,0.25,0.0,0.2,0.1,0.0
19,19,0.428571428571,0,0.8,0.25,0.0,0.2,0.1,0.0
20,20,0.428571428571,0,0.9,0.25,0.0,0.2,0.1,0.0
21,21,0.571428571429,0,0.0,0.25,0.0,0.0,0.1,0.0
22,22,0.571428571429,0,0.4,0.25,0.0,0.0,0.1,0.0
23,23,0.571428571429,0,0.5,0.25,0.0,0.0,0.1,0.0
24,24,0.571428571429,0,0.6,0.25,0.0,0.0,0.1,0.0
25,25,0.571428571429,0,0.7,0.25,0.0,0.0,0.1,0.0
26,26,0.571428571429,0,0.8,0.25,0.0,0.0,0.1,0.0
27,27,0.571428571429,0,0.9,0.25,0.0,0.0,0.1,0.0
28,28,0.571428571429,0,1.0,0.25,0.0,0.0,0.1,0.0
29,29,0.714285714286,0,0.0,0.25,0.0,0.2,0.1,0.0
30,30,0.714285714286,0,0.3,0.25,0.0,0.2,0.1,0.0526315789474
31,31,0.714285714286,0,0.4,0.25,0.0,0.2,0.1,0.0
32,32,0.714285714286,0,0.5,0.25,0.0,0.2,0.1,0.0384615384615
33,33,0.714285714286,0,0.6,0.25,0.0,0.2,0.1,0.0
```

Figura 23. Extracto fichero CSV - probabilidad accidente en la M-30 Calzada 1, zona 2

Por último, se procede a dividir el fichero en dos, uno de ellos contiene lo datos que se van a utilizar para realizar el entrenamiento de la red neuronal y el otro se utilizará para realizar el test y comprobar el resultado obtenido. Para ello, se utiliza la siguiente función:

```
def splitDFTrainTest(filePath):
        '''Split DF into two: 3/4 --> Train, 1/4 --> Test'''

        #Se obtiene el fichero CSV que se desea separar
        dfAux = getDataFrameFromCsv(filePath)
        #       'TMax', 'TMed','TMin','VelMedia', 'Result']
        fields =
['WeekDayNum','Vacation','NormHour','Quarter','Prec','TMed','VelMedia'
,'Probability']
        #fields = ['NormHour','Prec','Probability']
        df = pandas.DataFrame(columns = fields)
        df = dfAux[fields]

        train = pandas.DataFrame(columns=fields)
        test = pandas.DataFrame(columns=fields)

        #Para cada registro fila del data frame se genera un numero
aleatorio
        #entre 1 y 4, de tal manera que si el numero obtenido toma
        #lo valores 1, 2 o 3, entonces la fila formara parte
        #del data frame de entrenamiento, mientras que si toma
        #valor 4, formara parte del data frame de test
        for i in range(0,len(df)):
                #Generate a random number between 1 and 4
                rdNum = int(numpy.random.uniform(0,4)) + 1
                #If 1,2, or 3 --> Training DF; If 4 --> Validation DF
                if rdNum < 4:
                        train.loc[len(train)] = df.loc[i].values
                else:
                        test.loc[len(test)] = df.loc[i].values

        result = {'Train': train, 'Test': test}
        return result
```

4.4. Aplicando regresiones

Como ya se ha explicado en las secciones anteriores, el proceso llevado hasta el momento ha consistido en capturar los datos, estudiarlos y transformarlos para adaptarlos a los requerimientos.

Con los datos ya disponibles en el formato deseado, en esta sección se van a aplicar regresiones lineales con el objetivo de poder predecir la probabilidad de ocurrencia de un accidente en una calle y zona determinadas.

Para este parte del trabajo se utiliza RStudio, que tal y como se comentó en secciones anteriores, es un IDE para trabajar con R.

Una regresión lineal estudia la ecuación lineal que describe mejor la asociación entre variables (Universidad Internacional de La Rioja, 2016). Conocer la ecuación de regresión permite calcular predicciones de una variable a partir de los valores que tomen otra/s variable/s. El método de los mínimos cuadrados (MMC) es el procedimiento matemático que permite calcular la ecuación de la regresión y sus componentes.

Tal y como se explica en el libro *El paquete estadístico R* (Freijo, 2013), en R se utiliza la función *lm* para realizar regresiones lineales. Primero se carga el fichero CSV con las probabilidades de que ocurran accidentes en la M-30 Calzada 1, zona 2. A continuación, con la función *lm*, se realiza la regresión lineal utilizando la variable Probability como variable dependiente y las variables WeekDayNum, Vacation, NormHour, Quarter, Prec, TMed y VelMedia como predictoras. El script utilizado para realizar estas acciones es el siguiente:

```
#Se carga el fichero CSV con los datos que se ven a utilizar para el
test
csvTest =
read.csv('/Users/luis/Documents/BigData/UNIR/Master/9.TFM/Datasets/Acc
identes Desagregado/Validation_M30_Calz1_Zone2.csv')
#Se carga el fichero CSV con los datos que se ven a utilizar para el
entrenamiento/regresion
csvTrain =
read.csv('/Users/luis/Documents/BigData/UNIR/Master/9.TFM/Datasets/Acc
identes Desagregado/Train_M30_Calz1_Zone2.csv')
#Se estima el modelo realizando una regresion lineal
#La variable dependiente es Probability y las variables que siguen al
simbolo ~ son las variables predictoras
#En el segundo parametro se indica la variable que almacena los datos
regresion <-
lm(Probability~WeekDayNum+Vacation+NormHour+Quarter+Prec+TMed+VelMedia
, data = csvTest)
#Se visualiza el resultado de la regresion
print(summary(regresion))
```

La última instrucción imprime por consola el resultado de la regresión (ver *Figura 24*). Con los resultados obtenidos se puede apreciar que el valor R^2 es muy bajo (0.002646), lo que parece indicar que el ajuste del modelo no es bueno. Esta teoría toma más peso cuando se aprecia que la mayoría de los parámetros correspondientes a las variables predictoras no son significativos, como lo prueba el bajo valor que toma el estadístico t de Student para estos (columna *t value*).

```
Call:
lm(formula = Probability ~ WeekDayNum + Vacation + NormHour +
    Quarter + Prec + TMed + VelMedia, data = csvTest)

Residuals:
     Min       1Q   Median       3Q      Max
-0.02552 -0.01311 -0.01071 -0.00819  0.99280

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.0064092  0.0028970   2.212   0.0270 *
WeekDayNum   0.0022877  0.0024774   0.923   0.3558
Vacation    -0.0001962  0.0030176  -0.065   0.9482
NormHour     0.0065520  0.0024414   2.684   0.0073 **
Quarter      0.0054484  0.0026451   2.060   0.0394 *
Prec         0.0090929  0.0063826   1.425   0.1543
TMed        -0.0072370  0.0029819  -2.427   0.0152 *
VelMedia    -0.0036080  0.0036290  -0.994   0.3201
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.06446 on 8032 degrees of freedom
Multiple R-squared:  0.002646,  Adjusted R-squared:  0.001776
F-statistic: 3.044 on 7 and 8032 DF,  p-value: 0.003372
```

Figura 24. Salida regresión linear - accidentes de tráfico.

Para confirmar la teoría, se procede a visualizar los resultados tal y como se explica en el libro *El paquete estadístico R* (Freijo, 2013). El código utilizado es el siguiente:

```
#Se crea un objeto grafico y se define el numero de objetos graficos
que tiene
par(mfrow=c(2,1))
#Se establece el titulo del grafico
title<-'Probabilidad de accidente: valores observados y estimados'
#Se crea el grafico con la funcion plot() mostrando los valores
observados
plot(csvTest$Probability, main=title, xlab='Observacion', type='b',
col=1)
#Se anaden los valores estimados por la regresion
lines(fitted(regresion), type='b', col=2)
#Se crea el grafico de residuos
plot(residuals(regresion), main='Residuos', xlab='Observacion',
type='b')
```

En la *Figura 25* se presentan los gráficos obtenidos. El primero de ellos compara los valores observados (en negro) y los valores obtenidos al realizar la regresión (en rojo). Como se puede observar, el modelo no predice correctamente. En el segundo gráfico, se observan los residuos, es decir, las diferencias entre los valores observados y los estimados para la

variable dependiente (Probability). Estos valores deberían cumplir la hipótesis de tener media nula, pero observando el gráfico se parecía que no se cumple, por lo que se tiene otro motivo más para confirmar la teoría de que el modelo no es bueno.

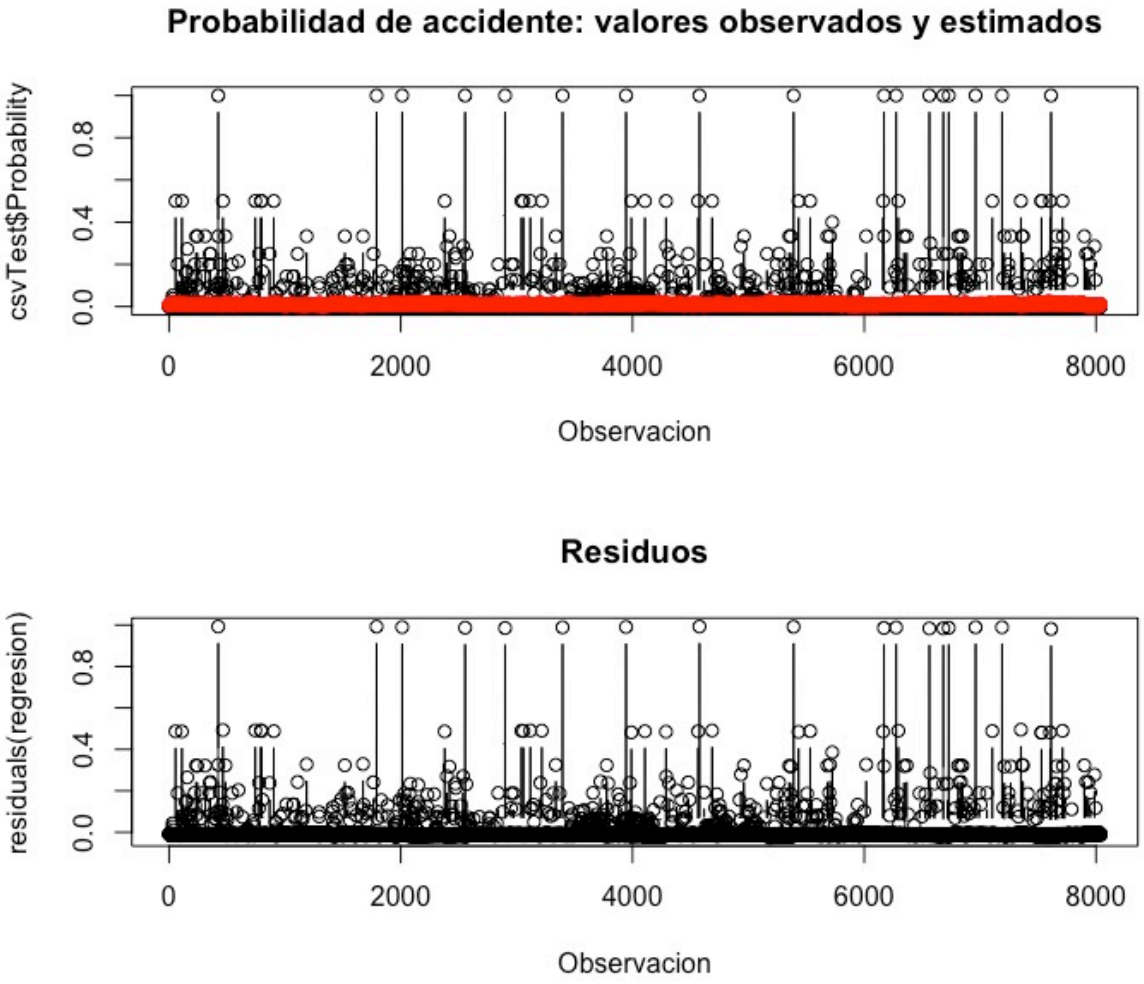


Figura 25. Gráficos obtenidos de realizar una regresión lineal.

4.5. Redes neuronales

En la sección anterior se ha comprobado que utilizar regresiones lineales con los datos disponibles no genera un modelo predictivo fiable. En este aparto se va a crear una red neuronal para comparar el resultado.

Las redes neuronales artificiales son una de las técnicas de aprendizaje automáticos por excelencia. Aprenden a partir de la experiencia y emulan el funcionamiento del cerebro humano (Universidad Internacional de La Rioja, 2016).

Para llevar a cabo esta parte del trabajo también se ha utilizado R, concretamente el paquete *neuralnet* (rdocumentation.org, 2017). Para validar el modelo se utiliza la técnica de

validación cruzada, que consiste en dividir el conjunto de datos disponibles en dos: uno para el entrenamiento y otro para la validación. Con la función *neuralnet* se entrena la red neuronal estableciendo los siguientes parámetros:

- Formula: se indica qué variables se quieren predecir y cuáles son las predictoras. El formato es el siguiente: $Y \sim X_1+X_2+X_3$.
- Data: los datos que contienen las variables especificadas en la fórmula.
- Hidden: indica el número de capas ocultas o intermedias de la red neuronal y el número de neuronas de cada una de ellas. Este parámetro admite un vector como entrada, en el que cada posición del mismo es una capa oculta y el valor almacenado en cada uno es el número de neuronas. De esta forma, si le asignamos el vector *c(2,4,3)* al parámetro, estamos creando una red neuronal con tres capas ocultas de dos, cuatro y tres neuronas respectivamente.
- Threshold: valor numérico con el que, si el error obtenido es menor que dicho valor, el algoritmo se para y da por bueno el resultado.
- Stepmax: número máximo de iteraciones del algoritmo.
- Rep: número de entrenamientos de la red neuronal.
- Algorithm: algoritmo utilizado para calcular la red neuronal.
- Err.fct: función utilizada para calcular el error.

Estos son los parámetros principales de la función. Para conocerlos todos, acceder al sitio web https://www.rdocumentation.org/packages/neuralnet/versions/1.33/topics/neuralnet.

Con la función neuralnet se entrena la red neuronal. A continuación, con la función *compute* se aplica dicha red neuronal al “data frame” de test. Se utiliza la raíz cuadrada del error cuadrático medio (RMSE por sus siglas en inglés - Root Mean Square Error) para poder comparar los modelos y determinar cuál de ellos es más preciso. El script utilizado es el siguiente:

```
#Se inicializa el espacio de trabajo
rm(list=ls())
#Se carga el paquete neuralnet
library(neuralnet)
#Se imprime por consola el momento en el que se inicia el
entrenamiento
print (paste('Init: ', Sys.time()))
#--------------------M30 calzada 1, zona 2
#Se carga el fichero CSV con los datos destinados al test
csvTest =
read.csv('/Users/luis/Documents/BigData/UNIR/Master/9.TFM/Datasets/Acc
identes Desagregado/Validation_M30_Calz1_Zone2.csv')
#Se carga el fichero CSV con los datos destinados al entrenamiento
csvTrain =
read.csv('/Users/luis/Documents/BigData/UNIR/Master/9.TFM/Datasets/Acc
identes Desagregado/Train_M30_Calz1_Zone2.csv')
#Se transforman a data frames
dfTest = as.data.frame(csvTest)
dfTrain = as.data.frame(csvTrain)
#Se establece la fromula que indica que la variable que se pretende
predecir
#es Probability, mientras que WeekDayNum, Vacation, NormHout, Quarter,
Prec,
#TMed y Velmedia con las variables predictoras
formula <-
as.formula("Probability~WeekDayNum+Vacation+NormHour+Quarter+Prec+TMed
+VelMedia")
#Se parametriza la funcion neuralnet
neuralNetworkTraining<-neuralnet(formula,dfTrain,hidden=2, threshold =
0.01, stepmax = 1e5)
#Grafico con el esquema de la red neuronal
plot(neuralNetworkTraining)
#Datos de test
dfTestClean <- dfTest[,c('WeekDayNum', 'Vacation', 'NormHour',
'Quarter', 'Prec', 'TMed', 'VelMedia')]
```

```
#Resultado obtenido de aplicar la red neuronal sobre los datos del
test
results <- compute(neuralNetworkTraining, dfTestClean)
#Data frame con los datos observados y los estimados
output <-
cbind(dfTestClean,dfTest[,c('Probability')],as.data.frame(results$net.
result))
names(output)[8]<-'Probability'
names(output)[names(output) == 'V1']<-'Estimated'
#Se imprime por consola la hora de finalizacion del proceso
print (paste('End: ', Sys.time()))
#Se guarda el resultado de la prediccion en un fichero CSV
write.csv(output,'/Users/luis/Documents/BigData/UNIR/Master/9.TFM/Data
sets/Accidentes
Desagregado/Output/20170907_RedNeuronalM30_Calz1_Zone2.csv')
#Se calcula el error de los valores estimados en comparacion con los
valores observados
error <- output[,'Probability']-output[,'Estimated']
#Se calcula la raiz cuadrada del erro cuadratico medio (RMSE)
rmseError <- sqrt(mean(error^2))
```

Parametrización 1

La fórmula utilizada es siempre "Probability ~ WeekDayNum + Vacation + NormHour + Quarter + Prec + TMed + VelMedia". Sin embargo, se va a probar distintas configuraciones de la red neuronal. En una primera aproximación, los valores utilizados son los siguientes:

- Hidden=2
- threshold=0.01
- stepmax=1e5

Por defecto, el algoritmo utilizado por la función *neuralnet* es *resilient backpropagation* (*rprop*). En la *Figura 26* se puede apreciar el esquema de la red neuronal obtenida. El valor del RMSE para esta parametrización de la red neuronal es de **0.05795262207**. Este valor es cercano a 0. Sin embargo, es engañoso debido a que todos los valores estimados son cercanos al 0 y la mayoría de las probabilidades también. Por este motivo, el error en la mayoría de las observaciones es muy pequeño. Sin embargo, si se analizan únicamente las observaciones con probabilidad mayor que 0 para comprobar el error, se observada que el valor del RMSE obtenido es **0.2240609515**.

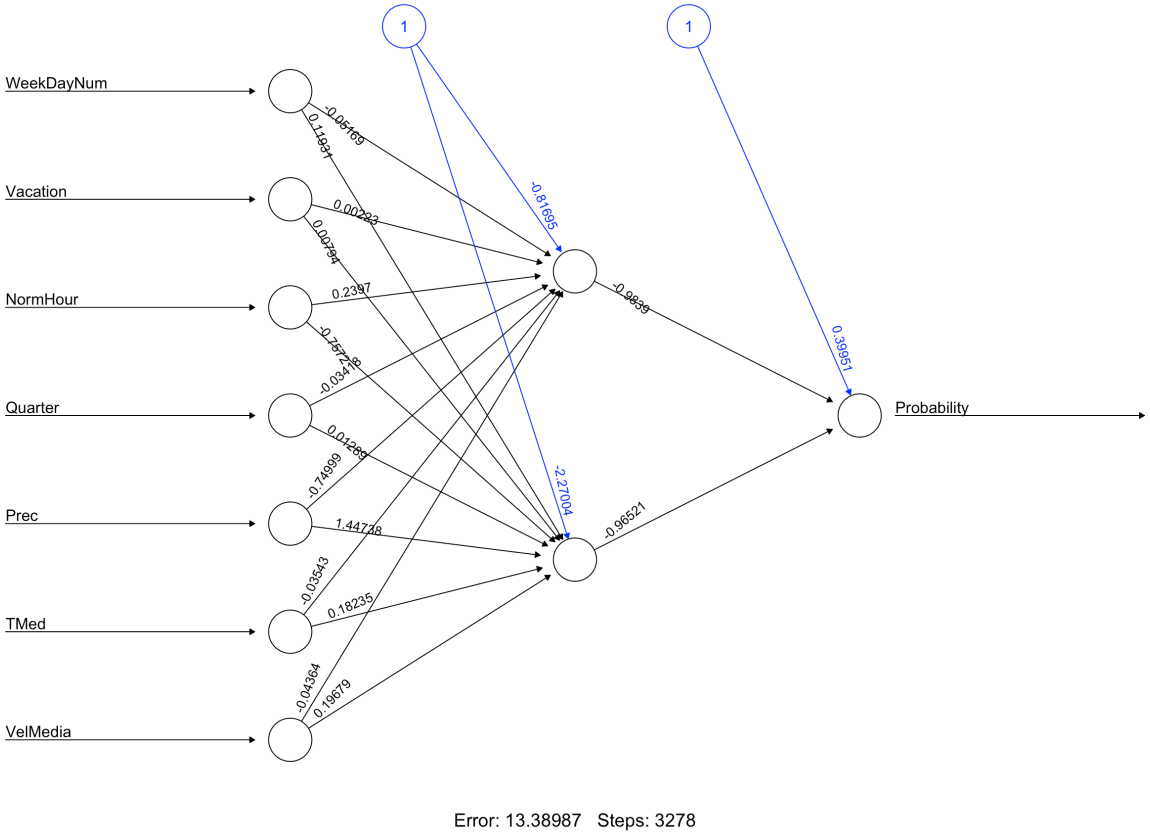


Figura 26. Esquema red neuronal parametrización 1

Parametrización 2

En este caso se va a incluir una capa oculta más de tres neuronas para comprobar el efecto que produce en el resultado. La configuración es la siguiente:

- Hidden=c(2,3)
- threshold=0.01
- stepmax=1e5

El esquema de la red neuronal obtenida se puede ver en la *Figura 27*. El valor obtenido de RMSE para esta configuración es **0.1073068407**. Al añadir una capa de neuronas al modelo se esperaba obtener un resultado mejor que en la parametrización anterior. Sin embargo, el error obtenido es mayor, por lo que el modelo pierde precisión con respecto al obtenido con la parametrización 1.

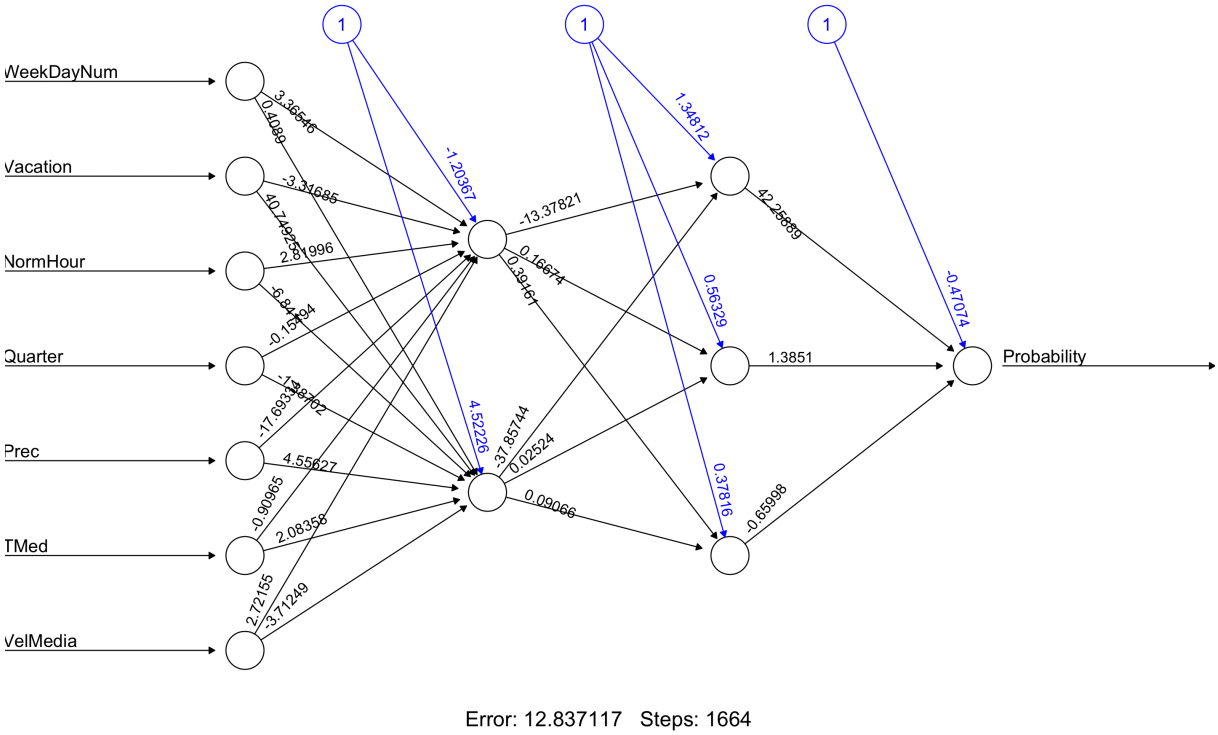


Figura 27. Red neuronal parametrización 2

Parametrización 3

El algoritmo utilizado por defecto en la función *neuralnet* es *rprop*, el cual es una variante del algoritmo *backpropagation*, para el que no es necesario fijar un *learning rate*. En las parametrizaciones anteriores se ha utilizado este algoritmo para construir los modelos. Con el objetivo de obtener modelos más precisos que los anteriores, se procede a probar el algoritmo *backpropagation* con un *learning rate* de 0.0001 (para valores mayores de este parámetro el algoritmo no converge). Por tanto, la parametrización en este caso es la siguiente:

- Hidden=0
- threshold=0.01
- stepmax=1e5
- algorithm = 'backprop'
- learningrate = 0.0001

El esquema de la red neuronal se puede observar en la *Figura 28*. El valor de RMSE obtenido para esta configuración es **0.05771381917**, el cual es algo mejor que el obtenido en la parametrización 2, pero similar al obtenido en la parametrización 1. Por tanto, el modelo obtenido con esta configuración no mejora notablemente los resultados.

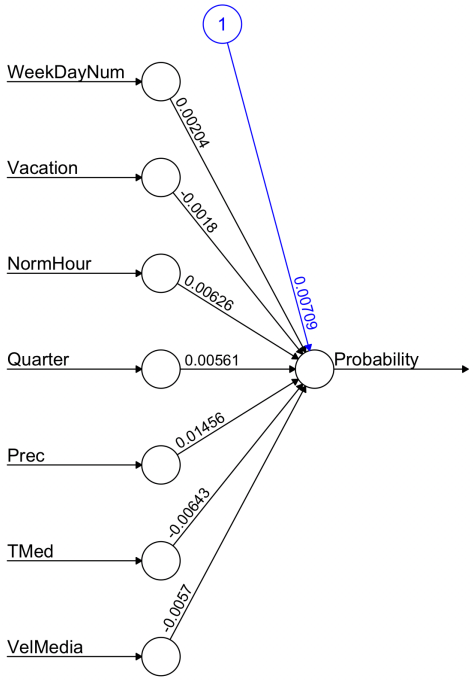


Figura 28. Red neuronal parametrización 3

Parametrización 4

Para calibrar el modelo se procede a incluir una capa oculta con dos neuronas, manteniendo el algoritmo *backpropagation*. Se espera obtener un mejor resultado dado que el número de capas ocultas ya no es 0, por lo que debería obtener una precisión mayor en comparación con la parametrización anterior. La configuración utilizada es la siguiente:

- Hidden=2
- threshold=0.01
- stepmax=1e5
- algorithm = 'backprop'
- learningrate = 0.0001

El esquema de la red neuronal se puede observar en la *Figura 29*. El valor de RMSE obtenido para esta configuración es **0.05774828713**. Como se puede observar, el error obtenido es similar a las parametrizaciones 1 y 4, con lo que esta configuración no mejora los modelos anteriores a pesar de haber utilizado una capa oculta de neuronas.

En las *Figuras 30 y 31* se representan los valores estimados y los observados, y el error del modelo respectivamente. La forma de los gráficos es muy parecida a la de los obtenidos calculando regresiones lineales sobre los datos.

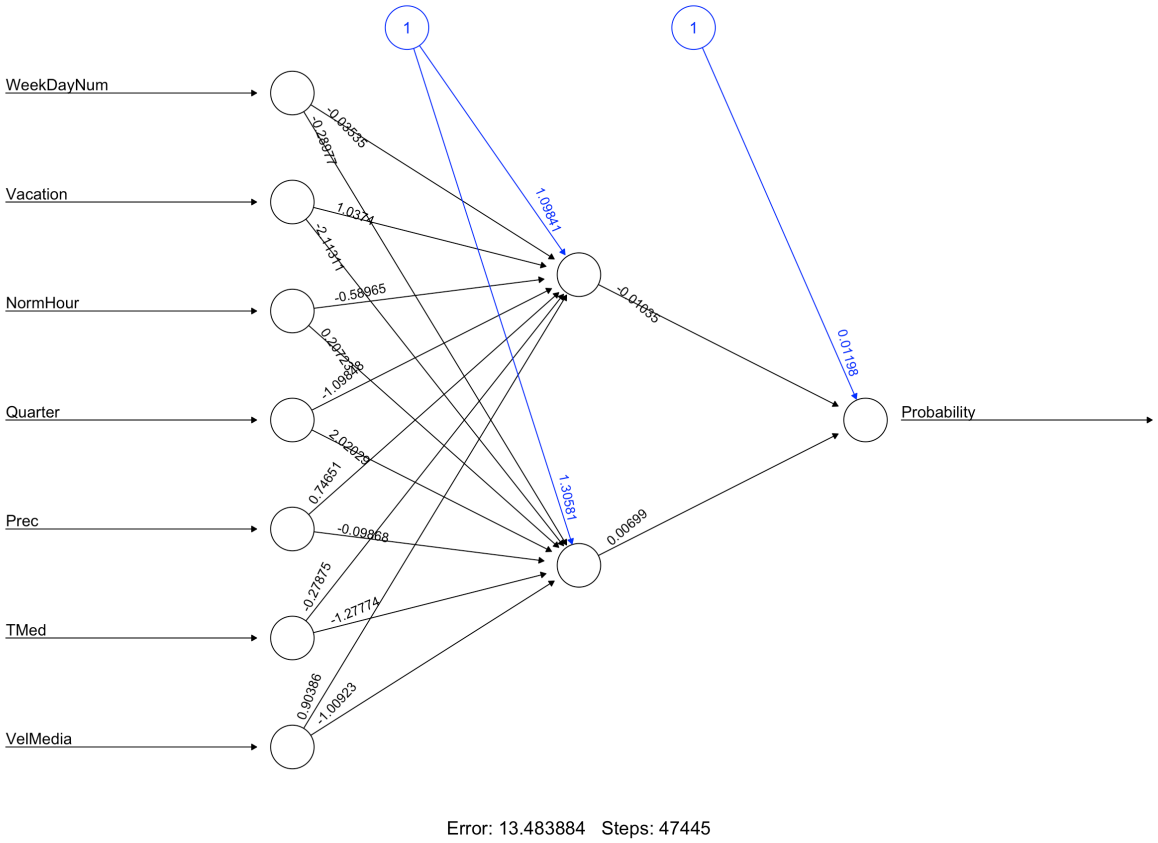


Figura 29. Red neuronal parametrización 4.

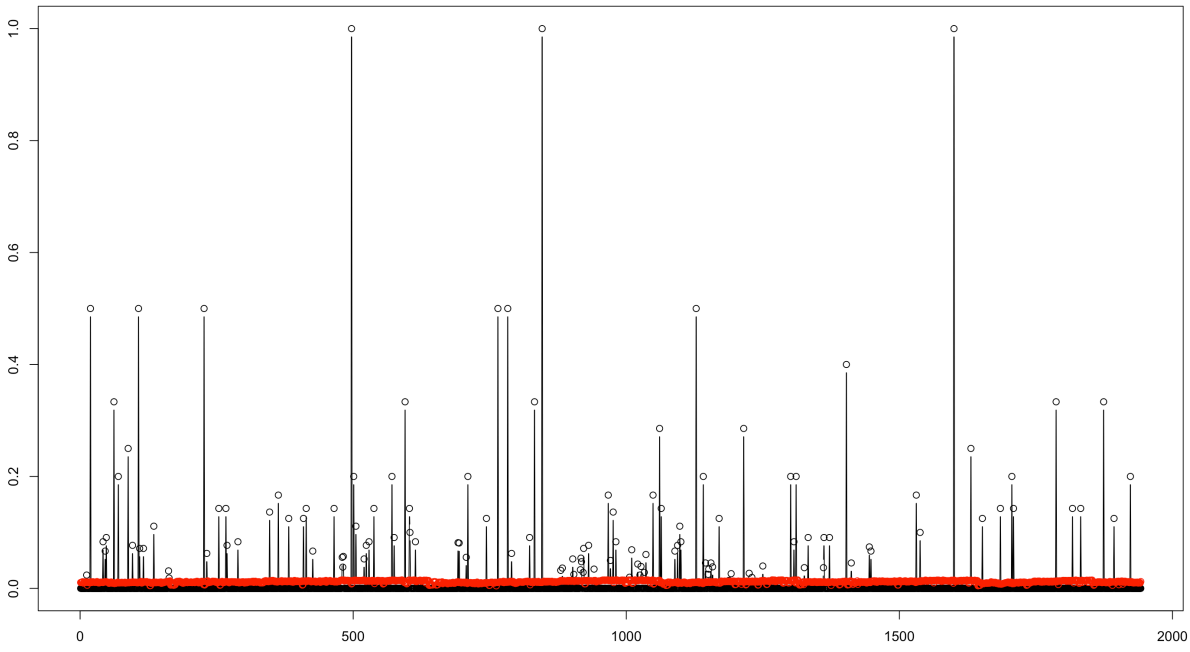


Figura 30. Datos observados en negro y estimados en rojo – parametrización 4.

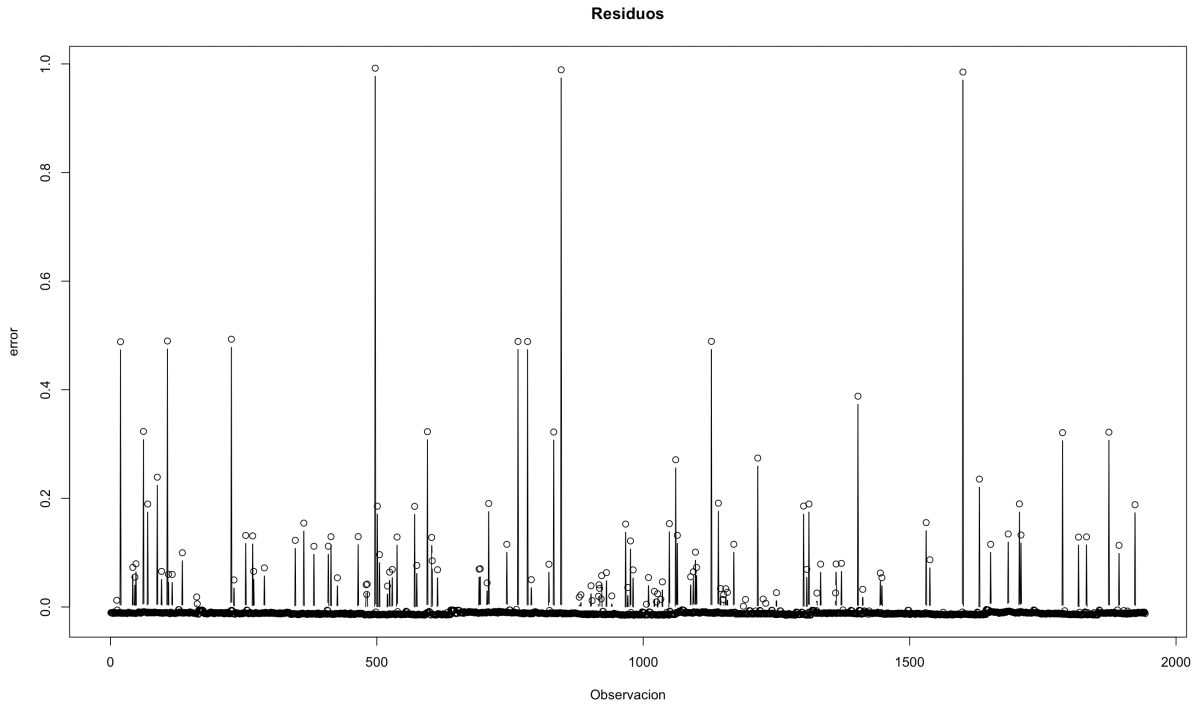


Figura 31. Residuos parametrización 4

Parametrización 5

En esta parametrización se ha utilizado el algoritmo por defecto de la función *neuralnet*: *rprop*. En este caso se ha activado la opción *linear.output*, se han creado tres capas ocultas de dos, tres y dos neuronas respectivamente, y se ha forzado a la función a repetir el entrenamiento cuatro veces para seleccionar el mejor resultado. Con esta configuración se pretende mejorar la precisión del modelo debido a la presencia de más capas ocultas que en las anteriores parametrizaciones, aplicando una regresión al resultado y repitiendo el entrenamiento cuatro veces. La configuración utilizada es la siguiente:

- Hidden= c(2,3,2)
- threshold=0.01
- stepmax=1e5
- linear.output = TRUE
- rep=4

El esquema de la red neuronal se puede observar en la *Figura 32*. El valor de RMSE obtenido para esta configuración es **0.05803558388**. Si se toman únicamente aquellas observaciones en las que la probabilidad de ocurrencia de un accidente es mayor que cero, se obtiene un RMSE de **0.2239421466**. Los valores obtenidos no mejoran los correspondientes a las parametrizaciones anteriores, mostrando valores similares a los de la parametrización 1, por lo que el modelo es igual de preciso en ambos casos.

En las *Figuras 33 y 34* se representan los valores estimados y los observados, y el error del modelo respectivamente.

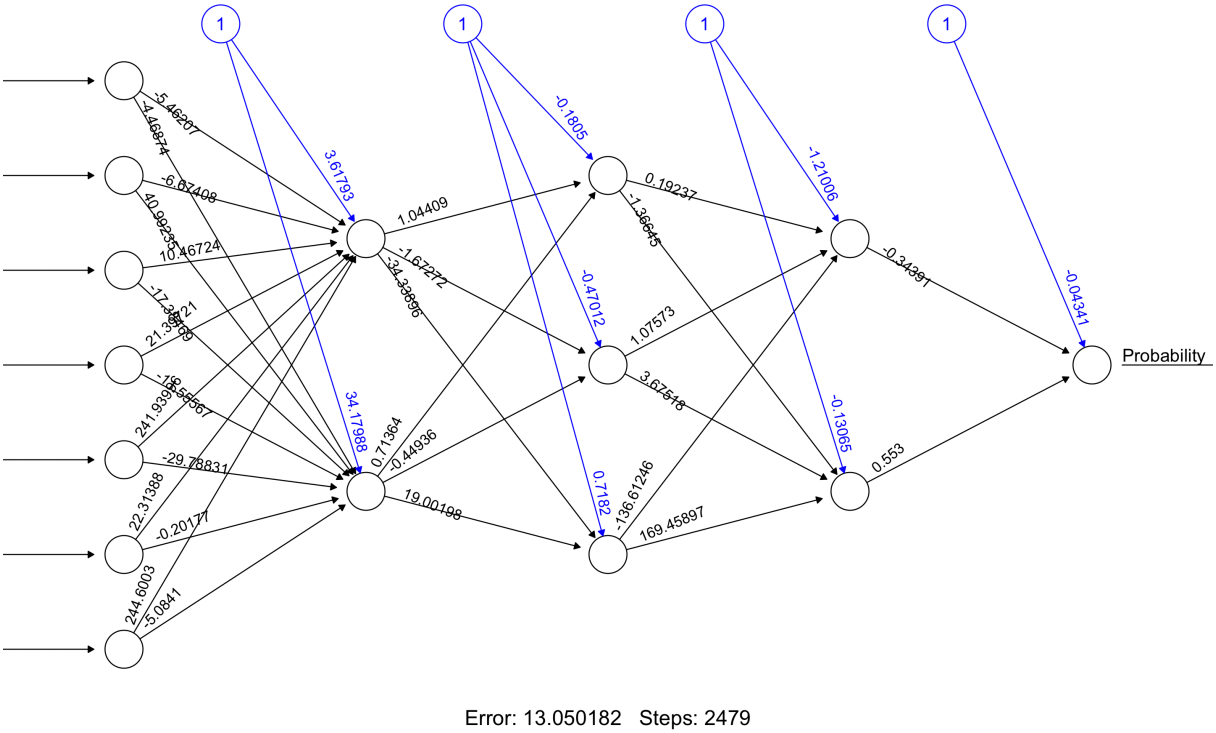


Figura 32. Red neuronal parametrización 5.

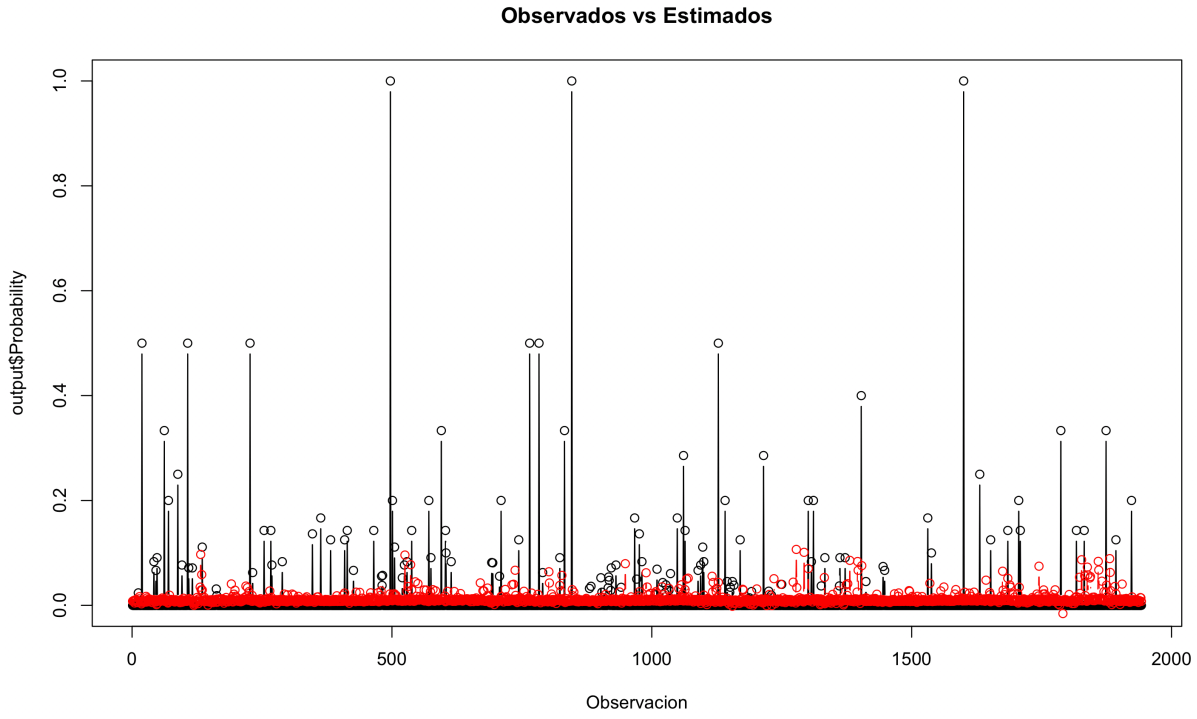


Figura 33. Valores observados vs estimaciones – parametrización 5.

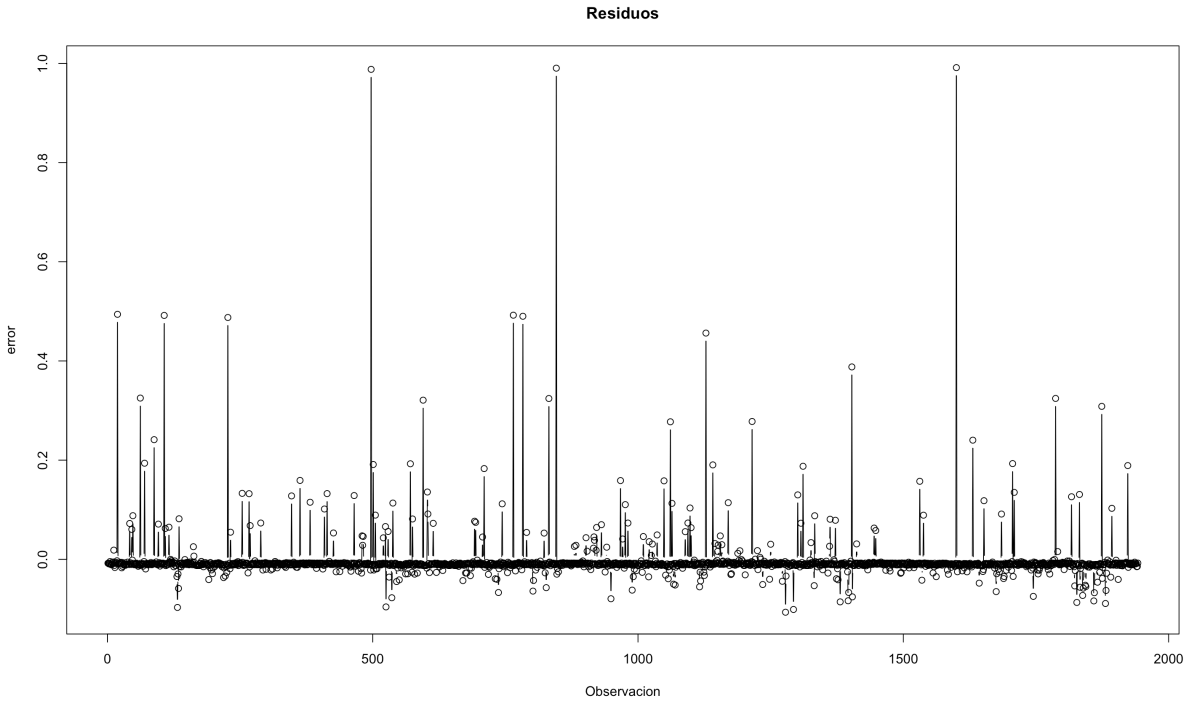


Figura 34. Residuos – parametrización 5.

Otras parametrizaciones

Con el objetivo de calibrar los modelos para que sean más precisos, se han realizado pruebas con otras parametrizaciones, ajustando el número de capas ocultas, bajando el threshold y aumentando el número máximo de iteraciones. Con estos cambios se esperaba obtener mejores resultados, ya que al bajar el threshold, el error obtenido debería ser menor. Sin embargo, en muchos casos el algoritmo no llega a converger a un resultado en el que el error sea menor o igual que el threshold configurado. Aquellos casos en los que se ha llegado a converger, el RMSE obtenido es muy parecido al mostrado en las parametrizaciones anteriores.

Algunas configuraciones a destacar:

- hidden = c(2,3), threshold = 0.01, stepmax = 5e5, algorithm = 'backprop', learningrate = 0.00001. El algoritmo ha tardado dos horas en agotar el número máximo de iteraciones (500.000). No ha llegado con converger, por lo que no se ha podido obtener un modelo.
- hidden=2, threshold = 0.001, stepmax = 1e7, algorithm = 'backprop', learningrate = 0.00001. En este caso se ha bajado el threshold a un valor muy pequeño. Con este cambio se corre el riesgo de que el algoritmo no llegue a converger, por lo que se ha aumentado el número máximo de iteraciones dos órdenes de magnitud (10.000.000). Con esta configuración, el algoritmo ha tardado más de 24 horas en ejecutarse y, tras agotar el número máximo de iteraciones, no ha convergido.

5. Conclusiones y trabajo futuro

Una de las causas de muerte externa más comunes en los países desarrollados son los accidentes de tráfico. Es un hecho que los accidentes provocan numerosos fallecimientos y lesiones crónicas, por lo que se pueden considerar como una lacra para la sociedad.

Los gobiernos invierten importante cantidad de recursos en mejorar la seguridad vial, pero la realidad es que la cifra de damnificados sigue siendo muy alta. En España en 2016 murieron 1160 personas en accidentes de tráfico (Dirección General de Tráfico, 2017).

Este trabajo se ha centrado en los accidentes de tráfico ocurridos en la ciudad de Madrid, pretendiendo encontrar un modelo que pueda predecir la probabilidad de ocurrencia de un accidente en una zona concreta. Para ello se ha buscado información histórica de accidentes de tráfico en la ciudad (cedido por el área de "AGSSyE - DGPM / Unidad de Calidad y Evaluación" del Ayuntamiento de Madrid), el calendario laboral de cada año (calendarioslaborales.com, 2012) y la información meteorológica histórica (Agencia Estatal de Meteorología, 2017). A continuación, se ha procedido a estudiar los datos y transformarlos para poder calcular modelos predictivos sobre estos.

Las variables elegidas para construir los modelos han sido: el día de la semana, la franja horaria, el trimestre del año, si el día es festivo, la velocidad media del viento en ese día, la temperatura media de ese día y el volumen de precipitaciones del día en el que se produce el accidente. Todas las variables se han escalado de tal manera que toman valores entre 0 y 1. Después se ha procedido a dividir las calles en las que se han producido los accidentes por zonas, para poder establecer un área relativamente reducida en la que poder aplicar el modelo. Tras hacer un ranking, se ha centrado el estudio en la Calle M-30 calzada 1, zona 2, obteniendo la probabilidad de que se produzca un accidente en función de las combinaciones que se dan en el histórico de accidentes de tráfico de las variables predictoras elegidas. Por ejemplo, si se dan cuatro casos en el histórico de accidentes en los que el día de la semana es 1, la franja horaria es 0.25, el valor del trimestre es 0.5, el valor de día festivo es 0, la velocidad media del viento es 0.2, la temperatura media es 0.3, y las precipitaciones toman el valor 0.1, y en uno de los casos se produce un accidente de tráfico en la Calle M-30 calzada 1 y en el kilómetro correspondiente a la zona 2 pero en los otros tres casos no se produce accidente en dicha área, entonces la probabilidad de ocurrencia de un accidente para esa combinación de variables es de 0.25 (1/4). Con los datos ya procesados, se ha divido el conjunto de datos en dos: datos de entrenamiento y datos de test.

El siguiente paso ha sido calcular una regresión lineal sobre los datos obtenidos del procesamiento anterior. Para ello se ha utilizado la función *lm()* de R. Los resultados obtenidos se pueden ver en la sección *4.4. Aplicando Regresiones*.

Con el objetivo de comparar distintos modelos, se han construido redes neuronales con los datos disponibles (ver sección *4.5. Redes Neuronales*). En esta parte del trabajo se ha utilizado el paquete *neuralet* de R y se han realizado diferentes parametrizaciones sobre la función de entrenamiento, cambiando el número de capas ocultas de la red, el algoritmo de cálculo de la red neuronal, etc. En dicha sección se han explicado cinco de ellas, considerándolas representativas del resultado obtenido. Muchas de las parametrizaciones probadas no han llegado a converger a un error inferior al threshold elegido en el número máximo de iteraciones dado, por lo que el entrenamiento no ha sido satisfactorio, tardando en ejecutarse más de veinticuatro horas.

El resultado obtenido tanto en la regresión lineal como en las parametrizaciones es muy similar. Para acotar el enfoque de los modelos, se ha reducido el área de la ciudad sobre la que se han construido los modelos. Esto ha propiciado que la variable dependiente (la probabilidad de ocurrencia de accidente en dicha zona) tome el valor 0 el 94% de las veces, tanto para los datos de entrenamiento como en los datos de validación. Debido al alto número de veces que toma el valor 0, el entrenamiento provoca que la mayoría de los resultados obtenidos en la estimación tome también el valor 0. Si se tiene en cuenta el error cuadrático medio de la estimación con respecto a los valores observados, el resultado es aceptable, ya que el RMSE para la mayoría de los modelos obtenidos es de aproximadamente 0.057, lo que indica que el error es pequeño. Sin embargo, este resultado es engañoso por lo que se ha comentado anteriormente (porcentaje elevado de observaciones con valor 0). Si se tienen en cuenta las estimaciones obtenidas para aquellos casos en los que el valor observado toma valores mayores que 0 (existe probabilidad de que se produzca un accidente), el RMSE obtenido con las predicciones es aproximadamente 0.22, una cifra considerablemente peor que el RMSE obtenido con el total de los datos.

Para ver lo comentado en el párrafo anterior de manera gráfica, se puede observar, por ejemplo, la *Figura 33*. En esta, se muestra un gráfico con las observaciones en negro y los resultados obtenidos de la predicción en rojo. Ambos valores se acumulan principalmente cerca del 0 y, por ello, el RMSE obtenido para el conjunto de todos los datos es tan bajo. Sin embargo, si se centra la atención en aquellas observaciones que toman un valor mayor que 0, se aprecia que los valores correspondientes de las estimaciones (en rojo) no se acercan al valor de la observación. Por este motivo, el RMSE para este conjunto de datos es mucho

mayor. Esto indica que, para los casos en los que se puede producir un accidente, el modelo no predice con exactitud la probabilidad de ocurrencia del mismo.

Uno de los motivos por los que los resultados obtenidos no son excesivamente positivos puede deberse a que las variables elegidas para la predicción probablemente no sean las más adecuadas, ya que la correlación lineal existente entre las variables predictoras y la variable dependiente no es alta (ver *Figura 35*).

	Probability
X	-0.028179305594
WeekDayNum	0.015188806129
Vacation	0.025724759573
NormHour	0.038349624183
Quarter	0.010060123079
Prec	-0.013034747128
TMed	-0.037287926116
VelMedia	0.005740920542
Probability	1.000000000000

Figura 35. Coeficiente de correlación de Pearson entre las variables predictoras y la variable a estimar

En el futuro se explorarán nuevas variables que expliquen mejor el comportamiento de los accidentes de tráfico en Madrid. Tal y como comentan Blanca Arenas, Francisco Aparicio y Camino González en su publicación “Intervalos de predicción de la frecuencia de accidentes en tramos de la Red de Carreteras Españolas, para flujos heterogéneos”, los analistas consideran los flujos de tráfico, la geometría de la vía o el tipo de vía como variables explicativas de la frecuencia de accidentes de tráfico. Por ello, una de las variables a utilizar se encuentra disponible en la sección de Open Data del Ayuntamiento de Madrid (Ayuntamiento de Madrid - Intensidad del tráfico, 2014). Se trata de un conjunto de datos en los que se tiene la información sobre la intensidad de tráfico en la ciudad. Hay disponible un histórico desde julio de 2013.

Adicionalmente, la mayoría de analistas consideran que los modelos lineales generales en sus alternativas de Poisson y de binomial negativa son los más adecuados para la construcción de modelos de predicción de frecuencia de accidentes de tráfico (Blanca Arenas, 2008). Por ello, se utilizarán estas técnicas para la construcción de modelos predictivos con el objetivo de obtener una mayor precisión en estos junto con el ajuste de dichos modelos mediante el tratamiento de outliers (J. Soler Flores, 2008).

6. Bibliografía

Agencia Estatal de Meteorología. (10 de 08 de 2017). *aemet.es*. Obtenido de www.aemet.es: https://opendata.aemet.es/centrodedescargas/productosAEMET?

Ayuntamiento de Madrid - Accidentes Tráfico. (21 de 07 de 2014). *http://datos.madrid.es*. Recuperado el 10 de 06 de 2017, de http://datos.madrid.es: http://datos.madrid.es/sites/v/index.jsp?vgnextoid=746b86396f847410VgnVCM2000000c205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD

Ayuntamiento de Madrid - Intensidad del tráfico. (24 de 09 de 2014). *datos.madrid.es*. Recuperado el 02 de 09 de 2017, de datos.madrid.es: http://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=33cb30c367e78410VgnVCM1000000b205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default

Ayuntamiento de Madrid. (01 de 09 de 2017). *www.madrid.es*. Recuperado el 01 de 09 de 2017, de www.madrid.es: http://datos.madrid.es/portal/site/egob/

Ayuso, F. J. (1999). Modelo de predicción de ocurrencia de accidentes en tramos de carretera mediante la medición continua de variables de influencia. Madrid, España.

Blanca Arenas, e. a. (2008). Intervalos de predicción de la frecuencia de accidentes en tramos de la Red de Carreteras Españolas, para flujos heterogéneos.

calendarioslaborales.com. (01 de 01 de 2012). *calendarioslaborales.com*. Obtenido de calendarioslaborales.com: http://www.calendarioslaborales.com/calendario-laboral-madrid-2012.htm

Catherine Pérez, E. C. (2001). ESTUDIO DE LA MORTALIDAD A 30 DÍAS POR ACCIDENTES DE TRÁFICO (EMAT-30). Barcelona, España.

Cecilia Montt, e. a. (01 de 10 de 2013). ANÁLISIS DE ACCIDENTES DE TRÁNSITO CON INTELIGENCIA COMPUTACIONAL. Santiago, Chile.

Chazallet, S. (2016). *Python 3: los fundamentos del lenguaje.* Barcelona, España: Ediciones ENI.

colegiosmadrid.net - calendario escolar 2012-2013. (01 de 01 de 2012). *colegiosmadrid.net*. Obtenido de colegiosmadrid.net: http://www.colegiosmadrid.net

Deportes Ayuntamiento de Madrid. (01 de 09 de 2017). *http://datos.madrid.es/*. Recuperado el 01 de 09 de 2017, de http://datos.madrid.es/: http://datos.madrid.es/sites/v/index.jsp?vgnextoid=374512b9ace9f310VgnVCM100000171f5a0aRCRD&buscar=true&Texto=&Sector=deporte&Formato=&Periodicidad=&orderByCombo=CONTENT_INSTANCE_NAME_DECODE

Dirección General de Tráfico. (03 de 01 de 2017). *DGT*. Recuperado el 29 de 08 de 2017, de DGT: http://revista.dgt.es/es/noticias/nacional/2017/01ENERO/0103balance-accidentes-2016.shtml#.WaWtKximNE4

djangogirls.org. (01 de 09 de 2017). *https://tutorial.djangogirls.org/es/python_installation/*. Recuperado el 01 de 09 de 2017, de https://tutorial.djangogirls.org/es/python_installation/: https://tutorial.djangogirls.org/es/python_installation/

Dominique Lord, B. N. (01 de 01 de 2000). Accident Prediction Models With and Without Trend: Application of the Generalized Estimating Equations (GEE) Procedure. Toronto, Canada.

educa.madrid.org - calendario escolar 2011-2012. (01 de 01 de 2011). *educa.madrid.org*. Obtenido de educa.madrid.org: http://www.educa.madrid.org/

Enrique Carbonell, R. B.-T. (2001). El ambiente de tráfico como generador de ansiedad en el conductor: Inventario de situaciones ansiógenas en el tráfico. Barcelona, España.

espormadrid.es - calendario escolar 2013-2014. (01 de 01 de 2013). *espormadrid.es*. Obtenido de espormadrid.es: http://www.espormadrid.es/2013/07/calendario-escolar-curso-2013-2014-en.html

espormadrid.es - calendario escolar 2014-2015. (01 de 01 de 2014). *espormadrid.es*. Obtenido de espormadrid.es: http://www.espormadrid.es/2014/09/calendario-escolar-curso-2014-15-de-la.html

europapress.es - calendario escolar 2015-2016. (01 de 01 de 2015). *europapress.es*. Obtenido de europapress.es: http://www.europapress.es/madrid/noticia-calendario-escolar-2015-2016-madrid-festivos-puentes-fiestas-locales-20150828133545.html

Freijo, J. B. (2013). *El paquete estadístico R, página 217.* Madrid, Madrid, España.

GNU. (02 de 09 de 2017). *https://www.gnu.org*. Recuperado el 02 de 09 de 2017, de https://www.gnu.org: https://www.gnu.org

https://sedeapl.dgt.gob.es/WEB_IEST_CONSULTA/. (17 de 02 de 2015). *https://sedeapl.dgt.gob.es/WEB_IEST_CONSULTA/*. Recuperado el 10 de 08 de 2017, de https://sedeapl.dgt.gob.es/WEB_IEST_CONSULTA/: https://sedeapl.dgt.gob.es/WEB_IEST_CONSULTA/

Instituto Nacional de Estadística (INE). (30 de 03 de 2016). *INE*. Recuperado el 29 de 08 de 2017, de INE: http://www.ine.es/prensa/np963.pdf

J. Soler Flores, e. a. (2008). Tratamiento de outliers en los modelos de predicción de accidentes de tráfico. Madrid, España.

M. Ruiz Ramosa, e. a. (01 de 01 de 2004). Evolución de la mortalidad por accidentes de tráfico en Andalucía desde 1975 hasta 2001 y predicción.

madrid.org - calendario escolar 2016-2017. (01 de 01 de 2016). *madrid.org*. Obtenido de madrid.org: http://www.madrid.org

Python Software Foundation. (02 de 09 de 2017). *https://www.python.org*. Recuperado el 02 de 09 de 2017, de https://www.python.org: https://www.python.org

rdocumentation.org. (02 de 08 de 2017). *rdocumentation.org*. Recuperado el 02 de 08 de 2017, de rdocumentation.org: https://www.rdocumentation.org/packages/neuralnet/versions/1.33/topics/neuralnet

r-project.org. (02 de 09 de 2017). *https://www.r-project.org/*. Recuperado el 02 de 09 de 2017, de https://www.r-project.org/: https://www.r-project.org/

rstudio.com. (02 de 09 de 2017). Recuperado el 02 de 09 de 2017, de https://www.rstudio.com/products/rstudio/download/

Sublime Text. (01 de 09 de 2017). *https://www.sublimetext.com*. Recuperado el 01 de 09 de 2017, de https://www.sublimetext.com: https://www.sublimetext.com

sublimetext.com. (02 de 09 de 2017). *sublimetext.com*. Recuperado el 02 de 09 de 2017, de sublimetext.com: https://www.sublimetext.com/3

Trochim, W. M. (01 de 01 de 2006). *socialresearchmethods.net*. Recuperado el 04 de 09 de 2017, de socialresearchmethods.net: https://www.socialresearchmethods.net/kb/dummyvar.php

Universidad Internacional de La Rioja. (01 de 10 de 2016). Análisis e interpretación de datos - Tema 3. Relación entre variavles. La Rioja, España.

Universidad Internacional de La Rioja. (01 de 12 de 2016). Métodos de almacenamiento de información. La Rioja, España.

Universidad Internacional de La Rioja. (02 de 10 de 2016). Técnicas de Inteligencia Artificial - Tema9. Redes Neuronales Artificiales. La Rioja, España.

Urbanismo Ayuntamiento de Madrid. (01 de 09 de 2017). *http://datos.madrid.es*. Recuperado el 01 de 09 de 2017, de http://datos.madrid.es: http://datos.madrid.es/sites/v/index.jsp?vgnextoid=374512b9ace9f310VgnVCM10000171f5a0aRCRD&buscar=true&Texto=&Sector=urbanismo-infraestructuras&Formato=&Periodicidad=&orderByCombo=CONTENT_INSTANCE_NAME_DECODE

Verzani, J. (2011). *Getting Started with RStudio.* USA: O'Reilly Media.

7. Anexos

7.1. Módulos Python

csvToDataFrame.py

```
import csv
import pandas

def getDataFrameFromCsv(pathFile):
        with open(pathFile) as auxCsv:
                #Se indica que el delimitador es una coma y las
comillas dobles se usan para delimitar un campo de texto
                dataCsv = csv.reader(auxCsv, delimiter=',',
quotechar='"')
                #Se convierte en una lista
                dataList = list(dataCsv)
                #Se crea un data frame con dicha lista
                dataDF = pandas.DataFrame(dataList[1:],columns =
dataList[0])
                return dataDF
```

jsonToDF.py

```
import pandas
import numpy

def getMeteorologyDataFrame():
	#Con la funcion read_json del modulo pandas cargamos el primer
fichero JSON
	df =
pandas.read_json('/Users/luis/Documents/BigData/UNIR/Master/9.TFM/Data
sets/Meteorologia/201201.json')
	#Se inicializa las variables year y month
	year = 2012
	month = 1
	#Para cada fichero
	for i in range(65):
		if month == 12 :
			month = 1
			year += 1
		else:
			month += 1
		fileName = str(year * 100 + month)
		jsonData =
pandas.read_json('/Users/luis/Documents/BigData/UNIR/Master/9.TFM/Data
sets/Meteorologia/' + fileName + '.json')
		#Se crea una lista con el data frame que contiene la
informacion
		#de los ficheros cargados hasta el momento y el nuevo
fichero
		dfsToConcat = [df,jsonData]
		#Se unen los dos elementos de la lista
		df = pandas.concat(dfsToConcat)
	#Se rellenan los valores nulos con ceros
	df = df.fillna(0)
	#Se formatea la fecha con el patron "DD/MM/YYYY"
	df.fecha = df.fecha.apply(lambda date: date.split('-')[2] + '/'
+ date.split('-')[1] + '/' + date.split('-')[0])
	return df
```

streetsWithMoreAccidents.py

```
from accidentsDataFrame import getAccidentsDataFrame,
getAccidentsZoneDataFrame, getVacationDays, getSchoolVacationDays
from accidentsDataFrame import getAccidentsZoneDateDataFrame,
getAccidentsZoneDateVacDataFrame,getM30Calz1Zone2DataFrame
from accidentsDataFrame import getVariablesGroupDataFrame,
getVariablesGroupYearsDataFrame
from jsonToDF import getMeteorologyDataFrame
import operator as op
import numpy
import pandas
from datetime import datetime
from csvToDataFrame import getDataFrameFromCsv
from datetime import datetime

def streetNumAccidentsSorted():
        '''Count num of accidets by street and sorts them descending'''
        #Se obtiene el data frame a partir del CSV creado
        #con los datos de los accidentes de trafico
        accDF = getAccidentsDataFrame()

        #Lista con las calles o vias
        accList = accDF['Lugar']

        streetsCounter = {}

        #Para cada elemento de la lista de calles
        #Se cuenta el numero de veces que aparece en
        #el data frame, que sera igual al numero de
        #accidentes que se dan en esa calle
        for acc in accList:
                if not streetsCounter.has_key(acc):
                        streetsCounter[acc] = 0
                streetsCounter[acc] += 1

        #Se ordena el resultado obtenido de mayor a menor
        sortedByValue = sorted(streetsCounter.items(),
key=op.itemgetter(1), reverse=True)
        return sortedByValue
```

```
def streetRankingByNumAccidentsTop20():
        '''Print a TOP 20 streets by num accidents'''
        sortedByValue = streetNumAccidentsSorted()
        for item in sortedByValue[0:20]:
                print item

def minMaxStreet(streetName, dataFrame):
        '''Return the min and max num/kms for a street suplied'''
        min = 100000000
        max = 0 #Get numbers from the street
        subDF = dataFrame.loc[dataFrame['Lugar'] == streetName]
        #Get min casting column Numero to integer
        min = numpy.min(subDF['Numero'].astype(int))
        #Get max casting column Numero to integer
        max = numpy.max(subDF['Numero'].astype(int))
        return {'min': min, 'max': max}

def topStreetAccidentsDividedByZones():
        '''Split TOP 20 streets in 5 zones'''
        numSplit = 5
        sortedByValue = streetNumAccidentsSorted()
        accDF = getAccidentsDataFrame()
        minMaxDF = pandas.DataFrame(columns=['Street', 'Min', 'Max'])
        #I split each street in 5 zones
        for street in sortedByValue[0:20]:
                minMax = minMaxStreet(street[0], accDF)
                minMaxDF.loc[len(minMaxDF.index)] = {'Street':
street[0], 'Min':minMax['min'], 'Max':minMax['max']}

        streets = pandas.DataFrame(sortedByValue[0:20],
columns=['Street', 'NumAcc'])
        top20Streets =
accDF.loc[accDF['Lugar'].isin(streets['Street'])]
        top20Streets['Zone'] = 0

        for i in range(0, len(top20Streets)):
        #for i in range(0, 10):
                minMaxSubDF =
minMaxDF.loc[minMaxDF['Street'].isin(top20Streets.Lugar[i:i+1])]
                zone = 0
                if len(minMaxSubDF) == 1:
                        num = int(top20Streets.Numero[i:i+1])
                        min = int(minMaxSubDF.Min[0:1])
                        max = int(minMaxSubDF.Max[0:1])
                        if num < max:
                                zone = int(num / ((max-min)/numSplit)) +
1
                        else:
                                zone = numSplit
```

```
                else:
                        zone = -1
                top20Streets.Zone[i:i+1] = zone
        return top20Streets

def streetAccidentsDividedByZones():
        '''Split TOP 20 streets in 5 zones'''

        #Numero de zonas en las que se va a dividir cada calle
        numSplit = 5
        sortedByValue = streetNumAccidentsSorted()
        accDF = getAccidentsDataFrame()

        #Se reemplaza los valores NA por ceros para poder obtener
        #correctamente el maximo y el minimo del campo Numero
        accDF.Numero = pandas.to_numeric(accDF.Numero).fillna(0)

        #Se crea un data frame vacio con 3 columnas en el que
        #se va a almacenar la calle, su maximo y su minimo
        minMaxDF = pandas.DataFrame(columns=['Street', 'Min', 'Max'])
        #Para cada valle
        for street in sortedByValue:
                #Se obtiene una lista con el maximo y el minimo para la
calle dada
                minMax = minMaxStreet(street[0], accDF)
                #Se anade al resultado al data frame final
                minMaxDF.loc[len(minMaxDF.index)] = {'Street':
street[0], 'Min':minMax['min'], 'Max':minMax['max']}

        #Se crea un nuevo data frame vacio con dos columnas
        streets = pandas.DataFrame(sortedByValue[0:20],
columns=['Street', 'NumAcc'])
        #Se inicializa la columna nueva Zone a 0 en todos los elementos
del data frame original
        accDF['Zone'] = 0

        for i in range(0, len(accDF)):
                #Se busca el maximo el minimo para una calle dada
                minMaxSubDF =
minMaxDF.loc[minMaxDF['Street'].isin(accDF.Lugar[i:i+1])]
                zone = 0
                #Si se ha encontrado la calle
                if len(minMaxSubDF) == 1:
                        num = int(accDF.Numero[i:i+1])
                        min = int(minMaxSubDF.Min[0:1])
                        max = int(minMaxSubDF.Max[0:1])

                        #Si la diferencia entre el maximo y el minimo
es mayor que el numero de zonas
```

```
                        if (max-min) > numSplit:
                                if num < max:
                                        zone = int(num / ((max-
min)/numSplit)) + 1
                                else:
                                        zone = numSplit
                        else:
                                zone = 1
                else:
                        zone = -1
                #Se modifica el valor de la columna Zone para ese
elemento de data frame
                accDF.Zone[i:i+1] = zone
        return accDF

def splitDateAndHour():
        #Se recupera el data frame guardado anteriormente con la zona
de cada calle
        accDF = getAccidentsZoneDataFrame()
        weekDay = []
        dates = []
        weekDayNum = []
        streetZone = []
        #Se crea un listado con equivalencias.
        #Dado el nombre de un mes, se le asigna el numero de mes
correspondiente
        monthEquivalence = {'Enero':'01',
'Febrero':'02','Marzo':'03','Abril':'04','Mayo':'05','Junio':'06','Jul
io':'07','Agosto':'08',

        'Septiembre':'09','Octubre':'10','Noviembre':'11',
'Diciembre':'12'}
        #Listado de equivalencias de dias de la semana
        weekNumEquivalence = {'Domingo':1,
'Lunes':2,'Martes':3,'Miercoles':4,'Jueves':5,'Viernes':6,'Sabado':7}
        #Para cada fila del fata frame
        for i in range(0, len(accDF)):
                #------------- Dia de la semana
                fechaSplitted = str(accDF.Fecha[i:i+1]).split(' ')
                weekDay.append(fechaSplitted[4])
                #------------- Numero del dia de la semana

        weekDayNum.append(weekNumEquivalence[fechaSplitted[4].replace('
Mi\xef\xbf\xbdrcoles','Miercoles').
                                replace('S\xef\xbf\xbdbado','Sabado')])
                #------------- Fechas
                aux = fechaSplitted[7].split('\n')
                dates.append(str(fechaSplitted[5]) + "/" +
str(monthEquivalence[fechaSplitted[6]]) + "/" + str(aux[0]))
```

```
        #Se anaden al data frame origen los elementos calculados
        accDF['WeekDayNum'] = weekDayNum
        accDF['WeekDay'] = weekDay
        accDF['Date'] = dates
        #Listado de los campos que se van a devolver del data frame
        #A los ya existentes se han anadido numero del dia de la
semana,
        #dia de la semana escrito y fecha
        fields = ['Fecha', 'RangoHorario', 'NumVictimas', 'Lugar',
'Numero', 'TipoAccidente', 'Zone', 'WeekDayNum', 'WeekDay', 'Date',]
        return accDF[fields]

def addVacationToDF(trunc):
        #Datos de accidentes de trafico
        accDF = getAccidentsZoneDateDataFrame()
        #Datos de vacaciones obtenidos del calendario laboral
        vacation = getVacationDays()
        #Datos de vacaciones obtenidos del calendario escolar
        schoolVacation = getSchoolVacationDays()
        #Datos meteorologicos
        meteorology = getMeteorologyDataFrame()
        #Esta funcion calcula los valores minimo y maximo de cada campo
        #para la informacion meteorolgica.
        meteorologyMinMax = minMaxMeteorology(meteorology)

        #Lista en la que se almacenan los datos de las vacaciones
laborales
        vactionList = []
        #Lista en la que se almacenan los datos de las vacaciones
escolares
        schoolVacList = []
        #Lista en la que se almacenan las horas normalizadas/eslacadas
        normalizedHours = []
        #Lista en la que se almacenan los datos de las precipitaciones
        normalizedPrec = []
        #Lista en la que se almacenan los datos de las rachas de viento
        normalizedRacha = []
        #Lista en la que se almacenan los datos de la temperatura
maxima
        normalizedTMax = []
        #Lista en la que se almacenan los datos de la temperatura media
        normalizedTMed = []
        #Lista en la que se almacenan los datos de la temperatura
minima
        normalizedTMin = []
        #Lista en la que se almacenan los datos de la velocidad media
        normalizedVelmedia = []

        #Se reemplaza las comas por puntos para cada columna
```

```
        meteorology.prec = meteorology.prec.apply(lambda x:
str(x).replace(',','.')).astype(float)
        meteorology.racha = meteorology.racha.apply(lambda x:
str(x).replace(',','.')).astype(float)
        meteorology.tmax = meteorology.tmax.apply(lambda x:
str(x).replace(',','.')).astype(float)
        meteorology.tmed = meteorology.tmed.apply(lambda x:
str(x).replace(',','.')).astype(float)
        meteorology.tmin = meteorology.tmin.apply(lambda x:
str(x).replace(',','.')).astype(float)
        meteorology.velmedia = meteorology.velmedia.apply(lambda x:
str(x).replace(',','.')).astype(float)

        #----------Vacaciones y datos meteorologicos
        #Para cada fecha presente en el data frame de accidentes de
trafico
        for date in accDF.Date:
                #Contiene un data frame de vacaciones laborales
filtrado por la fecha
                aux = vacation.loc[vacation.Date == date]
                #Si el data frame contiene algun elemento implica que
ese dia es festivo laboral
                if len(aux) > 0:
                        vactionList.append(1)
                else:
                        vactionList.append(0)
                #Contiene un data frame de vacaciones escolares
filtrado por la fecha
                aux = schoolVacation.loc[schoolVacation.Date == date]
                #Si el data frame contiene algun elemento implica que
ese dia es festivo escolar
                if len(aux) > 0:
                        schoolVacList.append(1)
                else:
                        schoolVacList.append(0)
                #Data frame meteorologico filtrado por la fecha
                aux = meteorology.loc[meteorology.fecha == date]
                if len(aux) > 0:
                        #Se resetan los indices del data frame
                        aux = aux.reset_index(drop=True)
                        #-------------Prec --> se escalan los valores
                        aux2 =
meteorologyMinMax.loc[meteorologyMinMax.Item == 'prec']
                        aux2 = aux2.reset_index(drop=True)
                        normalizedPrec.append(float((aux.prec -
aux2.Min)/(aux2.Max-aux2.Min)))
                        #-------------Racha --> se escalan los valores
                        aux2 =
meteorologyMinMax.loc[meteorologyMinMax.Item == 'racha']
```

```
                        aux2 = aux2.reset_index(drop=True)
                        normalizedRacha.append(float((aux.racha -
aux2.Min)/(aux2.Max-aux2.Min)))
                        #-------------Tmax --> se escalan los valores
                        aux2 =
meteorologyMinMax.loc[meteorologyMinMax.Item == 'tmax']
                        aux2 = aux2.reset_index(drop=True)
                        normalizedTMax.append(float((aux.tmax -
aux2.Min)/(aux2.Max-aux2.Min)))
                        #-------------Tmed --> se escalan los valores
                        aux2 =
meteorologyMinMax.loc[meteorologyMinMax.Item == 'tmed']
                        aux2 = aux2.reset_index(drop=True)
                        normalizedTMed.append(float((aux.tmed -
aux2.Min)/(aux2.Max-aux2.Min)))
                        #-------------Tmin --> se escalan los valores
                        aux2 =
meteorologyMinMax.loc[meteorologyMinMax.Item == 'tmin']
                        aux2 = aux2.reset_index(drop=True)
                        normalizedTMin.append(float((aux.tmin -
aux2.Min)/(aux2.Max-aux2.Min)))
                        #-------------Velmed --> se escalan los valores
                        aux2 =
meteorologyMinMax.loc[meteorologyMinMax.Item == 'velmedia']
                        aux2 = aux2.reset_index(drop=True)
                        normalizedVelmedia.append(float((aux.velmedia -
aux2.Min)/(aux2.Max-aux2.Min)))
                else:
                        normalizedPrec.append(0)
                        normalizedRacha.append(0)
                        normalizedTMax.append(0)
                        normalizedTMed.append(0)
                        normalizedTMin.append(0)
                        normalizedVelmedia.append(0)
        #-----------Se escala el campo Rango Horario
        for hour in accDF.RangoHorario:
                aux = str(hour).split(' ')
                aux = aux[1].split(':')
                normalizedHours.append(float(aux[0])/24)
        #-----------Se concatenan el nombre de la calle y la zona
        lugares = accDF.Lugar
        zones = accDF.Zone
        lugZone = zip(lugares,zones)
        result = []
        for lz in lugZone:
                result.append(lz[0] + '_' + lz[1])
        #Se crea un nuevo campo Quarter que indica el trimestre
        #al que pertenece la fecha
        accDF['Quarter'] = accDF['Date'].apply(lambda date:
```

```
                float(1) / 4 if int(date.split('/')[1]) <= 3 else
                float(2) / 4 if int(date.split('/')[1]) <= 6 else
                float(3) / 4 if int(date.split('/')[1]) <= 9 else
                float(4) / 4)
        #Se asignan los valores a los nuevos campos del data frame
principal
        accDF['Vacation'] = vactionList
        accDF['VacationSchool'] = schoolVacList
        accDF['NormHour'] = normalizedHours
        accDF['Prec'] = normalizedPrec
        accDF['Racha'] = normalizedRacha
        accDF['TMax'] = normalizedTMax
        accDF['TMed'] = normalizedTMed
        accDF['TMin'] = normalizedTMin
        accDF['VelMedia'] = normalizedVelmedia
        accDF['Result'] = result
        accDF['WeekDayNum'] = accDF['WeekDayNum'].apply(lambda day:
float(day)/7)

        #Si se quieren truncar los valores
        if trunc==True:
                accDF['Prec'] = accDF['Prec'].apply(lambda x:
'%.1f'%(x))
                accDF['Racha'] = accDF['Racha'].apply(lambda x:
'%.1f'%(x))
                accDF['TMax'] = accDF['TMax'].apply(lambda x:
'%.1f'%(x))
                accDF['TMed'] = accDF['TMed'].apply(lambda x:
'%.1f'%(x))
                accDF['TMin'] = accDF['TMin'].apply(lambda x:
'%.1f'%(x))
                accDF['VelMedia'] = accDF['VelMedia'].apply(lambda x:
'%.1f'%(x))
                accDF['NormHour'] = accDF['NormHour'].apply(lambda x:
'%.1f'%(x))


        #Se seleccionan los campos del data frame que se quieren
devolver
        fields =
['RangoHorario','Lugar','TipoAccidente','Zone','WeekDayNum','WeekDay',
'Date','Vacation','VacationSchool','NormHour',

        'Quarter','Prec','Racha','TMax','TMed','TMin','VelMedia','Resul
t']
        return accDF[fields]

def lugarZoneAccidentsSorted():
        accDF = getAccidentsZoneDateVacDataFrame()
```

```
        accList = accDF.Result

        streetsCounter = {}

        for acc in accList:
                if not streetsCounter.has_key(acc):
                        streetsCounter[acc] = 0
                streetsCounter[acc] += 1

        sortedByValue = sorted(streetsCounter.items(),
key=op.itemgetter(1), reverse=True)
        return sortedByValue

def groupDFByVariables():
        #Se obtiene el dataframe guardado previamente
        #con datos de accidentes enriquecidos con vacaciones
        #y meteorologia
        accDF = getAccidentsZoneDateVacDataFrame()
        #Lista de campos por los que se realiza la agrupacion
        header
=['WeekDayNum','Vacation','NormHour','Quarter','Prec','TMed','VelMedia
']
        #Data frame donde se almacena la agrupacion
        groupDF = pandas.DataFrame(columns=header)
        auxDF = pandas.DataFrame()
        #Para cada registro
        for i in range(len(accDF)):
                lenGroup = len(groupDF)
                lenAux = 0
                if lenGroup > 0:
                        #Se obtiene los registros que cumplan con el
filtro aplicado
                        auxDF = groupDF.loc[
                                (groupDF['WeekDayNum'] ==
accDF.loc[i,'WeekDayNum'])
                                &(groupDF['Vacation'] ==
accDF.loc[i,'Vacation'])
                                &(groupDF['NormHour'] ==
accDF.loc[i,'NormHour'])
                                & (groupDF['Quarter'] ==
accDF.loc[i,'Quarter'])
                                & (groupDF['Prec'] ==
accDF.loc[i,'Prec'])
                                & (groupDF['TMed'] ==
accDF.loc[i,'TMed'])
                                & (groupDF['VelMedia'] ==
accDF.loc[i,'VelMedia'])
                                ]
                        lenAux = len(auxDF)
```

```
                if lenAux == 0:
                        #Si no existe dicha combinacion, se crea en el
nuevo data frame
                        groupDF.loc[lenGroup,'WeekDayNum'] =
accDF.loc[i,'WeekDayNum']
                        groupDF.loc[lenGroup,'Vacation'] =
accDF.loc[i,'Vacation']
                        groupDF.loc[lenGroup,'NormHour'] =
accDF.loc[i,'NormHour']
                        groupDF.loc[lenGroup,'Quarter'] =
accDF.loc[i,'Quarter']
                        groupDF.loc[lenGroup,'Prec'] =
accDF.loc[i,'Prec']
                        groupDF.loc[lenGroup,'TMed'] =
accDF.loc[i,'TMed']
                        groupDF.loc[lenGroup,'VelMedia'] =
accDF.loc[i,'VelMedia']
        return groupDF

def groupDFByVariablesByYears():
        #Se obtiene el dataframe guardado previamente
        #con datos de accidentes enriquecidos con vacaciones
        #y meteorologia
        accDF = getAccidentsZoneDateVacDataFrame()
        #Lista de campos por los que se realiza la agrupacion
        header
=['Year','WeekDayNum','Vacation','NormHour','Quarter','Prec','TMed','V
elMedia']
        #Data frame donde se almacena la agrupacion
        groupDF = pandas.DataFrame(columns=header)

        #Se crea una nueva columna con el year
        accDF['Year'] = accDF['Date'].apply(lambda x:
datetime.strptime(x, '%d/%m/%Y').year)

        auxDF = pandas.DataFrame()
        #Para cada registro
        for i in range(len(accDF)):
                lenGroup = len(groupDF)
                lenAux = 0
                if lenGroup > 0:
                        #Se obtiene los registros que cumplan con el
filtro aplicado
                        auxDF = groupDF.loc[
                                (groupDF['Year'] == accDF.loc[i,'Year'])
                                &(groupDF['WeekDayNum'] ==
accDF.loc[i,'WeekDayNum'])
                                &(groupDF['Vacation'] ==
accDF.loc[i,'Vacation'])
```

```
                                &(groupDF['NormHour'] ==
accDF.loc[i,'NormHour'])
                                & (groupDF['Quarter'] ==
accDF.loc[i,'Quarter'])
                                & (groupDF['Prec'] ==
accDF.loc[i,'Prec'])
                                & (groupDF['TMed'] ==
accDF.loc[i,'TMed'])
                                & (groupDF['VelMedia'] ==
accDF.loc[i,'VelMedia'])
                                ]
                        lenAux = len(auxDF)
                if lenAux == 0:
                        #Si no existe dicha combinacion, se crea en el
nuevo data frame
                        groupDF.loc[lenGroup,'Year'] =
accDF.loc[i,'Year']
                        groupDF.loc[lenGroup,'WeekDayNum'] =
accDF.loc[i,'WeekDayNum']
                        groupDF.loc[lenGroup,'Vacation'] =
accDF.loc[i,'Vacation']
                        groupDF.loc[lenGroup,'NormHour'] =
accDF.loc[i,'NormHour']
                        groupDF.loc[lenGroup,'Quarter'] =
accDF.loc[i,'Quarter']
                        groupDF.loc[lenGroup,'Prec'] =
accDF.loc[i,'Prec']
                        groupDF.loc[lenGroup,'TMed'] =
accDF.loc[i,'TMed']
                        groupDF.loc[lenGroup,'VelMedia'] =
accDF.loc[i,'VelMedia']
        return groupDF

def getDFGroupByDateHour():
        accDF = getAccidentsZoneDateVacDataFrame()

        header =['Date','NormHour']
        groupDF = pandas.DataFrame(columns=header)
        auxDF = pandas.DataFrame()

        for i in range(len(accDF)):
        #for i in range(0,3):
                lenGroup = len(groupDF)
                lenAux = 0
                #print accDF.loc[i,'WeekDayNum']
                if lenGroup > 0:
                        auxDF = groupDF.loc[(groupDF['Date'] ==
accDF.loc[i,'Date'])
```

```
                                                & (groupDF['NormHour'] ==
accDF.loc[i,'NormHour'])]
                        lenAux = len(auxDF)
                #If ther is no this combination in the new FD
                if lenAux == 0:
                        groupDF.loc[lenGroup,'Date'] =
accDF.loc[i,'Date']
                        groupDF.loc[lenGroup,'NormHour'] =
accDF.loc[i,'NormHour']
        return groupDF

def dataFrameByDateHourStreetZone(streetZoneName):

        accDF = getAccidentsZoneDateVacDataFrame()
        header =
['WeekDayNum','WeekDay','Date','Vacation','VacationSchool','NormHour',
'Quarter','Prec','Racha','TMax','TMed','TMin','VelMedia']
        auxDF = pandas.DataFrame()
        auxDF2 = pandas.DataFrame()
        output = pandas.DataFrame()
        accidentExists = []

        for i in range(len(accDF)):
                lenOutput = len(output)
                lenAux = 0
                lenAux2 = 0
                if lenOutput > 0:
                        auxDF2 = output.loc[(output['Date'] ==
accDF.loc[i,'Date'])
                                                & (output['NormHour'] ==
accDF.loc[i,'NormHour'])]
                        lenAux2 = len(auxDF2)

                if lenAux2 == 0:
                        output.loc[lenOutput,'WeekDayNum'] =
accDF.loc[i,'WeekDayNum']
                        output.loc[lenOutput,'WeekDay'] =
accDF.loc[i,'WeekDay']
                        output.loc[lenOutput,'Date'] =
accDF.loc[i,'Date']
                        output.loc[lenOutput,'Vacation'] =
accDF.loc[i,'Vacation']
                        output.loc[lenOutput,'VacationSchool'] =
accDF.loc[i,'VacationSchool']
                        output.loc[lenOutput,'NormHour'] =
accDF.loc[i,'NormHour']
                        output.loc[lenOutput,'Quarter'] =
accDF.loc[i,'Quarter']
```

```
                        output.loc[lenOutput,'Prec'] =
accDF.loc[i,'Prec']
                        output.loc[lenOutput,'Racha'] =
accDF.loc[i,'Racha']
                        output.loc[lenOutput,'TMax'] =
accDF.loc[i,'TMax']
                        output.loc[lenOutput,'TMed'] =
accDF.loc[i,'TMed']
                        output.loc[lenOutput,'TMin'] =
accDF.loc[i,'TMin']
                        output.loc[lenOutput,'VelMedia'] =
accDF.loc[i,'VelMedia']

                        auxDF = accDF.loc[(accDF['Date'] ==
accDF.loc[i,'Date'])
                                        & (accDF['NormHour'] ==
accDF.loc[i,'NormHour'])
                                        & (accDF['Result'] ==
streetZoneName)]
                        lenAux = len(auxDF)
                        if lenAux > 0:
                                accidentExists.append(1)
                        else:
                                accidentExists.append(0)

        output['Result'] = accidentExists
        return output

def dataFrameOneStreetAndZone(streetZoneName):

        accDF = getAccidentsZoneDateVacDataFrame()
        m30Calz1Zone3DF = pandas.DataFrame()
        m30Calz1Zone3DF['WeekDayNum'] =
accDF['WeekDayNum']#.apply(lambda day: float(day)/7)
        m30Calz1Zone3DF['Vacation'] = accDF['Vacation']
        m30Calz1Zone3DF['VacationSchool'] = accDF['VacationSchool']
        m30Calz1Zone3DF['NormHour'] = accDF['NormHour']
        m30Calz1Zone3DF['Quarter'] = accDF['Quarter']
        m30Calz1Zone3DF['Prec'] = accDF['Prec']
        m30Calz1Zone3DF['Racha'] = accDF['Racha']
        m30Calz1Zone3DF['TMax'] = accDF['TMax']
        m30Calz1Zone3DF['TMed'] = accDF['TMed']
        m30Calz1Zone3DF['TMin'] = accDF['TMin']
        m30Calz1Zone3DF['VelMedia'] = accDF['VelMedia']
        output = []

        for result in accDF.Result:
                if result == streetZoneName:
                        output.append(1)
```

```
                else:
                        output.append(0)

        m30Calz1Zone3DF['Result'] = output
        return m30Calz1Zone3DF

def minMaxMeteorology(dataFrame):
        '''Return the min and max num/kms for a street suplied'''
        output = pandas.DataFrame(columns=['Item','Max','Min'])
        items=['']
        listItems = ['prec', 'racha', 'tmax', 'tmed', 'tmin',
'velmedia'] #racha --> racha maxima del viento
        for item in listItems:
                #Get min casting column Numero to integer
                minAux = numpy.min(dataFrame[item].apply(lambda x:
str(x).replace(',','.')).astype(float))
                #Get max casting column Numero to integer
                maxAux = numpy.max(dataFrame[item].apply(lambda x:
str(x).replace(',','.')).astype(float))
                aux = [item, maxAux,minAux]
                output.loc[len(output)] = aux

        return output

def splitDFTrainTest(filePath):
        '''Split DF into two: 3/4 --> Train, 1/4 --> Test'''

        #Se obtiene el fichero CSV que se desea separar
        dfAux = getDataFrameFromCsv(filePath)
        #       'TMax', 'TMed','TMin','VelMedia', 'Result']
        fields =
['WeekDayNum','Vacation','NormHour','Quarter','Prec','TMed','VelMedia'
,'Probability']
        #fields = ['NormHour','Prec','Probability']
        df = pandas.DataFrame(columns = fields)
        df = dfAux[fields]

        train = pandas.DataFrame(columns=fields)
        test = pandas.DataFrame(columns=fields)

        #Para cada registro fila del data frame se genera un numero
aleatorio
        #entre 1 y 4, de tal manera que si el numero obtenido toma
        #lo valores 1, 2 o 3, entonces la fila formara parte
        #del data frame de entrenamiento, mientras que si toma
        #valor 4, formara parte del data frame de test
        for i in range(0,len(df)):
                #Generate a random number between 1 and 4
                rdNum = int(numpy.random.uniform(0,4)) + 1
```

```
                #If 1,2, or 3 --> Training DF; If 4 --> Validation DF
                if rdNum < 4:
                        train.loc[len(train)] = df.loc[i].values
                else:
                        test.loc[len(test)] = df.loc[i].values

        result = {'Train': train, 'Test': test}
        return result

def getProbabilityDF(streetZoneName):
        #Se obtiene el data frame de las variables agrupadas
        #groupVariables = getVariablesGroupDataFrame()
        groupVariables =getVariablesGroupYearsDataFrame()
        #Se obtiene el data frame de los accidentes enriquecidos
        accDF = getAccidentsZoneDateVacDataFrame()
        #Se crea una nueva columna con el year
        accDF['Year'] = accDF['Date'].apply(lambda x:
str(datetime.strptime(x, '%d/%m/%Y').year))
        #Lista donde se almacenan las probabilidades
        probability = []
        #Sub data frame con los registros que coinciden
        #con la combinacion de variables
        totalCases = pandas.DataFrame()
        #Sub data frame con los registros que coinciden
        #con la combinacion de variables y con la zona
        trueCases = pandas.DataFrame()
        #Para cada combinacion de variables
        for i in range(len(groupVariables)):
                #Filtro sin la calle y zona
                totalCases = accDF.loc[
                        (accDF['Year'] == groupVariables.loc[i,'Year'])
                        & (accDF['WeekDayNum'] ==
groupVariables.loc[i,'WeekDayNum'])
                        & (accDF['Vacation'] ==
groupVariables.loc[i,'Vacation'])
                        & (accDF['NormHour'] ==
groupVariables.loc[i,'NormHour'])
                        & (accDF['Quarter'] ==
groupVariables.loc[i,'Quarter'])
                        & (accDF['Prec'] ==
groupVariables.loc[i,'Prec'])
                        & (accDF['TMed'] ==
groupVariables.loc[i,'TMed'])
                        & (accDF['NormHour'] ==
groupVariables.loc[i,'NormHour'])
                        & (accDF['VelMedia'] ==
groupVariables.loc[i,'VelMedia'])]
                #Filtro con la calle y zona
                trueCases = accDF.loc[
```

```
                        (accDF['Year'] == groupVariables.loc[i,'Year'])
                        & (accDF['WeekDayNum'] ==
groupVariables.loc[i,'WeekDayNum'])
                        & (accDF['Vacation'] ==
groupVariables.loc[i,'Vacation'])
                        & (accDF['NormHour'] ==
groupVariables.loc[i,'NormHour'])
                        & (accDF['Quarter'] ==
groupVariables.loc[i,'Quarter'])
                        & (accDF['Prec'] ==
groupVariables.loc[i,'Prec'])
                        & (accDF['TMed'] ==
groupVariables.loc[i,'TMed'])
                        & (accDF['NormHour'] ==
groupVariables.loc[i,'NormHour'])
                        & (accDF['VelMedia'] ==
groupVariables.loc[i,'VelMedia'])
                        & (accDF['Result'] == streetZoneName)]
                #Se calcula la probabilidad

        probability.append(float(len(trueCases))/float(len(totalCases))
)

        #Se crea un nuevo campo con la probabilidad de que se produzca
un accidente
        groupVariables['Probability'] = probability
        return groupVariables

#-------------------------------------------------------------------------
---------------------
#print streetRankingByNumAccidentsTop20()
#-------------------------------------------------------------------------
---------------------
#streetRankingByNumAccidentsTop20()

#Se obtiene el data frame dividido por zonas
#df = streetAccidentsDividedByZones()
#Utilizando la funcion to_csv del modulo pandas, se guarda el data
frame como fichero csv
#especificando la ruta donde se va a guardar el fichero y la
codificacion
#df.to_csv(path_or_buf='/Users/luis/Documents/BigData/UNIR/Master/9.TF
M/Code/Output.csv', encoding='latin-1')

#Se obtiene el data frame enriquecido con los datos de la fecha
#df = splitDateAndHour()
#Se guarda como fichero CSV
#df.to_csv(path_or_buf='/Users/luis/Documents/BigData/UNIR/Master/9.TF
M/Code/Output_Dates.csv')
```

```
#------------------Vacation + Meteorology
#df = addVacationToDF(True)
#df.to_csv(path_or_buf='/Users/luis/Documents/BigData/UNIR/Master/9.TF
M/Code/Output_Vacation.csv')

#aux = lugarZoneAccidentsSorted()
#for item in aux[0:20]:
#       print item

#------------------M30 Calzada 1, zona 2 DF
#dataFrameOneStreetAndZone
#df = dataFrameByDateHourStreetZone('AUTOVIA M-30 CALZADA 1 KM._2')
#df.to_csv(path_or_buf='/Users/luis/Documents/BigData/UNIR/Master/9.TF
M/Code/Output_New_M30_Calz1_Zone2.csv')

#------------------PASEO DE LA CASTELLANA NUM_2
#df = dataFrameOneStreetAndZone('PASEO DE LA CASTELLANA NUM_2')
#df.to_csv(path_or_buf='/Users/luis/Documents/BigData/UNIR/Master/9.TF
M/Code/Output_Castellana_Zone2.csv')

#------------------M30 Calzada 1, zona 2 DF, Train and Test
#dfAux =
splitDFTrainTest('/Users/luis/Documents/BigData/UNIR/Master/9.TFM/Data
sets/Accidentes Desagregado/Probability_M30_Calz1_Zone2.csv')
#trainDf = dfAux['Train']
#valDf = dfAux['Test']
#trainDf.to_csv(path_or_buf='/Users/luis/Documents/BigData/UNIR/Master
/9.TFM/Code/Train_M30_Calz1_Zone2.csv')
#valDf.to_csv(path_or_buf='/Users/luis/Documents/BigData/UNIR/Master/9
.TFM/Code/Validation_M30_Calz1_Zone2.csv')
#------------------Castellana, zona 2 DF, Train and Test
#trainDf.to_csv(path_or_buf='/Users/luis/Documents/BigData/UNIR/Master
/9.TFM/Code/Train_Castellana_Zone2.csv')
#valDf.to_csv(path_or_buf='/Users/luis/Documents/BigData/UNIR/Master/9
.TFM/Code/Validation_Castellana_Zone2.csv')

#initTime = datetime.now().time()
#endTime = datetime.now().time()
#print 'Guardado. \nInicio:' + str(initTime) + '\nFin:' + str(endTime)

#------------------Group variables
#aux = groupDFByVariables()
#aux.to_csv(path_or_buf='/Users/luis/Documents/BigData/UNIR/Master/9.T
FM/Code/Variables_Group.csv')
#------------------Group by years
#df = groupDFByVariablesByYears()
#df.to_csv(path_or_buf='/Users/luis/Documents/BigData/UNIR/Master/9.TF
M/Code/Variables_Group_By_Year.csv')
```

```
#--------------------Probabilit DF AUTOVIA M-30 CALZADA 1 KM._2
df = getProbabilityDF('AUTOVIA M-30 CALZADA 1 KM._2')
df.to_csv(path_or_buf='/Users/luis/Documents/BigData/UNIR/Master/9.TFM
/Code/Probability_M30_Calz1_Zone2_Years.csv')
```

7.2. Módulos R

LinearRegresions.R

```
#Se carga el fichero CSV con los datos que se ven a utilizar
csvTest =
read.csv('/Users/luis/Documents/BigData/UNIR/Master/9.TFM/Datasets/Acc
identes Desagregado/Probability_M30_Calz1_Zone2_Years.csv')
#Se obtienen los registros correspondientes a un year determinado
dfByYear<-csvTest[csvTest$Year=="2016",]
#Se estima el modelo realizando una regresion lineal
#La variable dependiente es Probability y las variables que siguen al
simbolo ~ son las variables predictoras
#En el segundo parametro se indica la variable que almacena los datos
regresion <-
lm(Probability~WeekDayNum+Vacation+NormHour+Quarter+Prec+TMed+VelMedia
, data = dfByYear)
#Se visualiza el resultado de la regresion
print(summary(regresion))
#Se crea un objeto grafico y se define el numero de objetos graficos
que tiene
par(mfrow=c(2,1))
#Se establece el titulo del grafico
title<-'Probabilidad de accidente: valores observados y estimados'
#Se crea el grafico con la funcion plot() mostrando los valores
observados
plot(dfByYear$Probability, main=title, xlab='Observacion', type='b',
col=1)
#Se a??aden los valores estimados por la regresi??n
lines(fitted(regresion), type='b', col=2)
#Se crea el grafico de residuos
plot(residuals(regresion), main='Residuos', xlab='Observacion',
type='b')
```

NeuralNetworking.R

```
#Se inicializa el espacio de trabajo
rm(list=ls())
#Se carga el paquete neuralnet
library(neuralnet)
#Se imprime por consola el momento en el que se inicia el
entrenamiento
print (paste('Init: ', Sys.time()))
#--------------------M30 calzada 1, zona 2
#Se carga el fichero CSV con los datos destinados al test
csvTest =
read.csv('/Users/luis/Documents/BigData/UNIR/Master/9.TFM/Datasets/Acc
identes Desagregado/Validation_M30_Calz1_Zone2.csv')
#Se carga el fichero CSV con los datos destinados al entrenamiento
csvTrain =
read.csv('/Users/luis/Documents/BigData/UNIR/Master/9.TFM/Datasets/Acc
identes Desagregado/Train_M30_Calz1_Zone2.csv')
#Se transforman a data frames
dfTest = as.data.frame(csvTest)
dfTrain = as.data.frame(csvTrain)
#Se establece la fromula que indica que la variable que se pretende
predecir
#es Probability, mientras que WeekDayNum, Vacation, NormHout, Quarter,
Prec,
#TMed y Velmedia con las variables predictoras
formula <-
as.formula("Probability~WeekDayNum+Vacation+NormHour+Quarter+Prec+TMed
+VelMedia")
#Se parametriza la funcion neuralnet con la formula, los datos de
entrenamiento,
#hidden (indica el numero de capas ocultas y el numero de neuronas de
cada capa)
#threshold (si el error toma un valor igual o inferior, el algoritmo
para),
#stepmax (numero maximo de iteraciones del algoritmo)
#algorithm (tipo de algoritmo)
#leasiningrate (valor numerico utilizado para indicar el ratio de
entrenamiento
#cuando se usar backpropagation tradicional)
```

```
#neuralNetworkTraining<-neuralnet(formula,dfTrain,hidden=c(2),
threshold = 0.01, stepmax = 1e5, algorithm = 'backprop',
 #                               learningrate = 0.00001, linear.output
= FALSE)
neuralNetworkTraining<-neuralnet(formula,dfTrain,hidden=c(3,5,3),
threshold = 0.01, stepmax = 1e5, linear.output = TRUE)
#neuralNetworkTraining<-neuralnet(formula,dfTrain,hidden=c(2,3),
threshold = 0.01, stepmax = 1e5)
#neuralNetworkTraining<-neuralnet(formula,dfTrain,hidden=c(2,3),
threshold = 0.01, stepmax = 1e5)
#Grafico con el esquema de la red neuronal
#par(mfrow=c(3,1))
plot(neuralNetworkTraining)
#Datos de test
dfTestClean <- dfTest[,c('WeekDayNum', 'Vacation', 'NormHour',
'Quarter', 'Prec', 'TMed', 'VelMedia')]
#Resultado obtenido de aplicar la red neuronal sobre los datos del
test
results <- compute(neuralNetworkTraining, dfTestClean)
#Data frame con los datos observados y los estimados
output <-
cbind(dfTestClean,dfTest[,c('Probability')],as.data.frame(results$net.
result))
names(output)[8]<-'Probability'
names(output)[names(output) == 'V1']<-'Estimated'
#Se imprime por consola la hora de finalizacion del proceso
print (paste('End: ', Sys.time()))
#Se guarda el resultado de la prediccion en un fichero CSV
write.csv(output,'/Users/luis/Documents/BigData/UNIR/Master/9.TFM/Data
sets/Accidentes
Desagregado/Output/20170907_RedNeuronalM30_Calz1_Zone2.csv')
#Se calcula el error de los valores estimados en comparacion con los
valores observados
error <- output[,'Probability']-output[,'Estimated']
#Se calcula la raiz cuadrada del erro cuadratico medio (RMSE)
rmseError <- sqrt(mean(error^2))

output2<-output[output$Probability>0,]
error2<-output2[,'Probability']-output2[,'Estimated']
rmseError2 <- sqrt(mean(error2^2))

plot(output$Probability, main='Observados vs Estimados',
xlab='Observacion', type='b', col=1)
lines(output$Estimated, type='b', col=2)
plot(error, main='Residuos', xlab='Observacion', type='b', col=1)
```