# ANÁLISIS Y CLASIFICACIÓN DE LA GRAVEDAD DE UN ACCIDENTE CON APRENDIZAJE AUTOMÁTICO

Manuel García Alfaro

Trabajo Final de Grado

Grado Universitario en Ingeniería Electrónica y Automática Industrial

Tutorizado por: David Úbeda González




Escuela Politécnica Superior De Elche

Universidad Miguel Hernández

Elche, España

Julio 2020

A mis padres, familiares y amigos por mostrarme su apoyo incondicional en los buenos y malos momentos que he pasado en este arduo camino, que comencé con ilusión y con afán por aprender y superarme como persona.

Muchas gracias.

**ABSTRACT**

Road safety is always in permanent improvement, this is because you can always reduce the number of fatalities on the roads or reduce the impact of accidents on people. That's why in this final year project we will analyse and look for patterns within the characteristics of these accidents to see what conclusions we can draw and if there is some room for improvement.

We use several techniques for the analysis of our data, we could include them as Big Data techniques, but sometimes this term is generic and confusing, since Big Data refers to the analysis of large amounts of data that can be analysed with computational techniques.

Otherwise, the fields of Artificial Intelligence such as Machine Learning and Deep Learning if they are better defined and each one is a subfield of the previous one. We can define artificial intelligence as an attempt by computers to approximate or attempt to resemble the power of reason and humans'behaviour. We will mainly focus our analysis on Machine Learning, which we can define as a set of statistical techniques and algorithms that allow us to make predictions and look for patterns in large data sets.

Therefore, we try to apply Machine Learning techniques road safety to be able to predict the severity of traffic accidents based on the characteristics in which an accident occurred.

**RESUMEN**

La seguridad vial se encuentra siempre en una mejora permanente, esto es debido a que siempre se podrá reducir el número de victimas mortales en las carreteras o reducir el impacto de los accidentes en las personas, por eso en este trabajo fin de grado nos dedicaremos a analizar y buscar patrones dentro de las características que tienen estos accidentes para ver que conclusiones podemos obtener y si existe cierto margen de mejora.

Utilizaremos varias técnicas para el análisis de nuestros datos podríamos englobarlas como técnicas de Big Data, pero a veces este término resulta genérico y confuso, ya que Big Data se refiere al análisis de grandes cantidades de datos que pueden ser analizados con técnicas convencionales de computación.

De otra manera los campos de la Inteligencia Artificial como son el Machine Learning y Deep Learning si que se encuentran mejor definidos ya que cada uno es un subcampo del anterior. Podemos definir la Inteligencia Artificial como un intento de los ordenadores de aproximarse o intentar semejar el poder de raciocinio y el comportamiento de los humanos.

Nosotros centraremos principalmente nuestro análisis en el Machine Learning, que podemos definir como un conjunto de técnicas y algoritmos de carácter estadístico que permite hacer predicciones y buscar patrones en grandes conjuntos de datos.

Por lo que intentaremos aplicar las técnicas de Machine Learning a la seguridad vial para poder predecir la gravedad de los accidentes de tráfico en función de las características en las que se produjo un accidente.

**ÍNDICE GENERAL**

**RESUMEN** ..... **3**
**ÌNDICE DE ILUSTRACIONES** ..... **7**
**ÍNDICE DE TABLAS** ..... **9**
**1. PRESENTACIÓN DEL PROBLEMA.** ..... **10**
1.1. INTRODUCCIÓN ..... 10
1.2. MOTIVACIÓN ..... 10
1.3. OBJETIVOS ..... 14
1.4. ESTRUCTURA DEL TRABAJO ..... 15
**2. ENTORNOS DE TRABAJO** ..... **16**
2.1. KAGGLE ..... 16
2.1.1. PRINCIPALES COMPONENTES ..... 17
2.2. SPYDER ..... 18
2.2.1 PRINCIPALES COMPONENTES ..... 19
2.3. KEPLER.GL ..... 20
2.3.1. PRINCIPALES COMPONENTES ..... 22
2.4. BIGML ..... 23
2.4.1. PRINCIPALES COMPONENTES ..... 24
**3. PROBLEMA EN PYTHON** ..... **26**
3.1. CONTEXTO ..... 26
3.2. ANÁLISIS DE LOS DATOS ..... 26
3.3. IMPORTAR LAS LIBRERÍAS ..... 31
3.4. CARGAR EL CONJUNTO DE DATOS ..... 32
3.5. ENTENDIENDO LOS DATOS ..... 33
3.6. DIMENSIONANDO LOS DATOS ..... 33
3.7. TIPOS DE DATOS ..... 34
3.8. ESTADÍSTICA DESCRIPTIVA ..... 35
3.9. VISUALIZACIÓN GRÁFICA ..... 37
3.10. CORRELACIÓN DE CARACTERÍSTICAS ..... 39
3.11. OBTENER VALORES FALTANTES EN LOS DATOS (MISSING VALUES) ..... 42
3.12. PREPROCESAMIENTO ..... 44

3.12.1. VARIABLE OBJETIVO ..... 44
3.12.2. ELIMINACIÓN DE CARACTERISTICAS IRRELEVANTES PARA NUESTRO MODELO ..... 46
3.12.3. MISSING VALUES ..... 47
3.12.4. CONVERSIÓN DE VARIABLES NUMERICAS ..... 49
3.12.5. CONVERSIÓN DE DATOS CATEGÓRICOS ..... 50
3.12.6. VARIABLES TEMPORALES ..... 54
3.13. SEPARACIÓN DE LOS DATOS ..... 57
3.14. DIMENSIONALIDAD DE LAS CARACTERÍSTICAS ..... 58
3.14.1. MÉTODOS DE SELECCIÓN DE CARACTERISTICAS ..... 59
3.14.2. MÉTODOS DE EXTRACCIÓN DE CARACTERÍSTICAS ..... 60
3.14.3. APLICACIÓN A NUESTRO PROYECTO ..... 61
3.15. DISTRIBUCIÓN DE CLASES ..... 64
3.16. RESOLVER PROBLEMA DE DESEQUILIBRIO ..... 65
3.17. ESCALADO DE CARACTERÍSTICAS ..... 66
3.17.1. NORMALIZACIÓN ..... 66
3.17.2. ESTANDARIZACIÓN ..... 68
3.18. SEPARACIÓN DE LOS DATOS ..... 70
3.19. APRENDIZAJE SUPERVISADO ..... 71
3.19.1 TIPOS DE ALGORITMOS SUPERVISADO ..... 71
3.20. MÉTRICAS DE RENDIMIENTO PARA ALGORITMOS DE CLASIFICACIÓN ..... 72
3.21. APLICACIÓN DE LOS ALGORITMOS DE CLASIFICACIÓN ..... 74
3.21.1. REGRESIÓN LOGÍSTICA ..... 75
3.21.2. K VECINOS MÁS CERCANOS ..... 78
3.21.3. MÁQUINAS DE VECTORES DE SOPORTE ..... 80
3.21.4. NAIVE BAYES ..... 81
3.21.5. ÁRBOLES DE DECISIÓN CLASIFICACIÓN ..... 82
3.21.6. BOSQUES ALEATORIOS CLASIFICACIÓN ..... 84
3.22. OPTIMIZACIÓN DE LOS HIPER-PARAMETROS ..... 85
3.23. APRENDIZAJE NO SUPERVISADO ..... 91
3.23.1. TIPOS DE APRENDIZAJE NO SUPERVISADO ..... 92

3.23.2 APLICACIÓN DE ALGORITMO DE AGRUPAMIENTO K-MEANS 94
3.23.3. APLICACIÓN DE ALGORITMOS DE ASOCIACIÓN ..... 99
**4. PROBLEMA EN BIGML ..... 102**
4.1. CONTEXTO ..... 102
4.2. KEPLER.GL ..... 106
4.3. PRIMEROS PASOS EN BIGML ..... 110
4.4. APRENDIZAJE SUPERVISADO EN BIGML ..... 113
4.4.1. BOSQUES ALEATORIOS CLASIFICACIÓN ..... 113
4.5. GRÁFICO DE INDEPENDENCIA PARCIAL (O PCP) ..... 119
4.6. PREDICCIÓN INDIVIDUAL ..... 122
4.7. MATRIZ DE CONFUSIÓN ..... 124
4.8. APRENDIZAJE NO SUPERVISADO EN BIGML ..... 125
4.8.1. ALGORITMO DE AGRUPACIÓN K-MEANS ..... 126
4.8.2. ALGORITMO DE ASOCIACIÓN ..... 128
**5. CONCLUSIONES GENERALES ..... 132**
5.1 TRABAJO FUTURO ..... 133
**BIBLIOGRAFÍA ..... 135**

## ÌNDICE DE ILUSTRACIONES

Ilustración 1: Vista en Kaggle. ..... 17
Ilustración 2: Vista en Spyder. ..... 19
Ilustración 3: Mapa de calor de accidents_2012_to_2014.csv. ..... 21
Ilustración 4: Modos de representación de los datos. ..... 21
Ilustración 5: Vista en BigML. ..... 24
Ilustración 6: Gráfico de barras de las características. ..... 38
Ilustración 7: Diagrama de cajas de las características. ..... 39
Ilustración 8: Mapa de correlación. ..... 41
Ilustración 9: Matriz de confusión y métricas de evaluación. ..... 73
Ilustración 10: Comparación de datos reales vs predicciones. ..... 76
Ilustración 11: División de los datos en la validación cruzada. ..... 89
Ilustración 12: Evaluación de las divisiones en la validación cruzada. ..... 89
Ilustración 13: Gráfico de barras PCA. ..... 96
Ilustración 14: Gráfico del método del codo. ..... 97
Ilustración 15: Gráfico de agrupación PCA. ..... 98
Ilustración 16: Localización de ficheros creados. ..... 104
Ilustración 17: Fichero 'X'. ..... 105
Ilustración 18: Fichero 'y'. ..... 105
Ilustración 19: Fichero 'Accidents'. ..... 105
Ilustración 20: Representación geográfica de los accidentes. ..... 106
Ilustración 21: Representación geográfica horaria 03:00am-04:00am. ..... 107
Ilustración 22: Representación geográfica horaria 17:00pm-18:00pm. ..... 108
Ilustración 23: Representación geográfica de los Lunes. ..... 109
Ilustración 24: Representación geográfica de los Domingos. ..... 110
Ilustración 25: Vista preliminar de nuestra base de datos. ..... 111
Ilustración 26: Análisis de los datos. ..... 112
Ilustración 27: Vista estadística de los datos. ..... 113
Ilustración 28: División de los datos en entrenamiento y test. ..... 114
Ilustración 29: Selección porcentaje en la división de los datos. ..... 115
Ilustración 30: Vista de los datasets generados de entrenamiento y test. ..... 115
Ilustración 31: Cómo crear un bosque aleatorio clasificación. ..... 116

Ilustración 32: Ajustar parámetros en un bosque aleatorio clasificación. .....116
Ilustración 33: Vista general de bosque aleatorio clasificación. .....117
Ilustración 34: Árbol generado en el bosque aleatorio clasificación. .....118
Ilustración 35: Cómo visualizar porcentajes de relevancia de las características en nuestro modelo.....119
Ilustración 36: Visualización porcentajes de relevancia de las características en nuestro modelo. .....119
Ilustración 37: PCP de las características más importantes de nuestro modelo. .....120
Ilustración 38: PCP con todas las características en la variable independiente. .....121
Ilustración 39: Predicción individual del modelo en Londres. .....122
Ilustración 40: Predicción individual del modelo en Wales. .....123
Ilustración 41: Cómo generar matriz de confusión del modelo. .....124
Ilustración 42: Matriz de confusión. .....125
Ilustración 43: Cómo cancelar una característica de nuestro dataset. .....126
Ilustración 44: Resultado una vez cancelada. .....126
Ilustración 45: Cómo generar un cluster. .....127
Ilustración 46: Cómo ajustar los parámetros en el cluster. .....127
Ilustración 47: Visualización del algoritmo k-means. .....128
Ilustración 48: Cómo crear un descubrimiento de asociaciones. .....129
Ilustración 49: Ajustar los parámetros del descubrimiento de asociaciones. ...129
Ilustración 50: Descubrimiento de asociaciones. .....130
Ilustración 51: Ejemplo de reglas de asociación para la localización. .....130

## ÍNDICE DE TABLAS

Tabla 1: Países con la tasa de mortalidad más alta. ..... 11
Tabla 2: Países con la tasa de mortalidad más alta. ..... 12
Tabla 3: Explicación de las variables utilizadas ..... 26
Tabla 4: Propiedades estadísticas de describe(). ..... 35
Tabla 5: Propiedades estadísticas de describe(include='all') ..... 36
Tabla 6: Nuevos valores de nuestra variable objetivo ..... 45
Tabla 7: Ejemplo variables dummies. ..... 51
Tabla 8: Resultados de las variables con mejor puntaje. ..... 63
Tabla 9: Ventajas y desventajas del Aprendizaje No Supervisado. ..... 92

# CAPÍTULO 1

# 1. PRESENTACIÓN DEL PROBLEMA.

## 1.1. INTRODUCCIÓN

Presentaremos una breve introducción sobre el contexto y la motivación principal detrás de nuestro trabajo, los objetivos perseguidos y estructurar como vamos a gestionar y canalizar toda la información de la que disponemos.

En este documento intentaremos llevar a cabo un proceso de investigación que nos servirá para la obtención del Trabajo Final de Grado que permite la obtención del título de Grado Universitario en Ingeniería Electrónica y Automática Industrial de la Universidad Miguel Hernández de Elche. Este trabajo supone 12 créditos ECTS (de los 240 créditos totales de la titulación), lo que equivale aproximadamente a 300 horas de trabajo.

## 1.2. MOTIVACIÓN

La Organización Mundial de la salud (OMS) declara que cada año mueren cerca de 1,3 millones de personas en las carreteras del mundo entero, y entre 20 y 50 millones padecen traumatismos no mortales. Los accidentes de tráfico son una de las principales causas de muerte independientemente de la edad, y la primera entre personas de entre 15 y 29 años.

De continuar la tendencia actual, en 2030 las colisiones en las vías de trafico se habrán convertido en la quinta causa más importante de muerte.

Según un informe de la OMS que se realizó en el año 2015 por países, se analizó las victimas por accidentes cada 100000 habitantes. Vamos a hacer una reflexión sobre dicho estudio.

Tabla 1: Países con la tasa de mortalidad más alta.

| PAÍS | CAUSAS | TASA DE MORTALIDAD |
|---|---|---|
| Congo | Malas condiciones de la vía, mala legislación en cuanto alcohol, uso de móvil y velocidad máxima. | Fallecen 33,2 personas por cada 100.000 conductores. |
| Liberia | No hay regulación sobre drogas, niños o uso del móvil y la velocidad máxima permitida en autopista apenas es de 72 km/h. | Fallecen 33,7 personas por cada 100.000 conductores. |
| Malawi | Los atracos son una constante dentro del país, así como la aparición de peatones y carros con animales en la calzada y un muy mal mantenimiento de los vehículos. | Fallecen 35 personas por cada 100.000 conductores. |

| | | |
|---|---|---|
| Tailandia | Grave problema en la combinación de alcohol y conducción. Ingesta de alcohol presente en 80% de los accidentes. | Fallecen 36,2 personas por cada 100.000 conductores. |
| Libia | Las condiciones meteorológicas y, sobre todo, el terrorismo. | Fallecen 73,4 personas por cada 100.000 conductores. |

Tabla 2: Países con la tasa de mortalidad más alta.

| **PAÍS** | **CAUSAS** | **TASA DE MORTALIDAD** |
|---|---|---|
| Reino Unido | Resalta por unas carreras en muy buen estado, aunque suspende en las congestiones de sus principales ciudades como veremos más adelante en nuestro estudio. | Fallecen 2,9 personas por cada 100.000 conductores. |
| Kiribiti | Tiene mucho que ver con los escasos 110.136 habitantes de su censo o con los | Fallecen 2,9 personas por cada 100.000 conductores. |

| | | |
|---|---|---|
| | apenas 811 km cuadrados de superficie, para hacernos una idea, España tiene más de 500.000 km cuadrados de territorio. | |
| Suecia | Sus duras normas sobre el uso de neumáticos de invierno y en materia de alcohol ofrecen estos resultados. | Fallecen 2,8 personas por cada 100.000 conductores. |
| Estados federados de micronesia | Igual que en el caso de Kiribiti, el bajo número de superficie y coches (tan sólo poco más de 8.000 vehículos activos a principios de la década). | Fallecen 1,9 personas por cada 100.000 conductores. |
| Mónaco | Esto sea debido a los escasos kilómetros circulables dentro del país. Ya que curiosamente su límite máximo dentro de poblado es de 70 km/h. | Fallecen 0 personas por cada 100.000 conductores. |

Se pueden sacar ciertas conclusiones de este informe, para la OMS la reducción en la tasa de mortalidad más bajas se debe sin duda a una mejor legislación sobre los riesgos que existen en seguridad vial, como el exceso de

velocidad, la conducción bajo los efectos del alcohol o el uso del cinturón de seguridad.

De los 175 países presentes en el informe 46 de ellos tienen en la actualidad leyes para regular los límites de velocidad y las tasas de alcohol permitida para conducir ya que estas suponen entre un 5% y un 35% de los accidentes en carretera en todo el mundo.

El uso del cinturón de seguridad reduce entre un 45 y un 50% el riesgo de mortalidad en los conductores y alrededor de un 25% la muerte o lesiones graves entre los ocupantes del asiento trasero.

Otro dato curioso es que 109 países de los 175 cuentan con un teléfono gratuito y de cobertura constante en todos los puntos del país, que hace que la atención profesional que reciban los heridos sea rápida y ayude a salvar vidas, el informe también hace constar que la proporción de heridos que mueren antes de llegar a un hospital es más del doble en países de ingresos bajos y medios que en países de altos ingresos.

En los países con un nivel de ingresos más elevados también se puede apreciar una mejor calidad en sus infraestructuras y en sus vehículos, debido a que éstos cuentan con frenado automático y con control de estabilidad que ayuda a reducir en gran medida el impacto de los accidentes.

El informe también recoge que el 26% de todas las muertes por accidentes de tráfico son de peatones y de ciclistas, y los motoristas y sus acompañantes representan el 28% del total de los fallecidos.

## 1.3. OBJETIVOS

El objetivo primordial de este trabajo final de grado es mediante el uso de la base de datos de accidentes en Reino Unido entre los años 2012 y 2014, calcular el grado de severidad de un accidente al aplicar los diferentes algoritmos de Aprendizaje Automático y ver con cuales se obtienen mejores predicciones o patrones que ayuden a entender mejor que provoca la gravedad de los casos.

También haremos uso de la plataforma BigML, que ofrece una solución mas intuitiva y visual para comprender nuestros resultados, y con la que trataremos los datos de una forma distinta añadiendo su respectiva localización para observar si esta es concluyente en nuestro análisis y si es posible obtener un patrón en la gravedad de los casos dependiendo de donde se produzcan estos.

## 1.4. ESTRUCTURA DEL TRABAJO

Vamos a dividir nuestro trabajo en cinco capítulos:

- Capítulo 1: Presentación del problema: Abordaremos los motivos que nos han llevado a la selección de este problema.
- Capítulo 2: Entornos de trabajo: En este capítulo vamos a hablar de los programas o interfaces que intervienen en nuestro trabajo.
- Capítulo 3: Problema en Python: Se crearán los modelos de Aprendizaje Supervisado y Aprendizaje No Supervisado en lenguaje de programación de Python mediante el programa Spyder.
- Capítulo 4: Problema en BigML: Se realizará otro enfoque más intuitivo y avanzado de los datos, junto con la representación de cómo la variable de localización nos ayuda a comprender mejor los resultados obtenidos.
- Capítulo 5: Conclusiones: Por último, se hará una valoración final del trabajo con las aportaciones obtenidas de éste y las líneas futuras a seguir.

# CAPÍTULO 2

# 2. ENTORNOS DE TRABAJO

## 2.1. KAGGLE

El primer entorno de trabajo del que vamos a hablar va a ser Kaggle, de donde además obtendremos nuestra base de datos accidents_2012_to_2014.csv para empezar a trabajar.

Kaggle es una popular plataforma donde se pueden encontrar una gran cantidad de recursos para aquellos que quieran aprender Machine Learning y ciencia de los datos. Cuenta con varias competiciones de Machine Learning con gran cantidad de premios y cientos de competidores. Los principales equipos que compiten cuentan con décadas de experiencia en el sector, abordando todo tipo de problemas de cualquier ámbito.

Kaggle tiene una finalidad didáctica, ya sea descargando bases de datos que suben los participantes y trabajando por su cuenta, o a través de sus competiciones.

Otro punto a destacar es que su comunidad siempre intenta ayudar y resolver tus dudas, podríamos decir que mejor que competiciones, en realidad se trata de proyectos de colaboración porque el objetivo principal no es necesariamente ganar, sino practicar y aprender de otros especialistas.

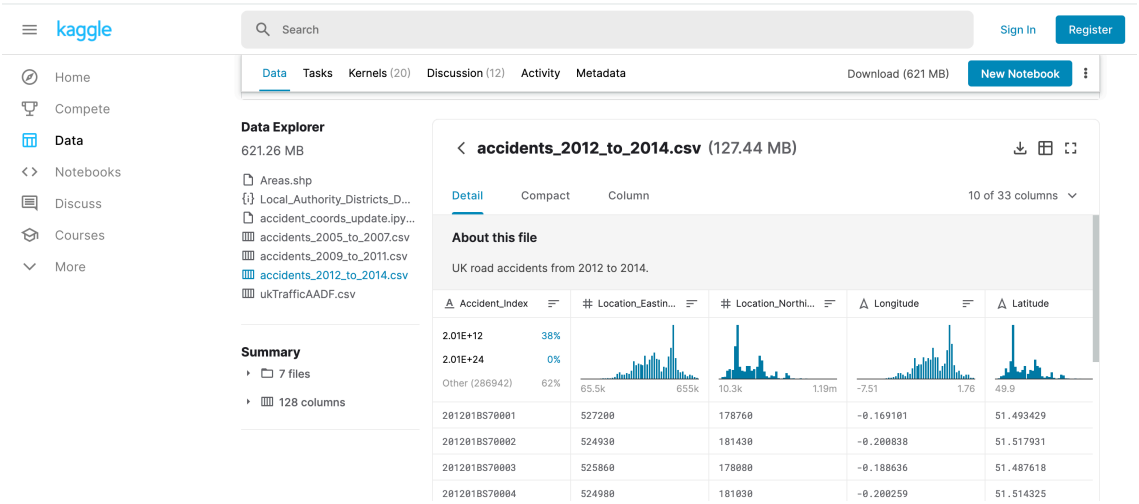



Ilustración 1: Vista en Kaggle.

### 2.1.1. PRINCIPALES COMPONENTES

**Competiciones**

La razón por la que existe Kaggle son estas pruebas de habilidad de modelado de Machine Learning, son una excelente manera de aprender las técnicas y perfeccionar tus habilidades en problemas interesantes que utilizan datos reales.

**Conjuntos de datos**

Kaggle ofrece todo tipo de bases de datos de diferentes áreas y con diferentes dimensiones para que elijamos el que más nos guste o se adecúe más a nuestro proyecto. De esta forma podemos poner en práctica los conocimientos que hemos aprendido descargando dichas bases de datos y trabajando por nuestra cuenta.

**Notebooks**

Permite la ejecución de código de Aprendizaje Automático en Python, R o Julia con Kaggle Notebooks, un entorno de programación en línea, que se ejecuta en los servidores de Kaggle, es completamente gratuito y no se tiene que instalar nada en el ordenador. Este entorno permite copiar y reconstruir los scripts de otros usuarios para así poder mejorarlos y volver a compartirlos con

la comunidad, para que comenten tus resultados y poder obtener mejores conclusiones.

**Discusiones**

Es un lugar donde se pueden realizar preguntas y obtener respuestas de los expertos en la comunidad de Kaggle.

**Cursos**

Dentro de Kaggle se ofrecen todo tipo de cursos de Machine Learning, para que los principiantes puedan progresar, son una buena forma de aprender de forma rápida y divertida para aquellos que quieran convertirse en científicos de datos.

## 2.2. SPYDER

El programa que vamos a utilizar para crear nuestro código será Spyder, donde aplicaremos todos los algoritmos de Aprendizaje Automático.

Si accedemos a la página web oficial de Spyder podemos encontrar la siguiente definición: Es un poderoso entorno científico escrito en Python, para Python, y diseñado por y para científicos, ingenieros y analistas de datos.

Ofrece una combinación única de la funcionalidad avanzada de edición, análisis, depuración y creación de perfiles de una herramienta de desarrollo integral con la exploración de datos, ejecución interactiva, inspección profunda y hermosas capacidades de visualización de un paquete científico.

Más allá de sus muchas funciones integradas, sus capacidades se pueden ampliar aún más a través de su sistema de complementos y API. Además, Spyder también se puede usar como una biblioteca de extensión PyQt5, lo que permite a los desarrolladores desarrollar su funcionalidad e integrar sus componentes como la consola interactiva, en su propio software PyQt.

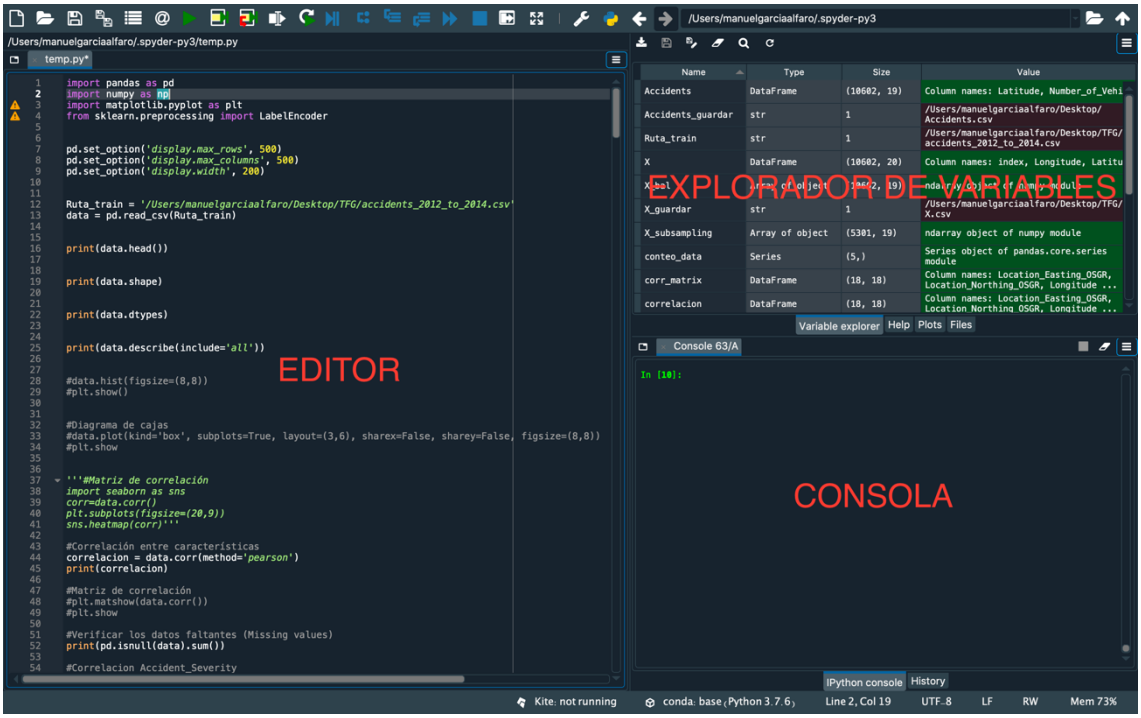



Ilustración 2: Vista en Spyder.

### 2.2.1 PRINCIPALES COMPONENTES

**Editor**

Podremos trabajar eficientemente en un editor multilingüe con un navegador de función / clase, herramientas de análisis de código, finalización automática de código, división horizontal / vertical y definición.

Editor

**Consola IPython**

Permite añadir tantas consolas IPython como se desee, dentro de la flexibilidad de una interfaz GUI completa; ejecuta tu código por línea, celda o archivo; y renderizar trazados en línea recta.

**Explorador de variables**

Permite interactuar y modificar las variables sobre la marcha, alguna de las tareas más frecuentes realizadas son las siguientes: trazar un histograma o series de tiempo, editar un marco de fecha o una matriz Numpy, ordenar una colección y profundizar en objetos anidados y otras tareas.

**Perfilador**

Encuentra y elimina cuellos de botella para desencadenar el rendimiento de su código.



**Depurador**

Permite Rastrear cada paso de la ejecución y de su código de forma interactiva.

## 2.3. KEPLER.GL

Kepler.gl es el entorno que utilizaremos para el análisis y ubicación de los puntos donde se produjeron los accidentes, nos permite filtrar características de los accidentes y ver que patrones siguen respecto a su ubicación.

Si accedemos a la página oficial de Kepler.gl, podemos encontrar la siguiente definición: Kepler.gl es una poderosa herramienta de análisis geoespacial de código abierto para conjuntos de datos a gran escala.

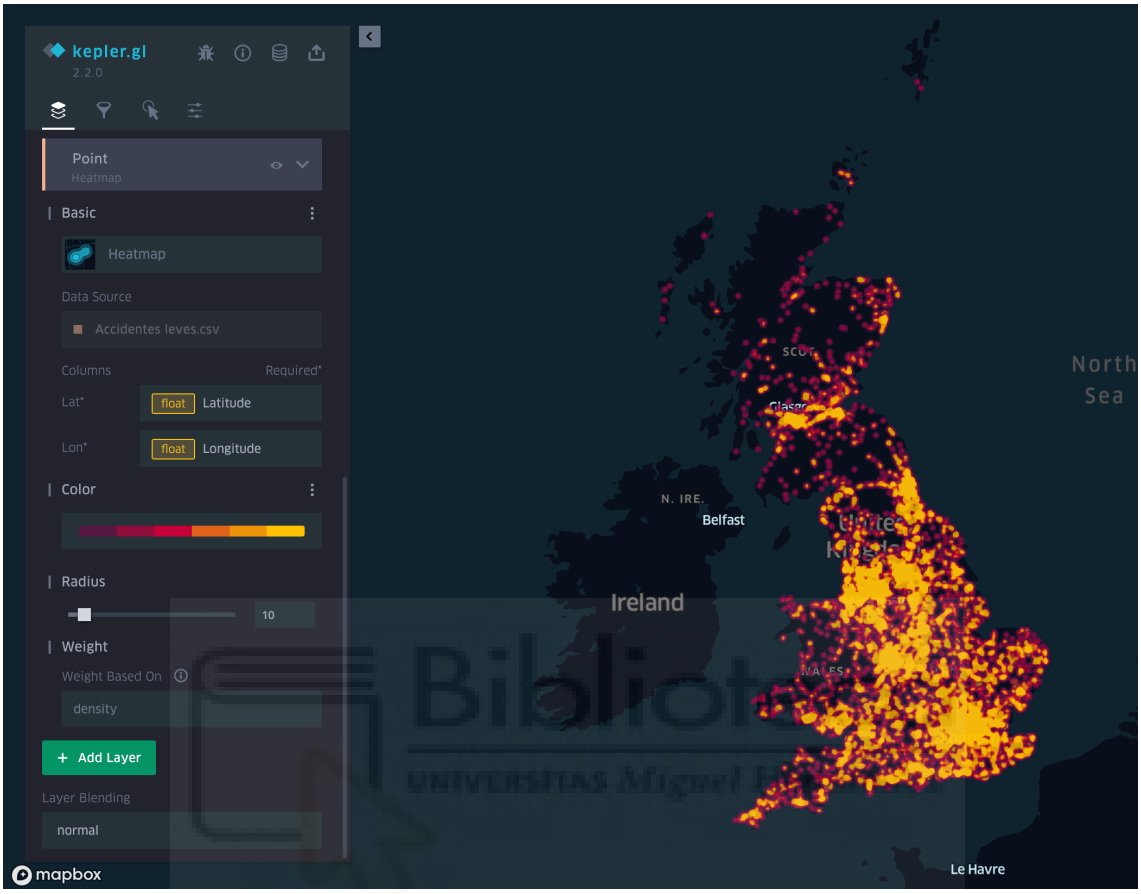



*Ilustración 3: Mapa de calor de accidents_2012_to_2014.csv.*

Kepler admite distintos formatos de trabajo como pueden ser CSV, GeoJson y Json.

También ofrece la posibilidad de hacer distintas representaciones de los puntos de localización como pueden ser las siguientes:

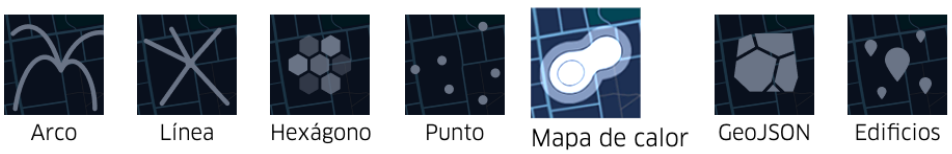



Ilustración 4: Modos de representación de los datos.

Además, se pueden configurar colores para distintas clasificaciones de una categoría para hacernos comprender mejor como funcionan nuestras bases de datos en función de su localización.

### 2.3.1. PRINCIPALES COMPONENTES

**Capas**

Una vez cargadas las bases de datos se puede crear una nueva capa en función de estos. Después podemos gestionar las opciones de añadir, personalizar, activar/desactivar, renombrar, reorganizar y también eliminar capas creadas.

**Filtros**

Se pueden filtrar las características de nuestras bases de datos y cuando se filtra una característica, nos aparecerá a su vez otro filtro con los valores que esta puede tener, para ajustar y poder así perfeccionar nuestra visualización a nuestro gusto.

**Interacciones**

En esta pestaña podemos modificar 3 tipos de configuraciones diferentes.

La primera opción permite la activación y desactivación para que se muestren las coordenadas en cualquier lugar del mapa mientras nos desplazamos.

La segunda opción que encontramos es el *tooltip*, que permite seleccionar aquellos campos que deseamos mostrar al seleccionar cualquier punto representado en el mapa.

La tercera opción que tenemos es la opción de *brush*, que permite configurar un radio de visualización de los datos a partir del elemento que seleccionemos con el puntero de nuestro ratón.

**Base del mapa**

Aquí se podrá establecer el tipo de mapa de fondo que se desee, tanto mapas con fondos claros como con fondos oscuro o incluso una imagen de mapa por satélite.

## 2.4. BIGML

Es un entorno que permite acercarnos más a los algoritmos que hemos implementado en Python, para entender mejor su comportamiento, ya que ofrece una mejor interpretación de los resultados, mucho más rápida y visual de la que obtenemos en Spyder o en entornos similares de programación.

Si accedemos a su página web, BigML se define como una plataforma integral de Machine Learning que proporciona una selección de algoritmos de Aprendizaje Automático robustos, diseñados para resolver problemas del mundo real mediante la aplicación de un marco único y estandarizado en toda su empresa. Donde se evita las dependencias de muchas bibliotecas dispares que aumentan la complejidad, los costos de mantenimiento y la deuda técnica en sus proyectos.


Ilustración 5: Vista en BigML.

### 2.4.1. PRINCIPALES COMPONENTES

**Fuentes**

Las fuentes son el primer paso de cualquier flujo de trabajo de BigML. Las características básicas que podemos encontrar en fuentes, incluyen los formatos de archivo, las opciones de carga, o las opciones avanzadas de configuración de análisis.

**Conjunto de datos**

EL conjunto de datos es el bloque de construcción fundamental para sus flujos de trabajo BigML. Permite filtrar, muestrear, agregar nuevos campos o dividir un conjunto de datos en conjuntos de datos de entrenamiento y prueba.

**Supervisado**

Incluye los algoritmos más importantes del Aprendizaje Supervisado, tanto algoritmos de clasificación como de regresión (árboles, conjuntos, regresiones lineales, regresiones logísticas, redes profundas) y pronósticos de series de tiempo.

**No supervisado**

Incluye los algoritmos más importantes del Aprendizaje No Supervisado, como son el análisis de conglomerados, la detección de anomalías, el modelado de

temas, el descubrimiento de asociaciones y el análisis de componentes principales (PCA).

**Predicciones**

En este apartado se permite hacer predicciones individuales o generar predicciones por lotes para un grupo de nuevas instancias.

# CAPITULO 3

# 3. PROBLEMA EN PYTHON

## 3.1. CONTEXTO

El gobierno del Reino Unido acumuló datos de tráfico de 2000 y 2016, registrando más de 1.6 millones de accidentes en el proceso y haciendo de este uno de los conjuntos de datos de tráfico más completos que existen. Es una imagen enorme de un país en proceso de cambio.

Tenga en cuenta que todos los datos de accidentes contenidos provienen de informes policiales, por lo que estos datos no incluyen incidentes menores. En este TFG abarcaremos un subconjunto de datos comprendidos entre los años 2012 y 2014 del total de los accidentes, debido al volumen del conjunto global y el nivel de dificultad que supondría manejar tal cantidad de datos.

## 3.2. ANÁLISIS DE LOS DATOS

Primeramente, vamos a proceder a explicar el significado de cada variable y a continuación pasaremos a ver individualmente todos los valores que pueden tomar dichas variables.

Tabla 3: Explicación de las variables utilizadas.

| VARIABLE | DESCRIPCIÓN |
|---|---|

| | |
|---|---|
| Accident_Index | Código identificador del accidente. |
| Location_Easting_OSGR | Localización mediante el sistema de referencia Ordnance Survey National Grid, es un sistema de referencias de cuadrícula geográfica utilizado en Gran Bretaña, distinto de la latitud y la longitud. |
| Location_Northing_OSGR | Localización mediante el sistema de referencia Ordnance Survey National Grid, es un sistema de referencias de cuadrícula geográfica utilizado en Gran Bretaña, distinto de la latitud y la longitud. |
| Longitud | Muestra la longitud en coordenadas GPS del punto final. |
| Latitud | Muestra la latitud en coordenadas GPS del punto final. |
| Police_Force | Fuerza policial responsable en el área del accidente, o fuerza policial asignada al accidente. |
| Accident_Severity | Indica la gravedad del accidente (Fatal, Serious, Slight). |
| Number_of_Vehiculs | Número de vehículos implicados en el accidente. |
| Number_of_Casualties | Número de víctimas. |
| Date | Fecha del accidente. |
| Day_a_week | Día de la semana. |
| Time | Hora a la que ocurrió el accidente. |
| Local_Authority_(District) | Distrito en el que ocurrió el accidente. |
| Local_Authority_(Highway) | |
| 1st_Road_Class | Tipo de vía. |
| 1st_Road_Number | Número de la vía. |

| | |
|---|---|
| Road_Type | Número de carriles de la carretera |
| Speed_limit | Límite de velocidad en la vía |
| Junction_Detail | Tipo de cruce donde se produjo el accidente. |
| Junction_Control | Tipo de regulación del cruce. |
| 2nd_Road_Class | Tipo de vía. |
| 2nd_Road_Number | Número de la vía. |
| Pedestrian_Crossing-Human_Control | Lo interpreto como si hubiera humanos controlando un paso de peatones cerca del accidente (debe estar a 50 metros ya que la opción "ninguno" lo dice).<br>Control por otra persona autorizada, Control por patrulla de cruce escolar, Ninguno dentro de los 50 metros. |
| Pedestrian_Crossing-Physical_Facilities | Tipos de cruces de peatones que hay cerca del accidente. |
| Light_Conditions | Condiciones del tipo de luz cuando se produjo el accidente: Daylight, Darkness lights lit, etc. |
| Weather_Conditions | Condiciones meteorológicas: Fine no high winds, Raining no high winds, etc. |
| Road_Surface_Conditions | Condiciones de la vía: Dry, Wet or damp, etc. |

| | |
|---|---|
| Special_Conditions_at_Site | Auto trac signal out, Oil or diesel, Roadworks, etc. |
| Carriageway_Hazards | Si ocurriera algún objeto o evento en el contexto del accidente que pudiera causar problemas, como cosas en el camino o algo que distraería a los conductores. |
| Urban_or_Rural_Area | Vía urbana o rural. |
| Did_Police_Officer_Attend_Scene_of_Accident | Si la policía atendió la escena del accidente (Yes, No). |
| LSOA_of_Accident_Location | Un área de super salida de capa inferior (LSOA) es un área geográfica. Hay un LSOA para cada código postal en Inglaterra y Gales. |
| Year | Año en el que se produjo el accidente. |

## Accident Table

**Accident Index Number** (13 chars)
Accident Year (YYYY) + Accident Ref (9 chars) *1.3*

**Police Force Code** *1.2*
For codes see page 11

**Accident Severity**
1 Fatal
2 Serious
3 Slight

**Number of Vehicles** *1.5*

**Number of Casualties** *1.6*

*Date* *1.7*
**Accident Day**

**Month**
1 January
2 February
3 March
4 April
5 May
6 June
7 July
8 August
9 September
10 October
11 November
12 December

**Year**

**Day of Week**
1 Sunday
2 Monday
3 Tuesday
4 Wednesday
5 Thursday
6 Friday
7 Saturday

*Time of Day (HHMM)* *1.9*
**Hour of Accident** (24 hour)
**Minute of Accident**

**Local Authority** (Borough/District) *1.10*
For codes see pages 12-15

*Location (OSGR)* *1.11*
**Easting**
**Northing**

**1st Road Class** *1.12*
1 Motorway
2 A(M)
3 A
4 B
5 C
6 Unclassified

**1st Road Number** *1.13*

**Road Type** *1.14*
1 Roundabout
2 One way street (from 2005)
3 Dual carriageway
6 Single carriageway
7 Slip road (from 2005)
9 Unknown
12 One way street/Slip road (1979-2004)

**Speed Limit (permanent)** *1.15*
MPH

**Junction Detail** *1.16*
0 Not at junction or within 20 metres
1 Roundabout
2 Mini-roundabout
3 T or staggered junction
5 Slip road
6 Crossroads
7 More than 4 arms (not roundabout)
8 Private drive or entrance
9 Other junction

**Junction Control** *1.17*
1 Authorised person
2 Auto traffic signal
3 Stop sign
4 Give way or uncontrolled

**2nd Road Class** *1.18*
1 Motorway
2 A(M)
3 A
4 B
5 C
6 Unclassified

**2nd Road Number** *1.19*

**Pedestrian Crossing - Human Control** *1.20A*
0 None within 50 metres
1 Control by school crossing patrol
2 Control by other authorised person

**Pedestrian Crossing - Physical Facilities** *1.20B*
0 No physical crossing facilities within 50 metres
1 Zebra
4 Pelican, puffin, toucan or similar non-junction pedestrian light crossing
5 Pedestrian phase at traffic signal junction
7 Footbridge or subway
8 Central refuge

**Light Conditions** *1.21*
1 Daylight
4 Darkness - lights lit
5 Darkness - lights unlit
6 Darkness - no lighting
7 Darkness - lighting unknown

**Weather Conditions** *1.22*
1 Fine no high winds
2 Raining no high winds
3 Snowing no high winds
4 Fine + high winds
5 Raining + high winds
6 Snowing + high winds
7 Fog or mist
8 Other
9 Unknown

**Road Surface Conditions** *1.23*
1 Dry
2 Wet or damp
3 Snow
4 Frost or ice
5 Flood over 3cm. deep
6 Oil or diesel (1999-2004)
7 Mud (1999-2004)

| | | |
|---|---|---|
| | **Special Conditions at Site** | *1.24* |
| 0 | None | |
| 1 | Auto traffic signal - out | |
| 2 | Auto signal part defective | |
| 3 | Road sign or marking defective or obscured | |
| 4 | Roadworks | |
| 5 | Road surface defective | |
| 6 | Oil or diesel (from 2005 - see A23) | |
| 7 | Mud (from 2005 - see A23) | |
| | **Carriageway Hazards** | *1.25* |
| 0 | None | |
| 1 | Vehicle load on road | |
| 2 | Other object on road | |
| 3 | Previous accident | |
| 4 | Dog on road (1979-2004) | |
| 5 | Other animal on road (1979-2004) | |
| 6 | Pedestrian in carriageway - not injured (from 2005) | |
| 7 | Any animal in carriageway (except ridden horse) (from 2005) | |
| | **Did a Police Officer Attend the Scene and Obtain the Details for this Report** (from 1999) | *1.26* |
| 1 | Yes | |
| 2 | No | |
| 3 | No - accident was reported using a self completion form | |

## 3.3. IMPORTAR LAS LIBRERÍAS

Uno de los primeros pasos en cualquier proyecto de Aprendizaje Automático es el de importar las librerías que utilizaremos. Se irán definiendo poco a poco conforme nos vayamos adentrando en el problema.

El importe de librerías se puede hacer conforme se va programando, pero lo que recomiendan la mayoría de programadores es importarlas al principio de nuestro programa para que sea más fácil conocer los módulos que se están utilizando dentro del programa y sea más legible para quien lo vaya a utilizar. En nuestro problema por el momento importaremos las siguientes librerías:

### NumPy

Es una librería de Python que ofrece un mayor soporte para realizar nuestras operaciones matemáticas de vectores y matrices garantizando una mejora en el rendimiento y acelerando la ejecución.

### Pandas

Consiste en una librería de Python que opera con datos ‘etiquetados’ para facilitar el trabajo, porque en la mayoría de casos reales se encuentran bases de datos incompletas, con datos erróneos o mal etiquetados y desordenados. Pandas garantiza las herramientas necesarias para trabajar con estos datos y

puede solventar los problemas que se encuentren, debido a que es capaz de remodelar errores que surgen al principio de trabajar con una base de datos.

**Matplotlib**

Es una librería de Python que se encarga de crear gráficas animadas e interactivas en 2D, es de bajo nivel, lo que implica que se necesitaran más líneas de código para generar los gráficos y diagramas que queramos, en contraposición a esta desventaja, ofrece una gran flexibilidad donde se puede implementar casi cualquier tipo de gráfico.

```
#IMPORTAR LAS LIBRERIAS
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

## 3.4. CARGAR EL CONJUNTO DE DATOS

Una vez tengamos las librerías importadas procedemos a descargar nuestro conjunto de datos directamente de la página de Kaggle y lo guardaremos en una carpeta específica dentro de nuestro ordenador.

A continuación, procedemos a leer los datos y convertirlos en un objeto tipo DataFrame mediante la librería importada anteriormente pandas, al tener un objeto de la librería pandas ahora este se le podrán pasar determinadas funciones pertenecientes a dicha librería para modelar los datos de la forma que queramos.

```
Ruta_train = '/Users/manuelgarciaalfaro/Desktop/TFG/accidents_2012_to_2014.csv'
data = pd.read_csv(Ruta_train)
```

Importante colocar correctamente la ruta de la ubicación donde se encuentra el archivo de lo contrario dará error, posteriormente pasamos esta ruta como argumento a la función read_csv contenida dentro de la librería de pandas, de esta forma conseguimos crear el objeto data que utilizaremos posteriormente.

## 3.5. ENTENDIENDO LOS DATOS

El siguiente procedimiento utilizado consiste en analizar los datos detenidamente para ver como estos se comportan y observar que preprocesamiento de estos realizar para ver qué modelo se va adecuar mejor a dichos datos. Para este paso utilizaremos funciones de la librería de pandas.

### Verificación de los datos

Observar los datos nos puede dar información relevante que no se puede obtener de otra forma. Esta forma de visualización nos puede dar ideas para posteriormente saber cómo tenemos que procesar y manejar nuestros datos para las distintas tareas que necesitemos realizar en nuestro proyecto.

La función head() nos sirve para para obtener una visualización de las 5 primeras filas.

```
#Entendiendo los datos
#Ver las 5 primeras filas
print(data.head())
```

Resultado una vez ejecutado el código:

```
  Accident_Index  Location_Easting_OSGR  Location_Northing_OSGR  Longitude   Latitude  Police_Force  Accident_Severity  Number_of_Vehicles  Number_of_Casualties        Date  Day_of_Week   Time  \
0  201201BS70001                 527200                  178760  -0.169101  51.493429             1                  3                   2                     1  19/01/2012            5  20:35
1  201201BS70002                 524930                  181430  -0.200838  51.517931             1                  3                   2                     1  04/01/2012            4  17:00
2  201201BS70003                 525860                  178080  -0.188636  51.487618             1                  3                   2                     1  10/01/2012            3  10:07
3  201201BS70004                 524980                  181030  -0.200259  51.514325             1                  3                   1                     1  18/01/2012            4  12:20
4  201201BS70005                 526170                  179200  -0.183773  51.497614             1                  3                   1                     1  17/01/2012            3  20:24

   Local_Authority_(District) Local_Authority_(Highway)  1st_Road_Class  1st_Road_Number          Road_Type  Speed_limit  Junction_Detail          Junction_Control  2nd_Road_Class  2nd_Road_Number  \
0                          12                 E09000020               3             308  Single carriageway           30              NaN  Automatic traffic signal               5                0
1                           1                 E09000033               4             412  Single carriageway           30              NaN   Giveway or uncontrolled               6                0
2                          12                 E09000020               3            3220        One way street           30              NaN   Giveway or uncontrolled               6                0
3                          12                 E09000020               5               0  Single carriageway           30              NaN   Giveway or uncontrolled               6                0
4                          12                 E09000020               4             325  Single carriageway           30              NaN   Giveway or uncontrolled               6                0

  Pedestrian_Crossing-Human_Control      Pedestrian_Crossing-Physical_Facilities                Light_Conditions      Weather_Conditions Road_Surface_Conditions Special_Conditions_at_Site  \
0            None within 50 metres  Pedestrian phase at traffic signal junction  Darkness: Street lights present and lit  Fine without high winds                  Dry                      None
1            None within 50 metres        No physical crossing within 50 meters  Darkness: Street lights present and lit  Fine without high winds                  Dry                      None
2            None within 50 metres           non-junction pedestrian crossing          Daylight: Street light present  Fine without high winds                  Dry                      None
3            None within 50 metres        No physical crossing within 50 meters          Daylight: Street light present  Fine without high winds                  Dry                      None
4            None within 50 metres        No physical crossing within 50 meters  Darkness: Street lights present and lit  Fine without high winds                  Dry                      None

  Carriageway_Hazards  Urban_or_Rural_Area Did_Police_Officer_Attend_Scene_of_Accident LSOA_of_Accident_Location  Year
0                None                    1                                       Yes                E01002821  2012
1                None                    1                                       Yes                E01004760  2012
2                None                    1                                       Yes                E01002893  2012
3                None                    1                                       Yes                E01002886  2012
4                None                    1                                       Yes                E01002890  2012
```

## 3.6. DIMENSIONANDO LOS DATOS

Para la realización de nuestro proyecto es de vital importancia saber la cantidad de datos que estamos manejando, tanto el número de filas como el de columnas que trataremos.

Si el número de filas es demasiado elevado, puede dificultar que algunos algoritmos tarden demasiado tiempo en cargar, de otra forma, si el número de

filas es demasiado pequeño puede que no sean suficientes para entrenar nuestro algoritmo.

Si el número de columnas es demasiado elevado igual que en las filas esto puede ocasionar problemas, puesto que puede hacer que algunos algoritmos se distraigan o disminuyan su eficiencia ocasionado por la gran dimensionalidad, no sufriremos muchos problemas siempre y cuando el número de filas sea 10 veces mayor que el número de columnas por poner un ejemplo.

Para conocer la dimensión de nuestros datos utilizaremos la propiedad shape de Pandas.

```
#Dimension de los datos
print(data.shape)
```

Resultado una vez ejecutado el código anterior:

```
(464697, 33)
```

El primer número hace referencia a las filas, mientras que el segundo a las columnas, por lo tanto, podremos decir que en nuestro conjunto de datos original tenemos 464697 filas y 33 columnas.

## 3.7. TIPOS DE DATOS

Es muy útil saber el tipo de datos que presenta cada característica, sabiendo esto podemos conocer qué datos necesitan una conversión a otros formatos, de esta forma más adelante cuando implementemos los algoritmos categóricos no obtendremos errores ya que estos funcionan principalmente con tipos de datos numéricos (int, float etc.)

Para saber el tipo de datos de nuestras categorías llamamos a la función dtypes de Pandas.

```
#Tipos de datos
print(data.dtypes)
```

Resultado una vez ejecutado el código anterior:

```
Accident_Index                                object
Location_Easting_OSGR                          int64
Location_Northing_OSGR                         int64
Longitude                                    float64
Latitude                                     float64
Police_Force                                   int64
Accident_Severity                              int64
Number_of_Vehicles                             int64
Number_of_Casualties                           int64
Date                                          object
Day_of_Week                                    int64
Time                                          object
Local_Authority_(District)                     int64
Local_Authority_(Highway)                     object
1st_Road_Class                                 int64
1st_Road_Number                                int64
Road_Type                                     object
Speed_limit                                    int64
Junction_Detail                              float64
Junction_Control                              object
2nd_Road_Class                                 int64
2nd_Road_Number                                int64
Pedestrian_Crossing-Human_Control             object
Pedestrian_Crossing-Physical_Facilities       object
Light_Conditions                              object
Weather_Conditions                            object
Road_Surface_Conditions                       object
Special_Conditions_at_Site                    object
Carriageway_Hazards                           object
Urban_or_Rural_Area                            int64
Did_Police_Officer_Attend_Scene_of_Accident    object
LSOA_of_Accident_Location                      object
Year                                           int64
dtype: object
```

## 3.8. ESTADÍSTICA DESCRIPTIVA

Hace referencia a las propiedades estadísticas de las características de nuestros datos. Para implementar esta función utilizamos describe() de nuestra librería de Pandas, las propiedades que describe esta propiedad son las siguientes:

Tabla 4: Propiedades estadísticas de describe().

| PROPIEDAD | DESCRIPCIÓN |
|---|---|
| Conteo | Cuenta el número de datos que hay en la característica. |
| Media | Hace la media de los datos de la característica. |
| Desviación estándar | Sirve para cuantificar la dispersión de nuestros datos. |
| Valor mínimo | Indica el valor mínimo de la característica. |

| | |
|---|---|
| 25% | Indica el percentil 25. |
| 50% | Indica el percentil 50. |
| 75% | Indica el percentil 75. |
| Valor máximo. | Indica el valor máximo de la característica. |

```
#Descripcion estadisticas
print(data.describe())
```

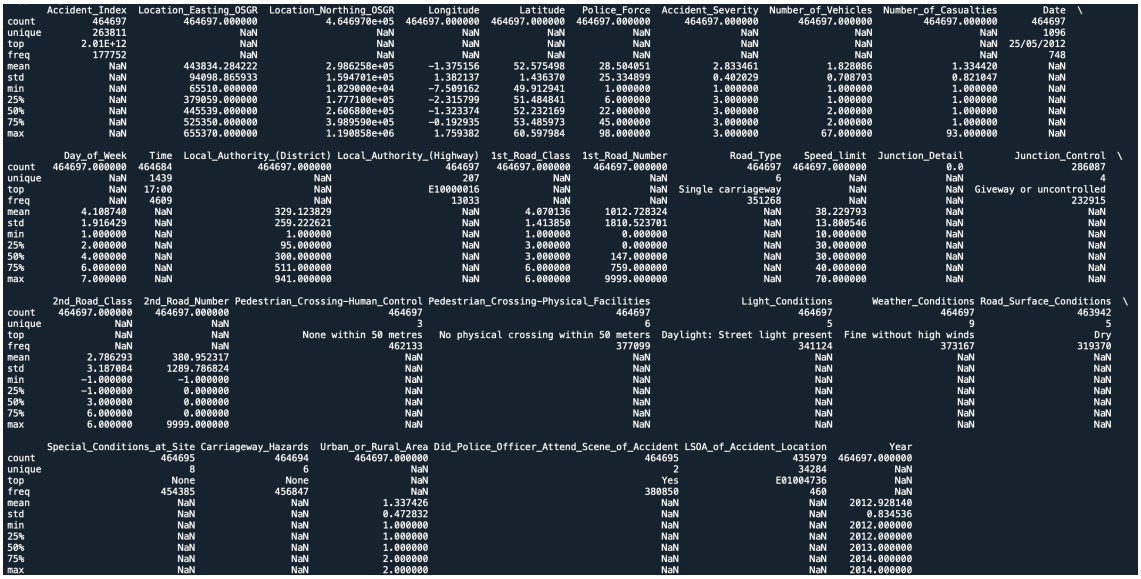
La función describe() omite filas y columnas que no contienen números, pero esta función también puede analizar datos tipos objeto para ello debemos añadir el parámetro include con el argumento 'all' a nuestra función describe(). Podemos observar que para las columnas del tipo objeto se evalúa un conjunto distinto de estadísticas:

Tabla 5: Propiedades estadísticas de describe(include='all')

| PROPIEDAD | DESCRIPCIÓN |
|---|---|
| Unique | Se refiere al número de objetos distintos en la columna |
| Top | Es el dato más frecuente que se produce en nuestra columna |
| Freq | Es la cantidad de veces que se produce Top en la columna |

```
#Descripcion estadisticas
print(data.describe(include='all'))
```

Resultado una vez ejecutado el código anterior:



| | Accident_Index | Location_Easting_OSGR | Location_Northing_OSGR | Longitude | Latitude | Police_Force | Accident_Severity | Number_of_Vehicles | Number_of_Casualties | Date |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 464697 | 464697.000000 | 464697.000000 | 464697.000000 | 464697.000000 | 464697.000000 | 464697.000000 | 464697.000000 | 464697.000000 | 464697 |
| unique | 263811 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 1096 |
| top | 2.01E+12 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 25/05/2012 |
| freq | 177752 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 748 |
| mean | NaN | 443834.284222 | 2.986258e+05 | -1.375156 | 52.575498 | 28.504051 | 2.833461 | 1.828086 | 1.334420 | NaN |
| std | NaN | 94098.865933 | 1.594701e+05 | 1.382137 | 1.436370 | 25.334099 | 0.402029 | 0.708703 | 0.821047 | NaN |
| min | NaN | 65510.000000 | 1.029000e+04 | -7.509162 | 49.912941 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | NaN |
| 25% | NaN | 379059.000000 | 1.777100e+05 | -2.315799 | 51.484841 | 6.000000 | 3.000000 | 1.000000 | 1.000000 | NaN |
| 50% | NaN | 445539.000000 | 2.606800e+05 | -1.323374 | 52.232169 | 22.000000 | 3.000000 | 2.000000 | 1.000000 | NaN |
| 75% | NaN | 525350.000000 | 3.989590e+05 | -0.192935 | 53.485973 | 45.000000 | 3.000000 | 2.000000 | 1.000000 | NaN |
| max | NaN | 655370.000000 | 1.190858e+06 | 1.759382 | 60.597984 | 98.000000 | 3.000000 | 67.000000 | 93.000000 | NaN |

| | Day_of_Week | Time | Local_Authority_(District) | Local_Authority_(Highway) | 1st_Road_Class | 1st_Road_Number | Road_Type | Speed_limit | Junction_Detail | Junction_Control |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 464697.000000 | 464684 | 464697.000000 | 464697 | 464697.000000 | 464697.000000 | 464697 | 464697.000000 | 0.0 | 286087 |
| unique | NaN | 1439 | NaN | 207 | NaN | NaN | 6 | NaN | NaN | 4 |
| top | NaN | 17:00 | NaN | E10000016 | NaN | NaN | Single carriageway | NaN | NaN | Giveway or uncontrolled |
| freq | NaN | 4609 | NaN | 13033 | NaN | NaN | 351268 | NaN | NaN | 232915 |
| mean | 4.108740 | NaN | 329.123829 | NaN | 4.070136 | 1012.728324 | NaN | 38.229793 | NaN | NaN |
| std | 1.916429 | NaN | 259.222621 | NaN | 1.413850 | 1810.523701 | NaN | 13.800546 | NaN | NaN |
| min | 1.000000 | NaN | 1.000000 | NaN | 1.000000 | 0.000000 | NaN | 10.000000 | NaN | NaN |
| 25% | 2.000000 | NaN | 95.000000 | NaN | 3.000000 | 0.000000 | NaN | 30.000000 | NaN | NaN |
| 50% | 4.000000 | NaN | 300.000000 | NaN | 3.000000 | 147.000000 | NaN | 30.000000 | NaN | NaN |
| 75% | 6.000000 | NaN | 511.000000 | NaN | 6.000000 | 759.000000 | NaN | 40.000000 | NaN | NaN |
| max | 7.000000 | NaN | 941.000000 | NaN | 6.000000 | 9999.000000 | NaN | 70.000000 | NaN | NaN |

| | 2nd_Road_Class | 2nd_Road_Number | Pedestrian_Crossing-Human_Control | Pedestrian_Crossing-Physical_Facilities | Light_Conditions | Weather_Conditions | Road_Surface_Conditions |
|---|---|---|---|---|---|---|---|
| count | 464697.000000 | 464697.000000 | 464697 | 464697 | 464697 | 464697 | 463942 |
| unique | NaN | NaN | 3 | 6 | 5 | 9 | 5 |
| top | NaN | NaN | None within 50 metres | No physical crossing within 50 meters | Daylight: Street light present | Fine without high winds | Dry |
| freq | NaN | NaN | 462133 | 377099 | 341124 | 373167 | 319370 |
| mean | 2.786293 | 380.952317 | NaN | NaN | NaN | NaN | NaN |
| std | 3.187084 | 1289.786824 | NaN | NaN | NaN | NaN | NaN |
| min | -1.000000 | -1.000000 | NaN | NaN | NaN | NaN | NaN |
| 25% | -1.000000 | 0.000000 | NaN | NaN | NaN | NaN | NaN |
| 50% | 3.000000 | 0.000000 | NaN | NaN | NaN | NaN | NaN |
| 75% | 6.000000 | 0.000000 | NaN | NaN | NaN | NaN | NaN |
| max | 6.000000 | 9999.000000 | NaN | NaN | NaN | NaN | NaN |

| | Special_Conditions_at_Site | Carriageway_Hazards | Urban_or_Rural_Area | Did_Police_Officer_Attend_Scene_of_Accident | LSOA_of_Accident_Location | Year |
|---|---|---|---|---|---|---|
| count | 464695 | 464694 | 464697.000000 | 464695 | 435979 | 464697.000000 |
| unique | 8 | 6 | NaN | 2 | 34284 | NaN |
| top | None | None | NaN | Yes | E01004736 | NaN |
| freq | 454385 | 456847 | NaN | 380850 | 460 | NaN |
| mean | NaN | NaN | 1.337426 | NaN | NaN | 2012.928140 |
| std | NaN | NaN | 0.472832 | NaN | NaN | 0.834536 |
| min | NaN | NaN | 1.000000 | NaN | NaN | 2012.000000 |
| 25% | NaN | NaN | 1.000000 | NaN | NaN | 2012.000000 |
| 50% | NaN | NaN | 1.000000 | NaN | NaN | 2013.000000 |
| 75% | NaN | NaN | 2.000000 | NaN | NaN | 2014.000000 |
| max | NaN | NaN | 2.000000 | NaN | NaN | 2014.000000 |

## 3.9. VISUALIZACIÓN GRÁFICA

Vamos a graficar nuestros datos con una sola variable, de esta forma intentaremos comprender mejor cómo se comportan el conjunto de nuestros datos de una forma visual, rápida y más comprensible para quien quiera analizar nuestra base de datos. Utilizaremos para graficarlos los siguientes métodos:

### Histograma

Este método es utilizado para tener un concepto de cómo se distribuyen nuestros atributos. Los histogramas sirven para detectar si un atributo tiene una distribución gaussiana, si está sesgado, o si tiene una distribución exponencial, también son útiles para detectar valores atípicos.

Para mostrar los histogramas de nuestra base de datos utilizaremos la función hist(). El parámetro figsize hace referencia a la dimensión que tendrá nuestro conjunto de histogramas, en nuestro caso hemos definido 8x8cm.

```
#Histograma
data.hist(figsize=(8, 8))
plt.show()
```

Resultado una vez ejecutado el código anterior:

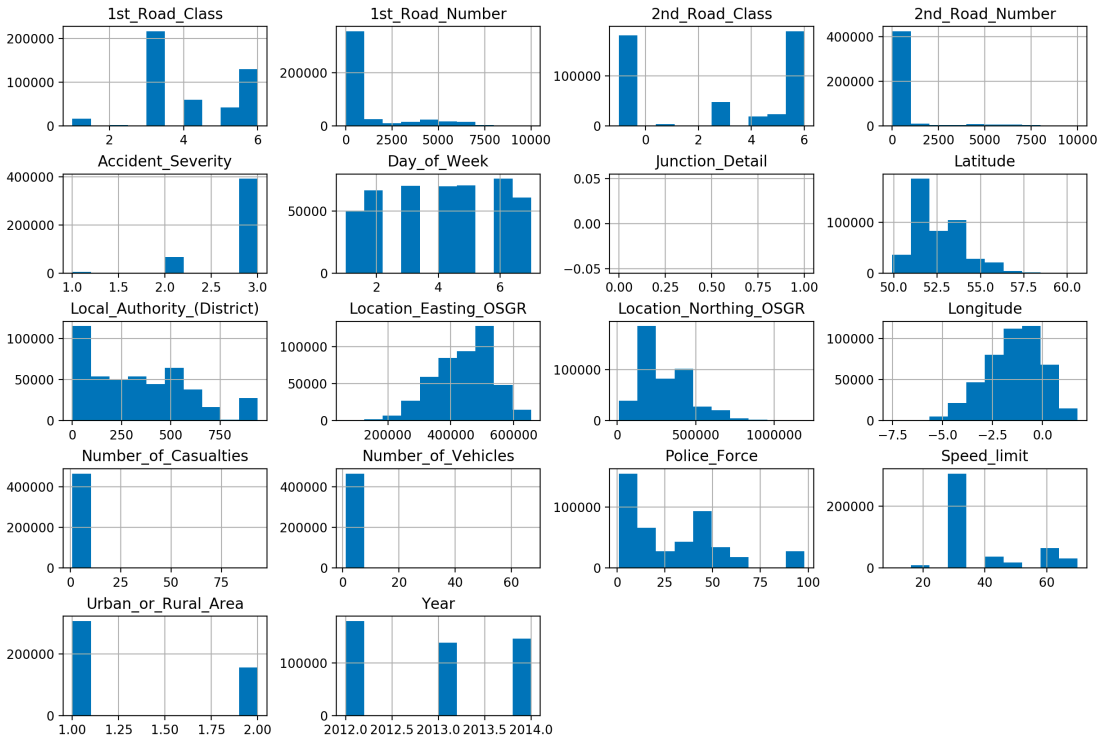



Ilustración 6: Gráfico de barras de las características.

Como podemos observar en los resultados de nuestro histograma solo está analizando los datos tipo int o float. Es decir, los datos tipo numéricos.

### Diagramas de cajas o Box Plot

Otro método para mostrar la distribución de nuestros atributos es utilizar los diagramas de cajas. Consisten en trazar una línea para la mediana y un cuadrado delimitado por los percentiles 25 y 75. Los parámetros más destacados aquí serán figsize (explicado anteriormente) y layout que nos indica el número de filas y de columnas que queremos para los histogramas, en nuestro caso el 3 indica el número de filas y el 6 el de columnas.

```
#Diagrama de cajas
data.plot(kind='box', subplots=True, layout=(3,6), sharex=False, sharey=False, figsize=(8,8))
plt.show
```

Resultado una vez ejecutado el código anterior:

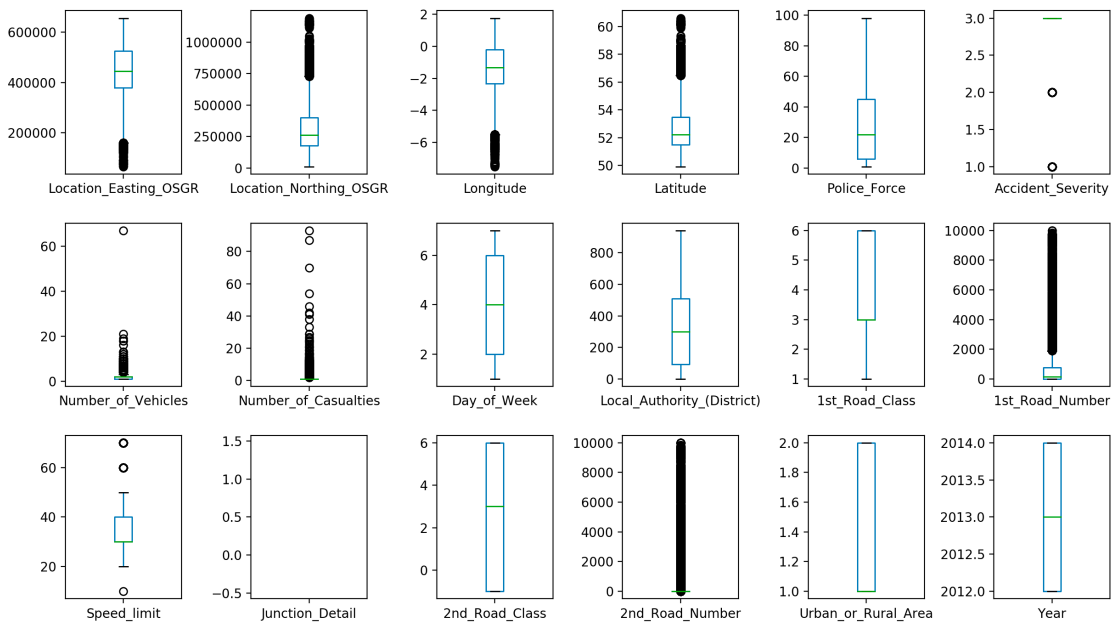



Ilustración 7: Diagrama de cajas de las características.

## 3.10. CORRELACIÓN DE CARACTERÍSTICAS

La correlación hace referencia a la relación que existe entre dos variables y cómo estas pueden varían al alterar una de ellas. En nuestro caso vamos a implementar el coeficiente de correlación de Pearson, este método asume que nuestros datos tienen una distribución normal. El intervalo del coeficiente de correlación es [-1, 1], indicando -1 una correlación negativa y 1 una correlación positiva, por el contrario, si nuestros valores se aproximan al 0 indican que no existe ninguna relación entre las variables. La función implementada para el cálculo de la correlación de Pearson es corr().

```
#Correlación entre características
correlacion = data.corr(method='pearson')
print(correlacion)
```

Resultado una vez ejecutado el código anterior:

| | Location_Easting_OSGR | Location_Northing_OSGR | Longitude | Latitude | Police_Force | Accident_Severity | Number_of_Vehicles | Number_of_Casualties | Day_of_Week |
|---|---|---|---|---|---|---|---|---|---|
| Location_Easting_OSGR | 1.000000 | -0.443488 | 0.999394 | -0.445512 | -0.341086 | 0.033165 | 0.014412 | -0.042245 | -0.002260 |
| Location_Northing_OSGR | -0.443488 | 1.000000 | -0.452043 | 0.999974 | 0.153230 | -0.031794 | -0.032548 | 0.034086 | 0.006142 |
| Longitude | 0.999394 | -0.452043 | 1.000000 | -0.453969 | -0.353312 | 0.033144 | 0.015349 | -0.040972 | -0.002335 |
| Latitude | -0.445512 | 0.999974 | -0.453969 | 1.000000 | 0.151249 | -0.031863 | -0.032358 | 0.034239 | 0.006144 |
| Police_Force | -0.341086 | 0.153230 | -0.353312 | 0.151249 | 1.000000 | -0.039979 | -0.005590 | 0.012450 | -0.001216 |
| Accident_Severity | 0.033165 | -0.031794 | 0.033144 | -0.031863 | -0.039979 | 1.000000 | 0.078750 | -0.060358 | 0.003478 |
| Number_of_Vehicles | 0.014412 | -0.032548 | 0.015349 | -0.032358 | -0.005590 | 0.078750 | 1.000000 | 0.252533 | 0.003376 |
| Number_of_Casualties | -0.042245 | 0.034086 | -0.040972 | 0.034239 | 0.012450 | -0.060358 | 0.252533 | 1.000000 | 0.000407 |
| Day_of_Week | -0.002260 | 0.006142 | -0.002335 | 0.006144 | -0.001216 | 0.003478 | 0.003376 | 0.000407 | 1.000000 |
| Local_Authority_(District) | -0.365255 | 0.114367 | -0.374428 | 0.112716 | 0.982313 | -0.043415 | -0.001769 | 0.019785 | -0.002014 |
| 1st_Road_Class | -0.059309 | 0.053008 | -0.058011 | 0.053068 | 0.046606 | -0.000410 | -0.147088 | -0.082932 | 0.004594 |
| 1st_Road_Number | -0.099017 | 0.055043 | -0.096585 | 0.055946 | 0.007602 | -0.007502 | -0.008277 | 0.006743 | 0.003172 |
| Speed_limit | -0.069761 | 0.064186 | -0.069084 | 0.063912 | 0.204729 | -0.073219 | 0.089665 | 0.141020 | -0.015876 |
| Junction_Detail | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 2nd_Road_Class | 0.057491 | -0.047574 | 0.057676 | -0.047352 | -0.123164 | 0.059245 | 0.064620 | -0.030371 | 0.011538 |
| 2nd_Road_Number | -0.024555 | 0.050909 | -0.022418 | 0.051648 | -0.022331 | 0.022077 | 0.022293 | 0.003012 | -0.000495 |
| Urban_or_Rural_Area | -0.108093 | 0.063538 | -0.106236 | 0.063078 | 0.248518 | -0.079522 | 0.044852 | 0.114752 | -0.018107 |
| Year | 0.028422 | -0.049919 | 0.028124 | -0.050133 | 0.070912 | 0.006445 | 0.005481 | -0.009160 | -0.002890 |

| | Local_Authority_(District) | 1st_Road_Class | 1st_Road_Number | Speed_limit | Junction_Detail | 2nd_Road_Class | 2nd_Road_Number | Urban_or_Rural_Area | Year |
|---|---|---|---|---|---|---|---|---|---|
| Location_Easting_OSGR | -0.365255 | -0.059309 | -0.099017 | -0.069761 | NaN | 0.057491 | -0.024555 | -0.108093 | 0.028422 |
| Location_Northing_OSGR | 0.114367 | 0.053008 | 0.055043 | 0.064186 | NaN | -0.047574 | 0.050909 | 0.063538 | -0.049919 |
| Longitude | -0.374428 | -0.058011 | -0.096585 | -0.069084 | NaN | 0.057676 | -0.022418 | -0.106236 | 0.028124 |
| Latitude | 0.112716 | 0.053068 | 0.055946 | 0.063912 | NaN | -0.047352 | 0.051648 | 0.063078 | -0.050133 |
| Police_Force | 0.982313 | 0.046606 | 0.007602 | 0.204729 | NaN | -0.123164 | -0.022331 | 0.248518 | 0.070912 |
| Accident_Severity | -0.043415 | -0.000410 | -0.007502 | -0.073219 | NaN | 0.059245 | 0.022077 | -0.079522 | 0.006445 |
| Number_of_Vehicles | -0.001769 | -0.147088 | -0.008277 | 0.089665 | NaN | 0.064620 | 0.022293 | 0.044852 | 0.005481 |
| Number_of_Casualties | 0.019785 | -0.082932 | 0.006743 | 0.141020 | NaN | -0.030371 | 0.003012 | 0.114752 | -0.009160 |
| Day_of_Week | -0.002014 | 0.004594 | 0.003172 | -0.015876 | NaN | 0.011538 | -0.000495 | -0.018107 | -0.002890 |
| Local_Authority_(District) | 1.000000 | 0.056267 | 0.019247 | 0.212592 | NaN | -0.124897 | -0.018911 | 0.263910 | 0.071632 |
| 1st_Road_Class | 0.056267 | 1.000000 | -0.083800 | -0.380226 | NaN | 0.098219 | -0.030052 | -0.213495 | 0.003845 |
| 1st_Road_Number | 0.019247 | -0.083800 | 1.000000 | -0.067390 | NaN | 0.031523 | 0.186657 | -0.018386 | -0.004414 |
| Speed_limit | 0.212592 | -0.380226 | -0.067390 | 1.000000 | NaN | -0.336064 | -0.047418 | 0.680879 | 0.008236 |
| Junction_Detail | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 2nd_Road_Class | -0.124897 | 0.098219 | 0.031523 | -0.336064 | NaN | 1.000000 | 0.148090 | -0.291616 | -0.002317 |
| 2nd_Road_Number | -0.018911 | -0.030052 | 0.186657 | -0.047418 | NaN | 0.148090 | 1.000000 | -0.039443 | -0.005371 |
| Urban_or_Rural_Area | 0.263910 | -0.213495 | -0.018386 | 0.680879 | NaN | -0.291616 | -0.039443 | 1.000000 | 0.017433 |
| Year | 0.071632 | 0.003845 | -0.004414 | 0.008236 | NaN | -0.002317 | -0.005371 | 0.017433 | 1.000000 |

La tabla anterior se puede visualizar más rápidamente y de forma más didáctica utilizando la librería Seaborn, es una librería de visualización de datos. Proporciona una interfaz de alto nivel para dibujar gráficos estadísticos atractivos e informativos.

```
#Matriz de correlación
import seaborn as sns
corr=data.corr()
plt.subplots(figsize=(20,9))
sns.heatmap(corr)
```

Resultado una vez ejecutado el código anterior:

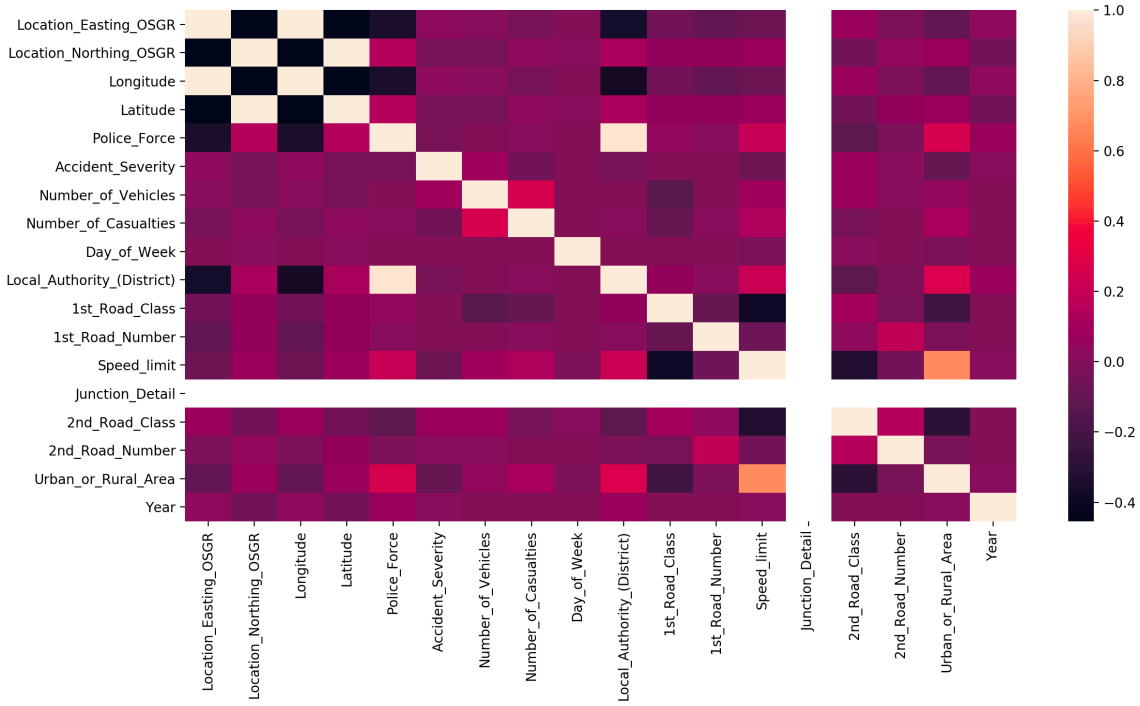



Ilustración 8: Mapa de correlación.

Si nos fijamos en los resultados observamos que la parte inferior de la matriz es simétrica a la parte superior de esta, también podemos observar que en dicha matriz se produce un espacio en blanco que corresponde a la característica Junction_Detail, que al igual que ocurría en el histograma y en el diagrama de cajas, esta variable no aporta ninguna información por lo que más adelante procederemos a su eliminación, por eso es importante este tipo de visualizaciones para darnos cuenta de qué variables, nos aportan información relevante y cuáles no.

Ahora bien, para visualizar sólo la correlación de un solo atributo con los demás atributos, sin que resulte tan tedioso estar mirando detenidamente la tabla anteriormente podemos utilizar el siguiente método:

```
#Correlacion Accident_Severity
corr_matrix = data.corr()
print(corr_matrix['Accident_Severity'].sort_values(ascending=False))
```

Resultado una vez ejecutado el código anterior:

```
Accident_Severity              1.000000
Number_of_Vehicles             0.078750
2nd_Road_Class                 0.059245
Location_Easting_OSGR          0.033165
Longitude                      0.033144
2nd_Road_Number                0.022077
Year                           0.006445
Day_of_Week                    0.003478
1st_Road_Class                -0.000410
1st_Road_Number               -0.007502
Location_Northing_OSGR        -0.031794
Latitude                      -0.031863
Police_Force                  -0.039979
Local_Authority_(District)    -0.043415
Number_of_Casualties          -0.060358
Speed_limit                   -0.073219
Urban_or_Rural_Area           -0.079522
Junction_Detail                     NaN
Name: Accident_Severity, dtype: float64
```

Observamos como muestra la correlación de la característica Accident_Severity con las demás características.

## 3.11. OBTENER VALORES FALTANTES EN LOS DATOS (MISSING VALUES)

Es muy importante determinar el número de datos faltantes en cada una de nuestras características, debido a que nuestros algoritmos de clasificación se comportaran de una forma bien distinta si el número de datos faltantes es muy elevado, para saber el número de datos faltantes llamaremos a la función isnull().

```
#Verificar los datos faltantes (Missing values)
print(pd.isnull(data).sum())
```

Resultado una vez ejecutado el código anterior:

```
Accident_Index                                  0
Location_Easting_OSGR                           0
Location_Northing_OSGR                          0
Longitude                                       0
Latitude                                        0
Police_Force                                    0
Accident_Severity                               0
Number_of_Vehicles                              0
Number_of_Casualties                            0
Date                                            0
Day_of_Week                                     0
Time                                           13
Local_Authority_(District)                      0
Local_Authority_(Highway)                       0
1st_Road_Class                                  0
1st_Road_Number                                 0
Road_Type                                       0
Speed_limit                                     0
Junction_Detail                            464697
Junction_Control                           178610
2nd_Road_Class                                  0
2nd_Road_Number                                 0
Pedestrian_Crossing-Human_Control               0
Pedestrian_Crossing-Physical_Facilities         0
Light_Conditions                                0
Weather_Conditions                              0
Road_Surface_Conditions                       755
Special_Conditions_at_Site                      2
Carriageway_Hazards                             3
Urban_or_Rural_Area                             0
Did_Police_Officer_Attend_Scene_of_Accident     2
LSOA_of_Accident_Location                   28718
Year                                            0
dtype: int64
```

Se puede observar cómo las características con más valores perdidos (missing values) pertenecen a las siguientes columnas:

- Junction_Detail: El 100% de los datos son valores perdidos.
- 2nd_Road_Class: El 39,12% de los datos de esta variable están asociados con valores perdidos, pero han tenido que ser calculados de otra forma, ya que vemos que no aparecen en la tabla. Esto es debido a que esta variable clasifica los valores perdidos como -1, por lo que hemos tenido que llamar a la función groupby() para determinar el número total de -1 que había en nuestra variable.

```
conteo_data=data.groupby('2nd_Road_Class').size()
print(conteo_data)
```

Resultado una vez ejecutado el código anterior:

```
2nd_Road_Class
-1    181794
 1      3132
 2       346
 3     47690
 4     18911
 5     22906
 6    189918
dtype: int64
```

- Junction_Control: El 38,43% de los datos son valores perdidos.
- LSOA_of_Accident_Location: El 6,18% de los datos son valores perdidos.
- Road_Surface_Conditions: El 1,62% de los datos son valores perdidos.

- Time: El 0,0027% de los datos son valores perdidos.
- Carriageway_Hazards: El 0,00064% de los datos son valores perdidos.
- Special_Conditions_at_Site: El 0,00043% de los datos son valores perdidos.
- Did_Police_Officer_Attend_Scene_of_Accident: El 0,00043% de los datos son valores perdidos.

Es importante hacernos una idea general del total de datos perdidos que tenemos en nuestro dataset para poder tratarlos en el prepocesamiento de una u otra forma.

## 3.12. PREPROCESAMIENTO

La cantidad y la calidad de información útil que contienen nuestros datos es un factor clave que determina lo bien que puede aprender un algoritmo de aprendizaje. Por este motivo, es totalmente imprescindible que nos aseguremos de escrutar y prepocesar nuestro conjunto de datos antes de implementar nuestros algoritmos de aprendizaje. Nuestra forma de preprocesar dichos datos se hace en función de la visualización que hemos realizado anteriormente de estos.

### 3.12.1. VARIABLE OBJETIVO

Vamos a comenzar el prepocesamiento de los datos tratando nuestra variable objetivo, en nuestro caso corresponde Accident_Severity que indica el grado de gravedad de un accidente y tiene los siguientes valores ya mencionados:

- Fatal = 1
- Serious = 2
- Slight = 3

Al ser 3 clases distintas convierte nuestro problema en un problema multiclase, donde ciertos algoritmos de clasificación se comportan de peor forma que si nuestro problema fuera dicotómico, es decir un problema donde solo existen

dos clases. A este tipo de problemas se le conoce como problema de clasificación binaria. Para solventar dicho problema reemplazaremos los valores de la clasificación Fatal y Serious por 1, por otro lado, la clasificación de accidentes Slight la sustituiremos por 0.

Tabla 6: Nuevos valores de nuestra variable objetivo.

| VALOR | CÓDIGO ANTERIOR | CÓDIGO ACTUALIZADO |
|---|---|---|
| Fatal | 1 | 1 |
| Serious | 2 | 1 |
| Slight | 3 | 0 |

Para realizar esta tarea utilizaremos la función replace(), pasándole como argumentos los valores los cuales vamos a reemplazar y a continuación el valor por el cual van a ser reemplazados. El código quedaría de la siguiente manera:

```
#Reemplazar valores clase objetivo
data['Accident_Severity'].replace([1,2,3],[1,1,0], inplace=True)
print(data['Accident_Severity'])
```

Para comprobar que efectivamente los resultados se han reemplazado correctamente llamaremos a la columna Accident_Severity.

```
0         3
1         3
2         3
3         3
4         3
         ..
464692    2
464693    3
464694    3
464695    2
464696    3
Name: Accident_Severity, Length: 464697, dtype: int64
```

Antes de utilizar la función replace()

```
0         0
1         0
2         0
3         0
4         0
         ..
464692    1
464693    0
464694    0
464695    1
464696    0
Name: Accident_Severity, Length: 464697, dtype: int64
```

Después de utilizar la función replace().

Mostramos las primeras 5 filas y las 5 últimas y observamos como los valores fueron reemplazados, por lo que podemos determinar que los valores fueron reemplazados correctamente.

### 3.12.2. ELIMINACIÓN DE CARACTERÍSTICAS IRRELEVANTES PARA NUESTRO MODELO

En este apartado procederemos a la eliminación de columnas que consideremos que no tienen relevancia para predecir nuestro modelo.

**Variables de localización**

Location_Easting_OSGR, Location_Northing_OSGR, Longitude, Latitude, LSOA_of_Accident_Location.

Vamos a prescindir de todas las variables de localización, dado que en principio consideraremos que estas variables no tienen mayor relevancia a la hora de determinar la gravedad de nuestro accidente.

**Numeración de carreteras**

1st_Road_Number, 2nd_Road_Number.

A priori el número de la carretera donde se produjo el accidente tampoco nos va a influir en nuestro proyecto, es similar al de la localización por lo cual prescindiremos de él.

**Variables de atendimiento del accidente**

Police_Force, Did_Police_Officer_Attend_Scene_of_Accident.

Eliminaremos las variables del tipo atendimiento del accidente, porque consideramos que es un hecho futuro a este, por lo que tampoco tendrá trascendencia en la gravedad de dicho accidente.

**Variable índice del accidente**

Accident_Index.

Eliminaremos la variable del índice del accidente, está solo serviría si fuera secuencial y no tuviéramos información de fechas, porque nos serviría para ubicar mas o menos cuando se produjo el accidente.

**Variable número de víctimas**

Number_of_Casualties.

El número de víctimas del accidente es una de las variables que eliminaremos esto se debe a que esta variable está directamente relacionada con la variable objetivo, y además se supone que este dato no lo tendrás, dado que es un dato posterior al accidente por lo que no debería estar integrado en nuestro modelo.

La función que utilizaremos para la eliminación de dichas columnas es drop(), pasándole como argumento el nombre de la columna a eliminar.

```
#Eliminacion de columnas irrelevantes

#Localizacion
data=data.drop(['Location_Easting_OSGR','Location_Northing_OSGR','Longitude','Latitude','LSOA_of_Accident_Location'], axis=1)
#Numeración de carreteras
data=data.drop(['1st_Road_Number','2nd_Road_Number'], axis=1)
#Atendió el escenario
data=data.drop(['Police_Force','Did_Police_Officer_Attend_Scene_of_Accident'], axis=1)
#Autoridad local donde se produjo el accidente
data=data.drop(['Local_Authority_(District)','Local_Authority_(Highway)'], axis=1)
#Índece del accidente
data=data.drop(['Accident_Index'], axis=1)
#Número de víctimas
data=data.drop(['Number_of_Casualties'], axis=1)
```

Resultado una vez ejecutado el código anterior:

```
   Accident_Severity  Number_of_Vehicles        Date  Day_of_Week   Time  1st_Road_Class           Road_Type  Speed_limit  Junction_Detail          Junction_Control  2nd_Road_Class  \
0                  1                   2  19/01/2012            5  20:35               3  Single carriageway           30              NaN  Automatic traffic signal               5
1                  1                   2  04/01/2012            4  17:00               4  Single carriageway           30              NaN   Giveway or uncontrolled               6
2                  1                   2  10/01/2012            3  10:07               3      One way street           30              NaN   Giveway or uncontrolled               6
3                  1                   1  18/01/2012            4  12:20               5  Single carriageway           30              NaN   Giveway or uncontrolled               6
4                  1                   1  17/01/2012            3  20:24               4  Single carriageway           30              NaN   Giveway or uncontrolled               6

  Pedestrian_Crossing-Human_Control      Pedestrian_Crossing-Physical_Facilities                  Light_Conditions       Weather_Conditions Road_Surface_Conditions Special_Conditions_at_Site  \
0            None within 50 metres  Pedestrian phase at traffic signal junction  Darkness: Street lights present and lit  Fine without high winds                  Dry                       None
1            None within 50 metres        No physical crossing within 50 meters  Darkness: Street lights present and lit  Fine without high winds                  Dry                       None
2            None within 50 metres            non-junction pedestrian crossing           Daylight: Street light present  Fine without high winds                  Dry                       None
3            None within 50 metres        No physical crossing within 50 meters           Daylight: Street light present  Fine without high winds                  Dry                       None
4            None within 50 metres        No physical crossing within 50 meters  Darkness: Street lights present and lit  Fine without high winds                  Dry                       None

  Carriageway_Hazards  Urban_or_Rural_Area  Year
0                None                   1  2012
1                None                   1  2012
2                None                   1  2012
3                None                   1  2012
4                None                   1  2012
```

Efectivamente vemos como las columnas han sido eliminadas.

### 3.12.3. MISSING VALUES

Vamos a proceder al tratamiento de los valores perdidos, primeramente, procederemos a reemplazar los datos faltantes de la columna Road_Surface_Conditions que cuenta con 755 valores faltantes, estos son insuficientes para eliminar la columna por completo, porque perderíamos información valiosa para nuestro proyecto, tampoco es aconsejable eliminar las filas de esos 755 valores, debido a que poseén información relevante, por lo que procederemos a reemplazar los 755 valores por el valor más repetido de la columna. Para saber cual es dicho valor, llamaremos a la función groupby(), que arroja la siguiente información:

```
Road_Surface_Conditions
Dry                            319370
Flood (Over 3cm of water)         863
Frost/Ice                        8140
Snow                             2824
Wet/Damp                       132745
dtype: int64
```

Observamos que el valor más repetido es Dry, por lo que procederemos a reemplazarlo por los valores perdidos, para ello utilizaremos la función replace().

```
#Missing values
#Reemplazamos los valores perdidos por la media
promedio='Dry'
data['Road_Surface_Conditions']=data['Road_Surface_Conditions'].replace(np.nan, promedio)
```

Resultado una vez ejecutado el código anterior:

```
Accident_Severity                            0
Number_of_Vehicles                           0
Date                                         0
Day_of_Week                                  0
Time                                        13
1st_Road_Class                               0
Road_Type                                    0
Speed_limit                                  0
Junction_Detail                         464697
Junction_Control                        178610
2nd_Road_Class                               0
Pedestrian_Crossing-Human_Control            0
Pedestrian_Crossing-Physical_Facilities      0
Light_Conditions                             0
Weather_Conditions                           0
Road_Surface_Conditions                      0
Special_Conditions_at_Site                   2
Carriageway_Hazards                          3
Urban_or_Rural_Area                          0
Year                                         0
dtype: int64
```

```
Road_Surface_Conditions
Dry                            320125
Flood (Over 3cm of water)         863
Frost/Ice                        8140
Snow                             2824
Wet/Damp                       132745
dtype: int64
```

Como observamos en la tabla, no aparecen los 755 valores como nulos en la columna Road_Surface_Conditions y si nos fijamos con detenimiento en el comando groupby() (al cuál nos hemos referido anteriormente), observamos como los valores Dry efectivamente han aumentado 755 veces.

Eliminamos las columnas que poseen un alto número de valores perdidos para esto utilizaremos la función drop().

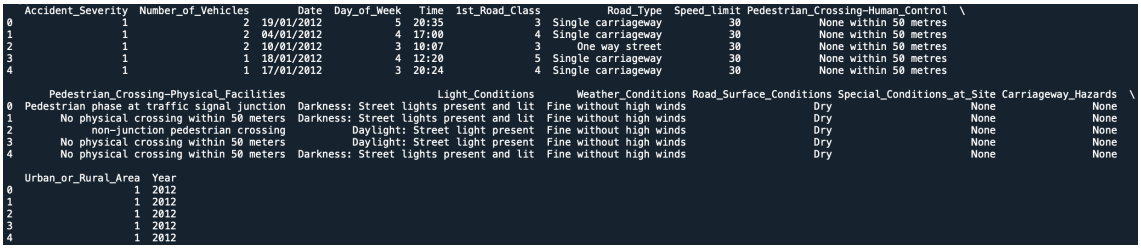
```
#Se eliminan las columnas que tienen muchos datos faltantes

#Eliminacion Junction_Detail 100% Missing values
data=data.drop(['Junction_Detail'], axis=1)

#Eliminacion 2nd_Road_Class 39% Missing values
data=data.drop(['2nd_Road_Class'], axis=1)

#Eliminacion Junction_Control 38% Missing values
data=data.drop(['Junction_Control'], axis=1)
```

Resultado una vez ejecutado el código anterior:



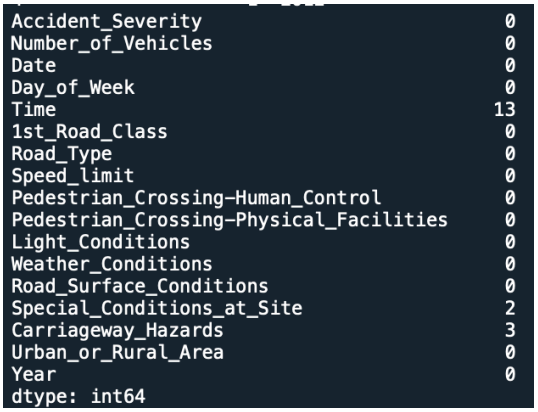
| | Accident_Severity | Number_of_Vehicles | Date | Day_of_Week | Time | 1st_Road_Class | Road_Type | Speed_limit | Pedestrian_Crossing-Human_Control |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 19/01/2012 | 5 | 20:35 | 3 | Single carriageway | 30 | None within 50 metres |
| 1 | 1 | 2 | 04/01/2012 | 4 | 17:00 | 4 | Single carriageway | 30 | None within 50 metres |
| 2 | 1 | 2 | 10/01/2012 | 3 | 10:07 | 3 | One way street | 30 | None within 50 metres |
| 3 | 1 | 1 | 18/01/2012 | 4 | 12:20 | 5 | Single carriageway | 30 | None within 50 metres |
| 4 | 1 | 1 | 17/01/2012 | 3 | 20:24 | 4 | Single carriageway | 30 | None within 50 metres |

| | Pedestrian_Crossing-Physical_Facilities | Light_Conditions | Weather_Conditions | Road_Surface_Conditions | Special_Conditions_at_Site | Carriageway_Hazards |
|---|---|---|---|---|---|---|
| 0 | Pedestrian phase at traffic signal junction | Darkness: Street lights present and lit | Fine without high winds | Dry | None | None |
| 1 | No physical crossing within 50 meters | Darkness: Street lights present and lit | Fine without high winds | Dry | None | None |
| 2 | non-junction pedestrian crossing | Daylight: Street light present | Fine without high winds | Dry | None | None |
| 3 | No physical crossing within 50 meters | Daylight: Street light present | Fine without high winds | Dry | None | None |
| 4 | No physical crossing within 50 meters | Darkness: Street lights present and lit | Fine without high winds | Dry | None | None |

| | Urban_or_Rural_Area | Year |
|---|---|---|
| 0 | 1 | 2012 |
| 1 | 1 | 2012 |
| 2 | 1 | 2012 |
| 3 | 1 | 2012 |
| 4 | 1 | 2012 |

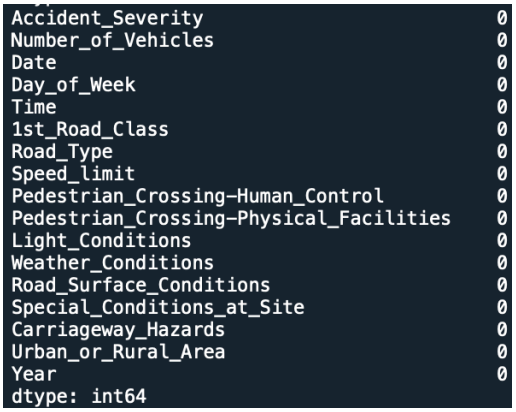
Es relevante señalar que la columna 2nd_Road_Class representa los datos perdidos como -1, por eso debemos tener cuidado cuando llamamos al método isnull(), porque éste, no realizará el conteo como un valor perdido sino como un valor numérico, dicho lo cual, nos damos cuenta de la importancia de realizar una visualización del comportamiento de nuestros datos para darnos cuenta de estas cosas.

Para terminar con los valores perdidos eliminaremos los valores que restan, eliminando las filas que contienen dichos valores, esto no es muy recomendable, porque eliminamos toda la información de la fila y estamos perdiendo información que podría ser relevante, pero como el número de filas con valores perdidos es minúsculo en comparación con el dataset apenas afectará a nuestro modelo, vamos a eliminar tan solo 18 filas de un total de 464697 filas. Para este fin utilizaremos la función dropna().



| | |
|---|---|
| Accident_Severity | 0 |
| Number_of_Vehicles | 0 |
| Date | 0 |
| Day_of_Week | 0 |
| Time | 13 |
| 1st_Road_Class | 0 |
| Road_Type | 0 |
| Speed_limit | 0 |
| Pedestrian_Crossing-Human_Control | 0 |
| Pedestrian_Crossing-Physical_Facilities | 0 |
| Light_Conditions | 0 |
| Weather_Conditions | 0 |
| Road_Surface_Conditions | 0 |
| Special_Conditions_at_Site | 2 |
| Carriageway_Hazards | 3 |
| Urban_or_Rural_Area | 0 |
| Year | 0 |
| dtype: int64 | |

Missing Values antes de dropna()



| | |
|---|---|
| Accident_Severity | 0 |
| Number_of_Vehicles | 0 |
| Date | 0 |
| Day_of_Week | 0 |
| Time | 0 |
| 1st_Road_Class | 0 |
| Road_Type | 0 |
| Speed_limit | 0 |
| Pedestrian_Crossing-Human_Control | 0 |
| Pedestrian_Crossing-Physical_Facilities | 0 |
| Light_Conditions | 0 |
| Weather_Conditions | 0 |
| Road_Surface_Conditions | 0 |
| Special_Conditions_at_Site | 0 |
| Carriageway_Hazards | 0 |
| Urban_or_Rural_Area | 0 |
| Year | 0 |
| dtype: int64 | |

Missing Values después de dropna()

### 3.12.4. CONVERSIÓN DE VARIABLES NÚMERICAS

Si observamos nuestros datos podemos ver como tenemos varias variables que han sido codificadas a variables numéricas, pero nosotros vamos a utilizar una codificación distinta, de manera que, debemos devolver esas variables a variables categóricas, y así obtener un mismo tipo para su posterior conversión a variables tipo dummies, que será explicado en el apartado siguiente.

```
#Reemplazar los valores clase 1st_Road_Class, Day_of_Week, Urban_or_Rural_Area
data['1st_Road_Class'].replace([1,2,3,4,5,6],['Motorway','A(M)','A','B','C','Unclassifield'], inplace=True)
print(data['1st_Road_Class'])

data['Day_of_Week'].replace([1,2,3,4,5,6,7],['Sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday'], inplace=True)
print(data['Day_of_Week'])

data['Urban_or_Rural_Area'].replace([1,2],['Urban','Rural'], inplace=True)
print(data['Urban_or_Rural_Area'])
```

Resultado una vez ejecutado el código anterior:

```
0              A
1              B
2              A
3              C
4              B
           ...
464692      A(M)
464693      A(M)
464694         B
464695         A
464696         B
Name: 1st_Road_Class, Length: 464680, dtype: object
0           Thursday
1          Wednesday
2            Tuesday
3          Wednesday
4            Tuesday
             ...
464692        Sunday
464693      Thursday
464694       Tuesday
464695     Wednesday
464696     Wednesday
Name: Day_of_Week, Length: 464680, dtype: object
0          Urban
1          Urban
2          Urban
3          Urban
4          Urban
           ...
464692     Rural
464693     Rural
464694     Rural
464695     Rural
464696     Rural
Name: Urban_or_Rural_Area, Length: 464680, dtype: object
```

Efectivamente vemos como los cambios se han realizado satisfactoriamente y podemos pasar a su posterior conversión a variables tipo dummies.

### 3.12.5. CONVERSIÓN DE DATOS CATEGÓRICOS

Como observamos en el conjunto de nuestros datos existen varias variables tipo objeto, las cuales se denominan variables categóricas, ahora veremos cómo tratamos estas variables para que nuestros algoritmos de clasificación puedan manejarlas.

**Características nominales y ordinales**

Cuando nos referimos a datos categóricos debemos distinguir entre características nominales y ordinales. Las características ordinales se definen como valores, los cuales pueden ser clasificados u ordenados. Un ejemplo sencillo de esto, podría ser el orden con el que clasificamos las camisetas por talla XL>L>M>S>XS, existe un claro orden que debe ser establecido y las variables deben tener pesos distintos cuando las codifiquemos. Podría establecerse por ejemplo el siguiente orden, 5>4>3>2>1 esto nos serviría para sustituir dichos valores categóricos por numéricos.

Sin embargo, qué ocurre cuando tratamos variables del tipo nominal, aquí el orden es totalmente irrelevante y de establecer dicho orden estaríamos cargando de peso valores que no los deberían de tener. Un ejemplo de esto (siguiendo el ejemplo anterior) sería, el color de una camiseta que puede ser azul, verde, roja, amarilla y naranja, si estableciéramos el valor de 5 para azul, 4 para verde, 3 para roja y sucesivamente estaríamos cometiendo un error, dado que, estaríamos considerando que azul es mayor que verde cuando en realidad no es así, para estos casos lo que se suele hacer es crear columnas para cada color y codificarlas con 1 y 0, este tipo de variables se les conoce como variables dummies.

Tabla 7: Ejemplo variables dummies.

| AZUL | VERDE | ROJO | AMARILLO | NARANJA |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |

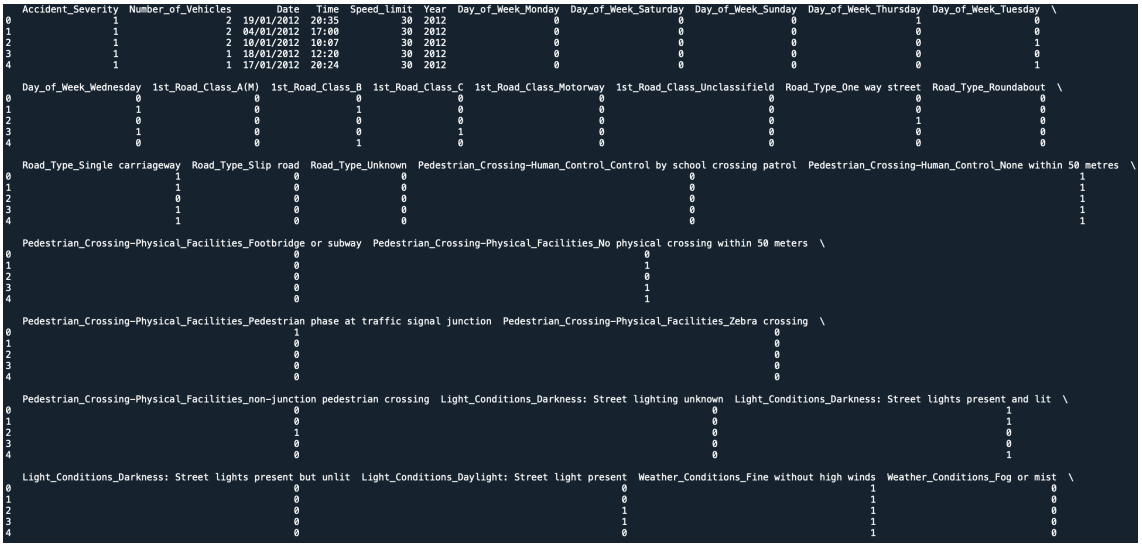
| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 |

Las variables categóricas que intentaremos codificar serán: Day_of_Week,1st_Road_class, Road_Type, Pedestrian_Crossing-Human_Control, Pedestrian_Crossing-Physical_Facilities, Light_Conditions, Weather_Conditions, Road_Surface_Conditions, Special_Conditions_at_Site Carriageway_Hazards y Urban_or_Rural_Area. Utilizaremos la function get dummies(), de esta forma conseguiremos cambiar el tipo de dato de tipo objeto a tipo entero.

Uno de los parámetros que tiene la función get_dummies, permite eliminar la primera columna generada para cada una de las características codificadas, esto se hace para eliminar la determinada colinealidad (cuando una característica es una combinación lineal de las demás, lo que dificulta el funcionamiento de los algoritmos). El parámetro del que disponemos es drop_first.

```
#Conversión datos categóricos
data=pd.get_dummies(data, columns = ['Day_of_Week','1st_Road_Class','Road_Type','Pedestrian_Crossing-Human_Control',
'Pedestrian_Crossing-Physical_Facilities','Light_Conditions','Weather_Conditions','Road_Surface_Conditions',
'Special_Conditions_at_Site','Carriageway_Hazards','Urban_or_Rural_Area'], drop_first = True)
```

Resultado una vez ejecutado el código anterior:



| | Accident_Severity | Number_of_Vehicles | Date | Time | Speed_limit | Year | Day_of_Week_Monday | Day_of_Week_Saturday | Day_of_Week_Sunday | Day_of_Week_Thursday | Day_of_Week_Tuesday | \ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 19/01/2012 | 20:35 | 30 | 2012 | 0 | 0 | 0 | 1 | 0 | |
| 1 | 1 | 2 | 04/01/2012 | 17:00 | 30 | 2012 | 0 | 0 | 0 | 0 | 0 | |
| 2 | 1 | 2 | 10/01/2012 | 10:07 | 30 | 2012 | 0 | 0 | 0 | 0 | 1 | |
| 3 | 1 | 1 | 18/01/2012 | 12:20 | 30 | 2012 | 0 | 0 | 0 | 0 | 0 | |
| 4 | 1 | 1 | 17/01/2012 | 20:24 | 30 | 2012 | 0 | 0 | 0 | 0 | 1 | |

| | Day_of_Week_Wednesday | 1st_Road_Class_A(M) | 1st_Road_Class_B | 1st_Road_Class_C | 1st_Road_Class_Motorway | 1st_Road_Class_Unclassified | Road_Type_One way street | Road_Type_Roundabout | \ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | |
| 3 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | |
| 4 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |

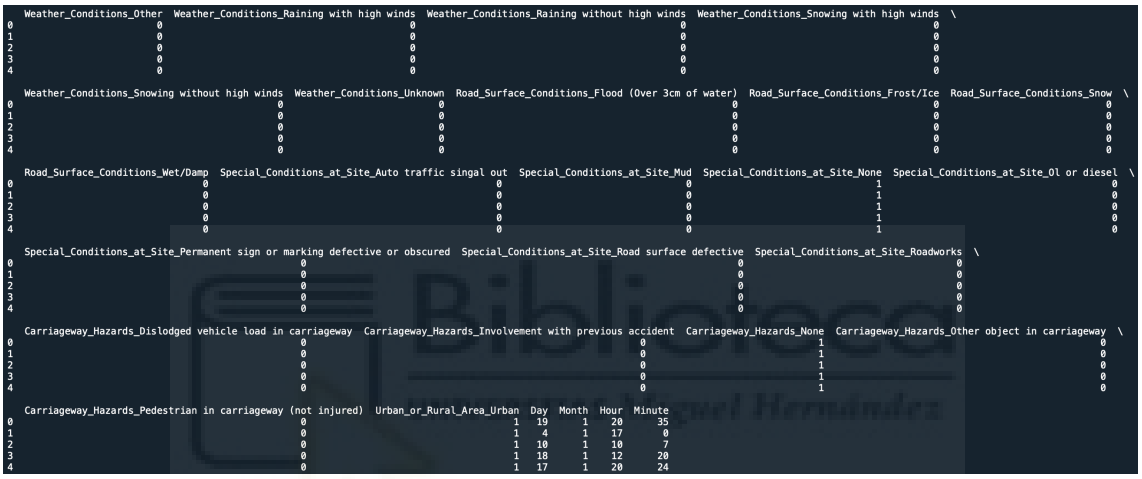
| | Road_Type_Single carriageway | Road_Type_Slip road | Road_Type_Unknown | Pedestrian_Crossing-Human_Control_Control by school crossing patrol | Pedestrian_Crossing-Human_Control_None within 50 metres | \ |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 1 | |
| 1 | 1 | 0 | 0 | 0 | 1 | |
| 2 | 0 | 0 | 0 | 0 | 1 | |
| 3 | 1 | 0 | 0 | 0 | 1 | |
| 4 | 1 | 0 | 0 | 0 | 1 | |

| | Pedestrian_Crossing-Physical_Facilities_Footbridge or subway | Pedestrian_Crossing-Physical_Facilities_No physical crossing within 50 meters | \ |
|---|---|---|---|
| 0 | 0 | 0 | |
| 1 | 0 | 1 | |
| 2 | 0 | 0 | |
| 3 | 0 | 1 | |
| 4 | 0 | 1 | |

| | Pedestrian_Crossing-Physical_Facilities_Pedestrian phase at traffic signal junction | Pedestrian_Crossing-Physical_Facilities_Zebra crossing | \ |
|---|---|---|---|
| 0 | 1 | 0 | |
| 1 | 0 | 0 | |
| 2 | 0 | 0 | |
| 3 | 0 | 0 | |
| 4 | 0 | 0 | |

| | Pedestrian_Crossing-Physical_Facilities_non-junction pedestrian crossing | Light_Conditions_Darkness: Street lighting unknown | Light_Conditions_Darkness: Street lights present and lit | \ |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | |
| 1 | 0 | 0 | 1 | |
| 2 | 1 | 0 | 0 | |
| 3 | 0 | 0 | 0 | |
| 4 | 0 | 0 | 1 | |

| | Light_Conditions_Darkness: Street lights present but unlit | Light_Conditions_Daylight: Street light present | Weather_Conditions_Fine without high winds | Weather_Conditions_Fog or mist | \ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | |
| 1 | 0 | 0 | 1 | 0 | |
| 2 | 0 | 1 | 1 | 0 | |
| 3 | 0 | 1 | 1 | 0 | |
| 4 | 0 | 0 | 1 | 0 | |



| | Weather_Conditions_Other | Weather_Conditions_Raining with high winds | Weather_Conditions_Raining without high winds | Weather_Conditions_Snowing with high winds | \ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 0 | 0 | |
| 2 | 0 | 0 | 0 | 0 | |
| 3 | 0 | 0 | 0 | 0 | |
| 4 | 0 | 0 | 0 | 0 | |

| | Weather_Conditions_Snowing without high winds | Weather_Conditions_Unknown | Road_Surface_Conditions_Flood (Over 3cm of water) | Road_Surface_Conditions_Frost/Ice | Road_Surface_Conditions_Snow | \ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 0 | 0 | 0 | |
| 2 | 0 | 0 | 0 | 0 | 0 | |
| 3 | 0 | 0 | 0 | 0 | 0 | |
| 4 | 0 | 0 | 0 | 0 | 0 | |

| | Road_Surface_Conditions_Wet/Damp | Special_Conditions_at_Site_Auto traffic singal out | Special_Conditions_at_Site_Mud | Special_Conditions_at_Site_None | Special_Conditions_at_Site_Ol or diesel | \ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | |
| 1 | 0 | 0 | 0 | 1 | 0 | |
| 2 | 0 | 0 | 0 | 1 | 0 | |
| 3 | 0 | 0 | 0 | 1 | 0 | |
| 4 | 0 | 0 | 0 | 1 | 0 | |

| | Special_Conditions_at_Site_Permanent sign or marking defective or obscured | Special_Conditions_at_Site_Road surface defective | Special_Conditions_at_Site_Roadworks | \ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 0 | |
| 2 | 0 | 0 | 0 | |
| 3 | 0 | 0 | 0 | |
| 4 | 0 | 0 | 0 | |

| | Carriageway_Hazards_Dislodged vehicle load in carriageway | Carriageway_Hazards_Involvement with previous accident | Carriageway_Hazards_None | Carriageway_Hazards_Other object in carriageway | \ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | |
| 1 | 0 | 0 | 1 | 0 | |
| 2 | 0 | 0 | 1 | 0 | |
| 3 | 0 | 0 | 1 | 0 | |
| 4 | 0 | 0 | 1 | 0 | |

| | Carriageway_Hazards_Pedestrian in carriageway (not injured) | Urban_or_Rural_Area_Urban | Day | Month | Hour | Minute |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 19 | 1 | 20 | 35 |
| 1 | 0 | 1 | 4 | 1 | 17 | 0 |
| 2 | 0 | 1 | 10 | 1 | 10 | 7 |
| 3 | 0 | 1 | 18 | 1 | 12 | 20 |
| 4 | 0 | 1 | 17 | 1 | 20 | 24 |

Como vemos en la captura se ha establecido una nueva columna para cada condición de nuestras variables dummies, a consecuencia de esto hemos aumentado el número de columnas considerablemente, pero sin que esto afecte a nuestro modelo, de forma que, todavía tenemos muchas filas en proporción con las columnas y hemos conseguido el cambio de tipo de variable que estábamos buscando.

```
Accident_Severity                                                          int64
Number_of_Vehicles                                                         int64
Date                                                                      object
Time                                                                      object
Speed_limit                                                                int64
Year                                                                       int64
Day_of_Week_Monday                                                         uint8
Day_of_Week_Saturday                                                       uint8
Day_of_Week_Sunday                                                         uint8
Day_of_Week_Thursday                                                       uint8
Day_of_Week_Tuesday                                                        uint8
Day_of_Week_Wednesday                                                      uint8
1st_Road_Class_A(M)                                                        uint8
1st_Road_Class_B                                                           uint8
1st_Road_Class_C                                                           uint8
1st_Road_Class_Motorway                                                    uint8
1st_Road_Class_Unclassifield                                               uint8
Road_Type_One way street                                                   uint8
Road_Type_Roundabout                                                       uint8
Road_Type_Single carriageway                                               uint8
Road_Type_Slip road                                                        uint8
Road_Type_Unknown                                                          uint8
Pedestrian_Crossing-Human_Control_Control by school crossing patrol        uint8
Pedestrian_Crossing-Human_Control_None within 50 metres                    uint8
Pedestrian_Crossing-Physical_Facilities_Footbridge or subway               uint8
Pedestrian_Crossing-Physical_Facilities_No physical crossing within 50 meters  uint8
Pedestrian_Crossing-Physical_Facilities_Pedestrian phase at traffic signal junction  uint8
Pedestrian_Crossing-Physical_Facilities_Zebra crossing                     uint8
Pedestrian_Crossing-Physical_Facilities_non-junction pedestrian crossing   uint8
Light_Conditions_Darkness: Street lighting unknown                         uint8
Light_Conditions_Darkness: Street lights present and lit                   uint8
Light_Conditions_Darkness: Street lights present but unlit                 uint8
Light_Conditions_Daylight: Street light present                            uint8
Weather_Conditions_Fine without high winds                                 uint8
Weather_Conditions_Fog or mist                                             uint8
Weather_Conditions_Other                                                   uint8
Weather_Conditions_Raining with high winds                                 uint8
Weather_Conditions_Raining without high winds                              uint8
Weather_Conditions_Snowing with high winds                                 uint8
Weather_Conditions_Snowing without high winds                              uint8
Weather_Conditions_Unknown                                                 uint8
Road_Surface_Conditions_Flood (Over 3cm of water)                          uint8
Road_Surface_Conditions_Frost/Ice                                          uint8
Road_Surface_Conditions_Snow                                               uint8
Road_Surface_Conditions_Wet/Damp                                           uint8
Special_Conditions_at_Site_Auto traffic singal out                         uint8
Special_Conditions_at_Site_Mud                                             uint8
Special_Conditions_at_Site_None                                            uint8
Special_Conditions_at_Site_Oil or diesel                                   uint8
Special_Conditions_at_Site_Permanent sign or marking defective or obscured uint8
Special_Conditions_at_Site_Road surface defective                          uint8
Special_Conditions_at_Site_Roadworks                                       uint8
Carriageway_Hazards_Dislodged vehicle load in carriageway                  uint8
Carriageway_Hazards_Involvement with previous accident                     uint8
Carriageway_Hazards_None                                                   uint8
Carriageway_Hazards_Other object in carriageway                            uint8
Carriageway_Hazards_Pedestrian in carriageway (not injured)                uint8
Urban_or_Rural_Area_Urban                                                  uint8
Day                                                                        int64
Month                                                                      int64
Hour                                                                       int64
Minute                                                                     int64
dtype: object
```

### 3.12.6. VARIABLES TEMPORALES

Este tipo de variables se pueden tratar de varias formas; una de ellas que suele ser bien aceptada consiste en cambiar el dato, tipo objeto al tipo datatime para fechas ../../.. y timedelta para horas ..:..:.. así es procesado con normalidad por nuestros algoritmos de clasificación, pero a nosotros nos surge el problema de que nuestras horas vienen en formato solo hora y minutos, y no podemos convertirlo a formato timedalta. Veamos un ejemplo en nuestro proyecto de cómo quedaría esta forma de preprocesamiento de fechas y horas.

Primero ejecutamos el código con la conversión de datos en la fecha:

```
#Conversión datos temporales
data['Date']=pd.to_datetime(data['Date'])
```

Resultado una vez ejecutamos el código anterior:

```
Accident_Severity                                                          int64
Number_of_Vehicles                                                         int64
Date                                                             datetime64[ns]
Time                                                                     object
Speed_limit                                                                int64
Year                                                                       int64
Day_of_Week_Monday                                                         uint8
Day_of_Week_Saturday                                                       uint8
Day_of_Week_Sunday                                                         uint8
Day_of_Week_Thursday                                                       uint8
Day_of_Week_Tuesday                                                        uint8
Day_of_Week_Wednesday                                                      uint8
1st_Road_Class_A(M)                                                        uint8
1st_Road_Class_B                                                           uint8
1st_Road_Class_C                                                           uint8
1st_Road_Class_Motorway                                                    uint8
1st_Road_Class_Unclassifield                                               uint8
Road_Type_One way street                                                   uint8
Road_Type_Roundabout                                                       uint8
Road_Type_Single carriageway                                               uint8
Road_Type_Slip road                                                        uint8
Road_Type_Unknown                                                          uint8
Pedestrian_Crossing-Human_Control_Control by school crossing patrol        uint8
Pedestrian_Crossing-Human_Control_None within 50 metres                    uint8
Pedestrian_Crossing-Physical_Facilities_Footbridge or subway               uint8
Pedestrian_Crossing-Physical_Facilities_No physical crossing within 50 meters uint8
Pedestrian_Crossing-Physical_Facilities_Pedestrian phase at traffic signal junction uint8
Pedestrian_Crossing-Physical_Facilities_Zebra crossing                     uint8
Pedestrian_Crossing-Physical_Facilities_non-junction pedestrian crossing   uint8
Light_Conditions_Darkness: Street lighting unknown                         uint8
Light_Conditions_Darkness: Street lights present and lit                   uint8
Light_Conditions_Darkness: Street lights present but unlit                 uint8
Light_Conditions_Daylight: Street light present                            uint8
Weather_Conditions_Fine without high winds                                 uint8
Weather_Conditions_Fog or mist                                             uint8
Weather_Conditions_Other                                                   uint8
Weather_Conditions_Raining with high winds                                 uint8
Weather_Conditions_Raining without high winds                              uint8
Weather_Conditions_Snowing with high winds                                 uint8
Weather_Conditions_Snowing without high winds                              uint8
Weather_Conditions_Unknown                                                 uint8
Road_Surface_Conditions_Flood (Over 3cm of water)                          uint8
Road_Surface_Conditions_Frost/Ice                                          uint8
Road_Surface_Conditions_Snow                                               uint8
Road_Surface_Conditions_Wet/Damp                                           uint8
Special_Conditions_at_Site_Auto traffic singal out                         uint8
Special_Conditions_at_Site_Mud                                             uint8
Special_Conditions_at_Site_None                                            uint8
Special_Conditions_at_Site_Oil or diesel                                   uint8
Special_Conditions_at_Site_Permanent sign or marking defective or obscured uint8
Special_Conditions_at_Site_Road surface defective                          uint8
Special_Conditions_at_Site_Roadworks                                       uint8
Carriageway_Hazards_Dislodged vehicle load in carriageway                  uint8
Carriageway_Hazards_Involvement with previous accident                     uint8
Carriageway_Hazards_None                                                   uint8
Carriageway_Hazards_Other object in carriageway                            uint8
Carriageway_Hazards_Pedestrian in carriageway (not injured)                uint8
Urban_or_Rural_Area_Urban                                                  uint8
dtype: object
```

Observamos como la conversión se ha realizado correctamente del tipo objeto al tipo datetime64.

Ahora procedemos a hacer lo mismo con el formato tipo horas:

```
#Conversión datos temporales
data['Time']=pd.to_timedelta(data['Time'])
```

Resultado una vez ejecutado el código anterior:

El programa no realiza la conversión, debido a que nuestro formato original para el atributo Time no cuenta con los segundos, y no se puede realizar la conversión de un formato a otro.

```
ValueError: expected hh:mm:ss format
```

En nuestro proyecto realizaremos otra forma distinta para tratar los datos tipo fecha y hora, primeramente vamos a separar la variable Date en day y month, y

después Time en hour y minute, observamos que son cadenas de caracteres (string), por lo que podemos realizar esto con la función slice() para la separación de caracteres, pasándole como argumento los intervalo de valores que queremos seleccionar. Con la función astype() convertimos el valor de la cadena de caracteres a un número entero, y por ultimo cogemos el valor seleccionado y lo pasamos como argumento a la función assign() con la que crearemos una nueva columna en nuestro dataset.

```
#Conversión del datos temporales
day=data.Date.str.slice(0,2).astype(int)
data=data.assign(Day=day.values)

month=data.Date.str.slice(3,5).astype(int)
data=data.assign(Month=month.values)

hour=data.Time.str.slice(0,2).astype(int)
data=data.assign(Hour=hour.values)

minute=data.Time.str.slice(3,5).astype(int)
data=data.assign(Minute=minute.values)

print(data.dtypes)
print(data.head())
```

Resultado una vez ejecutado el código anterior:

```
Accident_Severity                                                    int64
Number_of_Vehicles                                                   int64
Date                                                                object
Time                                                                object
Speed_limit                                                          int64
Year                                                                 int64
Day_of_Week_Monday                                                   uint8
Day_of_Week_Saturday                                                 uint8
Day_of_Week_Sunday                                                   uint8
Day_of_Week_Thursday                                                 uint8
Day_of_Week_Tuesday                                                  uint8
Day_of_Week_Wednesday                                                uint8
1st_Road_Class_A(M)                                                  uint8
1st_Road_Class_B                                                     uint8
1st_Road_Class_C                                                     uint8
1st_Road_Class_Motorway                                              uint8
1st_Road_Class_Unclassifield                                         uint8
Road_Type_One way street                                             uint8
Road_Type_Roundabout                                                 uint8
Road_Type_Single carriageway                                         uint8
Road_Type_Slip road                                                  uint8
Road_Type_Unknown                                                    uint8
Pedestrian_Crossing-Human_Control_Control by school crossing patrol  uint8
Pedestrian_Crossing-Human_Control_None within 50 metres              uint8
Pedestrian_Crossing-Physical_Facilities_Footbridge or subway         uint8
Pedestrian_Crossing-Physical_Facilities_No physical crossing within 50 meters  uint8
Pedestrian_Crossing-Physical_Facilities_Pedestrian phase at traffic signal junction  uint8
Pedestrian_Crossing-Physical_Facilities_Zebra crossing               uint8
Pedestrian_Crossing-Physical_Facilities_non-junction pedestrian crossing  uint8
Light_Conditions_Darkness: Street lighting unknown                   uint8
Light_Conditions_Darkness: Street lights present and lit             uint8
Light_Conditions_Darkness: Street lights present but unlit           uint8
Light_Conditions_Daylight: Street light present                      uint8
Weather_Conditions_Fine without high winds                           uint8
Weather_Conditions_Fog or mist                                       uint8
Weather_Conditions_Other                                             uint8
Weather_Conditions_Raining with high winds                           uint8
Weather_Conditions_Raining without high winds                        uint8
Weather_Conditions_Snowing with high winds                           uint8
Weather_Conditions_Snowing without high winds                        uint8
Weather_Conditions_Unknown                                           uint8
Road_Surface_Conditions_Flood (Over 3cm of water)                    uint8
Road_Surface_Conditions_Frost/Ice                                    uint8
Road_Surface_Conditions_Snow                                         uint8
Road_Surface_Conditions_Wet/Damp                                     uint8
Special_Conditions_at_Site_Auto traffic singal out                   uint8
Special_Conditions_at_Site_Mud                                       uint8
Special_Conditions_at_Site_None                                      uint8
Special_Conditions_at_Site_Oil or diesel                             uint8
Special_Conditions_at_Site_Permanent sign or marking defective or obscured  uint8
Special_Conditions_at_Site_Road surface defective                    uint8
Special_Conditions_at_Site_Roadworks                                 uint8
Carriageway_Hazards_Dislodged vehicle load in carriageway            uint8
Carriageway_Hazards_Involvement with previous accident               uint8
Carriageway_Hazards_None                                             uint8
Carriageway_Hazards_Other object in carriageway                      uint8
Carriageway_Hazards_Pedestrian in carriageway (not injured)          uint8
Urban_or_Rural_Area_Urban                                            uint8
Day                                                                  int64
Month                                                                int64
Hour                                                                 int64
Minute                                                               int64
dtype: object
```

| Day | Month | Hour | Minute |
|---|---|---|---|
| 19 | 1 | 20 | 35 |
| 4 | 1 | 17 | 0 |
| 10 | 1 | 10 | 7 |
| 18 | 1 | 12 | 20 |
| 17 | 1 | 20 | 24 |

Observamos cómo se han producidos cuatro nuevas columnas, por lo tanto, podemos proceder a eliminar las columnas tipo objeto Time y Date. Sus valores han sido almacenados en estas nuevas columnas y son redundantes para nuestro proyecto. Así nos fijamos en que la columna Minute aporta muy poca información y no tendrá ninguna incidencia a la hora de calcular la gravedad de nuestro accidente, por lo que procedemos a eliminarla, utilizando la misma función que ya hemos utilizado antes para la eliminación de columnas.

```
#Eliminamos variables por irrelevante o reduntantes
data=data.drop(['Minute','Date','Time'], axis=1)
```

Ahora por fin tenemos los datos libres de valores perdidos y con todas las variables convertidas en datos tipo enteros.

```
Accident_Severity                                                          int64
Number_of_Vehicles                                                         int64
Speed_limit                                                                int64
Year                                                                       int64
Day_of_Week_Monday                                                         uint8
Day_of_Week_Saturday                                                       uint8
Day_of_Week_Sunday                                                         uint8
Day_of_Week_Thursday                                                       uint8
Day_of_Week_Tuesday                                                        uint8
Day_of_Week_Wednesday                                                      uint8
1st_Road_Class_A(M)                                                        uint8
1st_Road_Class_B                                                           uint8
1st_Road_Class_C                                                           uint8
1st_Road_Class_Motorway                                                    uint8
1st_Road_Class_Unclassifield                                               uint8
Road_Type_One way street                                                   uint8
Road_Type_Roundabout                                                       uint8
Road_Type_Single carriageway                                               uint8
Road_Type_Slip road                                                        uint8
Road_Type_Unknown                                                          uint8
Pedestrian_Crossing-Human_Control_Control by school crossing patrol        uint8
Pedestrian_Crossing-Human_Control_None within 50 metres                    uint8
Pedestrian_Crossing-Physical_Facilities_Footbridge or subway               uint8
Pedestrian_Crossing-Physical_Facilities_No physical crossing within 50 meters  uint8
Pedestrian_Crossing-Physical_Facilities_Pedestrian phase at traffic signal junction  uint8
Pedestrian_Crossing-Physical_Facilities_Zebra crossing                     uint8
Pedestrian_Crossing-Physical_Facilities_non-junction pedestrian crossing   uint8
Light_Conditions_Darkness: Street lighting unknown                         uint8
Light_Conditions_Darkness: Street lights present and lit                   uint8
Light_Conditions_Darkness: Street lights present but unlit                 uint8
Light_Conditions_Daylight: Street light present                            uint8
Weather_Conditions_Fine without high winds                                 uint8
Weather_Conditions_Fog or mist                                             uint8
Weather_Conditions_Other                                                   uint8
Weather_Conditions_Raining with high winds                                 uint8
Weather_Conditions_Raining without high winds                              uint8
Weather_Conditions_Snowing with high winds                                 uint8
Weather_Conditions_Snowing without high winds                              uint8
Weather_Conditions_Unknown                                                 uint8
Road_Surface_Conditions_Flood (Over 3cm of water)                          uint8
Road_Surface_Conditions_Frost/Ice                                          uint8
Road_Surface_Conditions_Snow                                               uint8
Road_Surface_Conditions_Wet/Damp                                           uint8
Special_Conditions_at_Site_Auto traffic singal out                         uint8
Special_Conditions_at_Site_Mud                                             uint8
Special_Conditions_at_Site_None                                            uint8
Special_Conditions_at_Site_Oil or diesel                                   uint8
Special_Conditions_at_Site_Permanent sign or marking defective or obscured uint8
Special_Conditions_at_Site_Road surface defective                          uint8
Special_Conditions_at_Site_Roadworks                                       uint8
Carriageway_Hazards_Dislodged vehicle load in carriageway                  uint8
Carriageway_Hazards_Involvement with previous accident                     uint8
Carriageway_Hazards_None                                                   uint8
Carriageway_Hazards_Other object in carriageway                            uint8
Carriageway_Hazards_Pedestrian in carriageway (not injured)                uint8
Urban_or_Rural_Area_Urban                                                  uint8
Day                                                                        int64
Month                                                                      int64
Hour                                                                       int64
dtype: object
```

## 3.13. SEPARACIÓN DE LOS DATOS

Una vez hemos conocido cómo se comportan nuestros datos y hemos realizado el procesamiento de estos, procederemos a separar nuestro conjunto de datos en ‘X’, para todas las variables independientes, y en un segundo conjunto de variables dependientes ‘y’.

Procedemos a crear una variable ‘X’ para los datos de entrada y una variable ‘y’ para la columna correspondiente a ‘Accident_Severity’.

```
#Separacion de los datos
X = np.array(data.drop(['Accident_Severity'], 1))
y = np.array(data['Accident_Severity'])
```

Existen varias formas de separar los datos, nosotros utilizaremos la función drop() para eliminar la columna Accident_Severity de nuestro dataset original y después convertiremos el conjunto resultante en un array, mediante la función array() de la librería numpy, por último, asignaremos el array creado a nuestra variable ‘X’. Para la ‘y’ simplemente seleccionaremos la columna Accident_Severity y la convertiremos en un array con la función array(), después procederemos a asignarla a la variable ‘y’.

Si nos fijamos en el variable explorer para comprobar si se han separado bien nuestras variables, obtenemos los siguientes resultados:

- Para la ‘X’


- Para la ‘y’


## 3.14. DIMENSIONALIDAD DE LAS CARACTERÍSTICAS

Una alternativa para reducir la complejidad de nuestro modelo y de esta forma evitar el sobreajuste, es la de reducir la dimensionalidad mediante la selección de características. Los conjuntos de datos en ocasiones cuentan con un gran

número de características, esto ocasiona que sean difíciles de procesar y pueden generar los siguientes problemas en nuestro proyecto:

- Las características adicionales actúan como un ruido para el cual el modelo puede tener un rendimiento extremadamente bajo.
- El modelo tarda más tiempo en entrenarse.
- Asignación de recursos innecesarios para estas características.

Existen dos técnicas principales para la reducción de la dimensionalidad, una es la selección de características y la otra, la extracción de características. Mediante la selección de características seleccionamos un subconjunto de las características que tenemos originalmente, mientras que, en la extracción de características construiremos un nuevo subespacio a partir de la información del conjunto de características original.

### 3.14.1. MÉTODOS DE SELECCIÓN DE CARACTERÍSTICAS

**Métodos de filtros**

Los métodos de filtro consisten en seleccionar el mejor subconjunto de características para nuestra variable objetivo, se utilizan generalmente como un paso de procesamiento de datos, esta selección de características es independiente de cualquier algoritmo de clasificación.

Las características se clasificarán según puntajes estadísticos, que determinarán la correlación de las características con nuestra variable objetivo.

Algunos de los métodos de filtro más utilizado son la correlación de Pearson, LDA, ANOVA y chi-cuadrado.

**Métodos de envoltura**

Estos métodos al contrario que ocurre en los métodos de filtro, necesitan de un algoritmo, para así poder generar el mejor subconjunto de características que se adecúe mejor al rendimiento de nuestro algoritmo. El problema de estos métodos suele ser que resultan muy costosos computacionalmente hablando.

Algunos métodos de envoltura comunes son los siguientes:

1. Selección hacia delante (Forward Selection)
2. Eliminación hacia atrás (Backward Selection)
3. Eliminación de características recursivas (Recursive Feature)

**Métodos integrados**

Consisten en la combinación de los métodos de filtro y de envoltura. Se implementan mediante algoritmos los cuales tiene sus propios métodos para seleccionar las características.

Algunos ejemplos son la regresión LASSO y RIDGE, que tienen funciones de penalización incorporadas para reducir el sobreajuste.

### 3.14.2. MÉTODOS DE EXTRACCIÓN DE CARACTERÍSTICAS

**Análisis de componentes principales (PCA)**

Es una técnica de transformación lineal sin supervisión, sirve para extraer las características y reducir la dimensionalidad, ayudándonos a identificar patrones en los datos basándose en la correlación de características. En otras palabras, el PCA ayuda a encontrar las direcciones de varianza máxima en datos de alta dimensión y las proyecta en un nuevo subespacio con dimensiones iguales o menores que el original.

**Análisis discriminante lineal (ADL)**

Es una técnica para la reducción de dimensionalidad, supervisada para maximizar la divisibilidad de clases, es frecuente su utilización para reducir el grado de sobreajuste debido a la maldición de la dimensionalidad en modelos no regularizados. La idea principal que hay detrás del ADL es muy similar al de PCA. El objetivo principal del ADL es encontrar el subespacio de características que optimice la divisibilidad de clases.

**Análisis de componentes principales con kernels (ACPK)**

Esta técnica sirve para la reducción de la dimensionalidad no lineal en nuestro conjunto de datos. Es una técnica empleada cuando el conjunto de nuestros

datos no es lineal y por lo tanto no es buena idea aplicar el análisis PCA ni el ADL, podemos abordar estos problemas proyectándolos en un nuevo espacio de características de dimensionalidad superior, donde las clases pasan a ser separables linealmente.

### 3.14.3. APLICACIÓN A NUESTRO PROYECTO

Nuestro proyecto tiene una dimensionalidad corta, de manera que en un principio no es necesario aplicar ninguna de las técnicas anteriormente descritas. Utilizaremos ésta de un modo anecdótico y para observar cómo se comportan las variables con nuestra variable objetivo de una forma distinta a la correlación vista antes.

Aplicaremos la técnica de Chi-cuadrado de los métodos de filtro, esta técnica consiste en una prueba estadística aplicada a los grupos de características para evaluar la probabilidad de correlación entre ellos utilizando su distribución de frecuencia.

Los métodos de filtro no eliminan la multicolinealidad, por lo tanto, se debe eliminar esta antes de entrenar los modelos.

Vamos a implementar una prueba estadística de Chi-cuadrado es conveniente destacar que esta prueba se puede aplicar siempre y cuando no tengamos valores negativos en nuestro conjunto de datos.

La librería sciket-learn viene con la clase SelectKBest que incluye la función Chi-cuadrado para la selección de características.

```
#Métodos de filtro
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

#Extracion de caracteristicas chi-cuadrado
prueba = SelectKBest(score_func=chi2, k=5)
entrenamiento=prueba.fit(X, y)

#Puntuaje caracteristicas chi-cuadrado
np.set_printoptions(precision=3,suppress=True)
print(entrenamiento.scores_)
```

Resultado una vez ejecutado el siguiente código:

```
[   18.302 12326.822     0.        4.647    63.806   118.383    15.71
    11.651    29.97     16.089     2.304     5.475    20.015   205.67
    23.95    221.951     4.395     7.396     3.496     2.45      0.016
     1.568    22.057    83.66     44.489    12.872     6.364     1.266
    16.384   123.163     7.454    23.906    11.934     0.001    29.995
     3.478     1.556    17.002     0.477     0.671     8.256    10.225
     2.216     3.181     0.091     3.125     0.496     1.272     1.226
     1.036    17.267     0.047    12.526     3.95    661.32      0.383
    13.681    72.075]
```

```
Number_of_Vehicles                                                          int64
Speed_limit                                                                 int64
Year                                                                        int64
Day_of_Week_Monday                                                          uint8
Day_of_Week_Saturday                                                        uint8
Day_of_Week_Sunday                                                          uint8
Day_of_Week_Thursday                                                        uint8
Day_of_Week_Tuesday                                                         uint8
Day_of_Week_Wednesday                                                       uint8
1st_Road_Class_A(M)                                                         uint8
1st_Road_Class_B                                                            uint8
1st_Road_Class_C                                                            uint8
1st_Road_Class_Motorway                                                     uint8
1st_Road_Class_Unclassifield                                                uint8
Road_Type_One way street                                                    uint8
Road_Type_Roundabout                                                        uint8
Road_Type_Single carriageway                                                uint8
Road_Type_Slip road                                                         uint8
Road_Type_Unknown                                                           uint8
Pedestrian_Crossing-Human_Control_Control by school crossing patrol         uint8
Pedestrian_Crossing-Human_Control_None within 50 metres                     uint8
Pedestrian_Crossing-Physical_Facilities_Footbridge or subway                uint8
Pedestrian_Crossing-Physical_Facilities_No physical crossing within 50 meters uint8
Pedestrian_Crossing-Physical_Facilities_Pedestrian phase at traffic signal junction uint8
Pedestrian_Crossing-Physical_Facilities_Zebra crossing                      uint8
Pedestrian_Crossing-Physical_Facilities_non-junction pedestrian crossing    uint8
Light_Conditions_Darkness: Street lighting unknown                          uint8
Light_Conditions_Darkness: Street lights present and lit                    uint8
Light_Conditions_Darkness: Street lights present but unlit                  uint8
Light_Conditions_Daylight: Street light present                             uint8
Weather_Conditions_Fine without high winds                                  uint8
Weather_Conditions_Fog or mist                                              uint8
Weather_Conditions_Other                                                    uint8
Weather_Conditions_Raining with high winds                                  uint8
Weather_Conditions_Raining without high winds                               uint8
Weather_Conditions_Snowing with high winds                                  uint8
Weather_Conditions_Snowing without high winds                               uint8
Weather_Conditions_Unknown                                                  uint8
Road_Surface_Conditions_Flood (Over 3cm of water)                           uint8
Road_Surface_Conditions_Frost/Ice                                           uint8
Road_Surface_Conditions_Snow                                                uint8
Road_Surface_Conditions_Wet/Damp                                            uint8
Special_Conditions_at_Site_Auto traffic singal out                          uint8
Special_Conditions_at_Site_Mud                                              uint8
Special_Conditions_at_Site_None                                             uint8
Special_Conditions_at_Site_Oil or diesel                                    uint8
Special_Conditions_at_Site_Permanent sign or marking defective or obscured  uint8
Special_Conditions_at_Site_Road surface defective                           uint8
Special_Conditions_at_Site_Roadworks                                        uint8
Carriageway_Hazards_Dislodged vehicle load in carriageway                   uint8
Carriageway_Hazards_Involvement with previous accident                      uint8
Carriageway_Hazards_None                                                    uint8
Carriageway_Hazards_Other object in carriageway                             uint8
Carriageway_Hazards_Pedestrian in carriageway (not injured)                 uint8
Urban_or_Rural_Area_Urban                                                   uint8
Day                                                                         int64
Month                                                                       int64
Hour                                                                        int64
dtype: object
```

Los puntajes obtenidos de izquierda a derecha se asocian con las características de la lista de arriba abajo, por ejemplo, el primer valor de 18.302 es el puntaje asociado a la característica Number_of_Vehicles y el último puntaje de 72.075 se asocia a la característica Hour.

De esta forma podemos crear un grupo con las características con el puntaje más alto y asignarlos a la variable ‘X’. Por ejemplo, cogiendo las cinco características con el puntaje más alto nuestra ‘X’ sería la siguiente:

Tabla 8: Resultados de las variables con mejor puntaje.

| COLUMNA | VARIABLE | PUNTAJE |
|---|---|---|
| 2 | Speed_limit | 12326.822 |
| 16 | Road_Type_Roundabout | 221.951 |
| 14 | 1st_Road_Class_Unclassifield | 205.67 |
| 30 | Light_Conditions_Daylight: Street light present | 123.163 |
| 6 | Day_of_Week_Sunday | 118.383 |

Estas son las 5 características que tienen mayor impacto para nuestra variable dependiente. Utilizando el método iloc[ ] de pandas se permite la selección de filas y columnas y podemos crear nuestro nuevo conjunto de datos con estas 5 características, el código quedaría de la siguiente manera:

```
X=data.iloc[:, [2,16,14,30,6]].values
print(X)
```

Resultado una vez ejecutado el código anterior:

```
[[30  0  0  0  0]
 [30  0  0  0  0]
 [30  0  0  1  0]
 ...
 [40  0  0  0  0]
 [60  0  0  0  0]
 [60  0  0  1  0]]
```

Esto es simplemente anecdótico y ésta no será nuestra ‘X’, simplemente era un ejemplo para ver como se podría reducir la dimensionalidad en nuestro proyecto. Para continuar con nuestro problema, procederemos a deshacer los cambios en la ‘X’ que acabamos de crear y seguiremos lo que estábamos haciendo anteriormente a este apartado.

## 3.15. DISTRIBUCIÓN DE CLASES

En los problemas de clasificación es necesario conocer si nuestros datos de la clase objetivo se encuentran equilibrados. Cuando los datos se encuentran altamente desequilibrados, es decir, más muestras de una clase que de otra, se requiere realizar un tratamiento para solventar dicho problema, dado que la mayoría de los algoritmos procesan mal los datos desbalanceados. Para saber si nuestros datos se encuentran desbalanceados, vamos a implementar la función groupby() de la librería Pandas junto con el argumento de la columna que queremos conocer:

```
#Distribucion de clases
conteo_data=data.groupby('Accident_Severity').size()
print(conteo_data)
```

Resultado una vez ejecutado el código anterior:

```
Accident_Severity
0    392610
1     72087
dtype: int64
```

Observamos un claro desbalanceamiento en nuestra muestra, la clase minoritaria pertenece al 18.36% de los datos totales, mientras que la clase mayoritaria muestra el resto de los datos de la muestra, es decir, el 81.64% de los datos restantes. Para solventar este tipo de problemas se utilizan varias técnicas:

**Recopilar más datos**

Este método consiste en la recopilación de un mayor número de datos, ya que podría ser que nuestra muestra estuviera sesgada y se hubieran recopilado más datos de un caso que del otro.

**Utilizar las métricas de evaluación correctas**

Cuando apliquemos las métricas de evaluación a nuestro problema, debemos ir con mucho cuidado, porque la utilización de datos desequilibrados puede resultar peligrosa y dar resultados erróneos. Cuando esto ocurre se pueden

utilizar métricas alternativas como son: Precisión, Especificidad, Sensibilidad, Puntaje de F1, AUC.

**Remuestreo del conjunto de los datos**

Otra técnica empleada para tratar el problema del desbalanceamiento es intentar obtener distintos conjuntos de datos. Esto se realiza de la siguiente manera, o bien se aumentan las muestras de la clase minoritaria, dejando la clase mayoritaria conforme está, para que se igualen el total de muestras de uno y otro lado, o por el contrario, se reducirán las muestras de la clase mayoritaria y se dejaran como están las de la clase minoritaria.

**Sub-muestreo aleatorio**

Cuando eliminamos muestras de la clase mayoritaria aleatoriamente para igualarla a la clase minoritaria se produce un vacío, esto puede ayudar a mejorar el tiempo de ejecución de nuestro modelo, pero también baraja el inconveniente de haber eliminado una cantidad de información que podría resultar útil para nuestro modelo.

**Sobre-muestreo aleatorio**

En este caso al aumentar el número de muestras de la clase minoritaria para igualar las de la clase mayoritaria, utilizamos una técnica que no conduce a la pérdida de información, de otra forma, puede provocar un exceso de muestras que podrían alterar el resultado de nuestros algoritmos de clasificación.

## 3.16. RESOLVER PROBLEMA DE DESEQUILIBRIO

Como hemos comentado en el apartado anterior tenemos un problema de desequilibrio, que puede afectar a nuestros modelos de aprendizaje, por eso hemos decido aplicar la técnica descrita anteriormente de sub-muestreo aleatorio para solventar dicho problema.

La librería de sciket-learn implementa la función resample, que nos ayudará al muestreo descendente de la clase mayoritaria, eliminando los ejemplos de entrenamiento del conjunto de los datos.

```
#Sub-muestreo aleatorio
from sklearn.utils import resample
print ('Número de clases 1 antes:',X[y==1].shape[0])
X_subsampling, y_subsampling = resample(X[y==1],y[y==1],replace=True,
        n_samples=X[y==0].shape[0], random_state=123)
print ('Número de clases 1 después:',X_subsampling.shape[0])
```

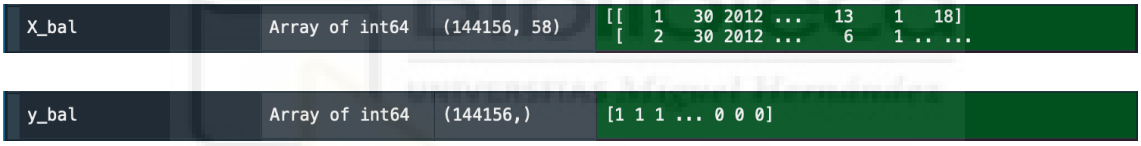
Resultado una vez ejecutado el código anterior:

```
Número de clases 0 antes: 392602
Número de clases 0 después: 72078
```

Combinamos las muestras de la clase 0 y la clase 1, y obtenemos una 'X_bal' con 144156 filas y 58 columnas, mientras la 'y_bal' tiene una dimensión de 144156 filas y 1 columna.

```
#Combinamos las muestras de una y otra clase
X_bal=np.vstack((X[y==0],X_subsampling))
y_bal=np.hstack((y[y==0],y_subsampling))
```

Resultado una vez ejecutado el código anterior:







## 3.17. ESCALADO DE CARACTERÍSTICAS

Es un paso vital en el preprocesado de los datos que en ocasiones es olvidado. Los árboles de decisión y los bosques aleatorios, son dos de los pocos algoritmos en los que no es necesario un escalado en nuestras características. Pero, en la mayoría de nuestros algoritmos de aprendizaje, funcionarán mejor si las características están en la misma escala. Existen dos formas distintas de llevar nuestros datos a la misma escala: estandarización y normalización.

### 3.17.1. NORMALIZACIÓN

Consiste en cambiar la escala de los datos de la característica para que tenga una longitud de 1. Es una técnica utilizada cuando tenemos datos dispersos con atributos en diferentes escalas. Para normalizar nuestros datos implementaremos la función Normalizer() de la librería scikit-learn, es conveniente destacar que se aplica solo en nuestra variable 'X_bal'.

**Scikit-Learn**

La librería de scikit-learn es una de las más importantes para el Aprendizaje Supervisado, a partir de ahora será una de las que vamos a utilizar para desarrollar nuestros algoritmos de Aprendizaje Supervisado. Se construyó con las librerías de Numpy, SciPy y Matplotlib, su uso será parecido a los anteriores. Scikit-learn proporciona al usuario una interfaz precisa para la implementación de los algoritmos, además de facilitar el uso y dotar los algoritmos de un alto rendimiento. Se ha hecho fundamental en la industria del Aprendizaje Automático en Python, consiguiendo gran popularidad entre los científicos de datos. También cuenta con una guía de usuario muy didáctica, que ameniza la comprensión y el estudio de los algoritmos de Aprendizaje Supervisado, como los algoritmos de Aprendizaje No Supervisados y que serán una parte fundamental de este trabajo, dicha guía de usuario se puede encontrar en la página:

https://scikit-learn.org/stable/user_guide.html#

```
#Normalización de los datos
from sklearm.preprocessing import Normalizer
X_normalizer = Normalizer().fit_transform(X_bal)
print (X_normalizer[0:5,:])
```

Resultado una vez ejecutado el código anterior

```
[[4.96931849e-04 1.49079555e-02 9.99826879e-01 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  4.96931849e-04 0.00000000e+00 0.00000000e+00 4.96931849e-04
  0.00000000e+00 0.00000000e+00 0.00000000e+00 4.96931849e-04
  0.00000000e+00 0.00000000e+00 4.96931849e-04 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  4.96931849e-04 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 4.96931849e-04
  0.00000000e+00 0.00000000e+00 4.96931849e-04 6.46011403e-03
  4.96931849e-04 8.94477327e-03]
 [9.93901617e-04 1.49085243e-02 9.99865027e-01 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  4.96950809e-04 0.00000000e+00 0.00000000e+00 0.00000000e+00
  4.96950809e-04 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 4.96950809e-04 0.00000000e+00 0.00000000e+00
  0.00000000e+00 4.96950809e-04 4.96950809e-04 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  4.96950809e-04 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 4.96950809e-04
  0.00000000e+00 0.00000000e+00 4.96950809e-04 2.98170485e-03
  4.96950809e-04 5.96340970e-03]
 [4.96942770e-04 1.49082831e-02 9.99848854e-01 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 4.96942770e-04 0.00000000e+00 0.00000000e+00
  4.96942770e-04 0.00000000e+00 0.00000000e+00 0.00000000e+00
  4.96942770e-04 0.00000000e+00 4.96942770e-04 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 4.96942770e-04 4.96942770e-04 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  4.96942770e-04 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  4.96942770e-04 0.00000000e+00 4.96942770e-04 6.46025601e-03
  4.96942770e-04 5.96331324e-03]
```

```
[9.93864065e-04 1.49079610e-02 9.99827250e-01 0.00000000e+00
 4.96932033e-04 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 4.96932033e-04 0.00000000e+00 0.00000000e+00 0.00000000e+00
 4.96932033e-04 0.00000000e+00 4.96932033e-04 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 4.96932033e-04
 0.00000000e+00 0.00000000e+00 4.96932033e-04 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 4.96932033e-04 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 4.96932033e-04
 0.00000000e+00 0.00000000e+00 4.96932033e-04 5.46625236e-03
 9.93864065e-04 9.44170862e-03]
[9.93909472e-04 1.49086421e-02 9.99872929e-01 4.96954736e-04
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 4.96954736e-04 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 4.96954736e-04 0.00000000e+00 0.00000000e+00 0.00000000e+00
 4.96954736e-04 0.00000000e+00 0.00000000e+00 0.00000000e+00
 4.96954736e-04 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 4.96954736e-04 4.96954736e-04 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 4.96954736e-04 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 4.96954736e-04
 0.00000000e+00 0.00000000e+00 4.96954736e-04 2.48477368e-03
 1.49086421e-03 4.47259262e-03]]
```

### 3.17.2. ESTANDARIZACIÓN

La estandarización de los datos se realiza cuando nuestras características se encuentran en diferentes escalas, esto puede provocar errores en ciertos algoritmos, porque las características no se parecen a los datos estándar normalmente distribuidos. Aunque la técnica de la normalización es muy utilizada, cuando necesitamos valores en un intervalo limitado, la estandarización puede ser mas práctica para los algoritmos de Aprendizaje Automático. La razón es que muchos modelos lineales, como la regresión logística y la SVM, inicializan los pesos a 0, o valores cercanos a 0. Mediante la estandarización, centramos las columnas de características a una media con una desviación estándar de 1, así las columnas de características toman la forma de una distribución normal, lo que hace mas fácil aprender los pesos a nuestro algoritmo. Además, la estandarización mantiene la información perteneciente a los 'outliers' (valore atípicos), permitiendo que el algoritmo sea menos sensible a ellos en comparación con la normalización, que escala los valores en un rango limitado de valores. Por estos motivos aplicaremos la estandarización a nuestro conjunto de datos antes que la normalización. Para estandarizar nuestros datos implementaremos la función StandarScaler() de la librería scikit-learn, es conveniente destacar que se aplica solo en nuestra variable 'X_bal'.

```
#Estandarización de los datos
from sklearn.preprocessing import StandardScaler
X_estandar = StandardScaler().fit_transform(X_bal)
print (X_estandar[0:5,:])
```

Resultado una vez ejecutado el código anterior:

```
[[-1.02753794 -0.63813427 -1.10510902 -0.40376569 -0.3981477  -0.36259944
  -0.42038796 -0.4174019  -0.41460246 -0.05079608 -0.39013639 -0.31421848
  -0.18173689 -0.62175721 -0.13942862 -0.24954917 -1.85607282 -0.09472561
  -0.05737568 -0.04753521  0.07446772 -0.05614566 -2.10651058  3.65899744
  -0.17197501 -0.25023853 -0.12152619  1.99193702 -0.07642067 -1.59196779
   0.48246959 -0.07380551 -0.13341392 -0.12457617 -0.36301569 -0.03604013
  -0.07246334 -0.12884336 -0.04234308 -0.13022748 -0.07313744 -0.63009754
  -0.03882761 -0.05450568  0.15132103 -0.05940906 -0.03670937 -0.05725386
  -0.09715579 -0.03194948 -0.04049444  0.13341392 -0.08479044 -0.05294733
   0.76230483 -0.30615558 -1.66127411  0.85714279]
 [ 0.30661586 -0.63813427 -1.10510902 -0.40376569 -0.3981477  -0.36259944
  -0.42038796 -0.4174019  -0.41460246 -0.05079608 -0.39013639 -0.31421848
  -0.18173689 -0.62175721 -0.13942862 -0.24954917  0.53877196 -0.09472561
  -0.05737568 -0.04753521  0.07446772 -0.05614566 -2.10651058 -0.27329891
  -0.17197501  3.99618712 -0.12152619 -0.5020239  -0.07642067  0.62815341
   0.48246959 -0.07380551 -0.13341392 -0.12457617 -0.36301569 -0.03604013
  -0.07246334 -0.12884336 -0.04234308 -0.13022748 -0.07313744 -0.63009754
  -0.03882761 -0.05450568  0.15132103 -0.05940906 -0.03670937 -0.05725386
  -0.09715579 -0.03194948 -0.04049444  0.13341392 -0.08479044 -0.05294733
   0.76230483 -1.10615336 -1.66127411 -0.29203321]
 [-1.02753794 -0.63813427 -1.10510902 -0.40376569 -0.3981477  -0.36259944
  -0.42038796 -0.4174019  -0.41460246 -0.05079608 -0.39013639 -0.31421848
  -0.18173689  1.60834485 -0.13942862 -0.24954917  0.53877196 -0.09472561
  -0.05737568 -0.04753521  0.07446772 -0.05614566  0.47471872 -0.27329891
  -0.17197501 -0.25023853 -0.12152619 -0.5020239  -0.07642067  0.62815341
   0.48246959 -0.07380551 -0.13341392 -0.12457617 -0.36301569 -0.03604013
  -0.07246334 -0.12884336 -0.04234308 -0.13022748 -0.07313744 -0.63009754
  -0.03882761 -0.05450568  0.15132103 -0.05940906 -0.03670937 -0.05725386
  -0.09715579 -0.03194948 -0.04049444 -7.49547001 11.79378203 -0.05294733
   0.76230483 -0.30615558 -1.66127411 -0.29203321]
 [ 0.30661586 -0.63813427 -1.10510902 -0.40376569  2.51163074 -0.36259944
  -0.42038796 -0.4174019  -0.41460246 -0.05079608 -0.39013639 -0.31421848
  -0.18173689 -0.62175721 -0.13942862 -0.24954917  0.53877196 -0.09472561
  -0.05737568 -0.04753521  0.07446772 -0.05614566  0.47471872 -0.27329891
  -0.17197501 -0.25023853 -0.12152619  1.99193702 -0.07642067 -1.59196779
   0.48246959 -0.07380551 -0.13341392 -0.12457617 -0.36301569 -0.03604013
  -0.07246334 -0.12884336 -0.04234308 -0.13022748 -0.07313744 -0.63009754
  -0.03882761 -0.05450568  0.15132103 -0.05940906 -0.03670937 -0.05725386
  -0.09715579 -0.03194948 -0.04049444  0.13341392 -0.08479044 -0.05294733
   0.76230483 -0.53472637 -1.36840996  1.04867212]
 [ 0.30661586 -0.63813427 -1.10510902  2.47668397 -0.3981477  -0.36259944
  -0.42038796 -0.4174019  -0.41460246 -0.05079608  2.56320614 -0.31421848
  -0.18173689 -0.62175721 -0.13942862 -0.24954917  0.53877196 -0.09472561
  -0.05737568 -0.04753521  0.07446772 -0.05614566 -2.10651058 -0.27329891
   5.81479846 -0.25023853 -0.12152619 -0.5020239  -0.07642067  0.62815341
   0.48246959 -0.07380551 -0.13341392 -0.12457617 -0.36301569 -0.03604013
  -0.07246334 -0.12884336 -0.04234308 -0.13022748 -0.07313744 -0.63009754
  -0.03882761 -0.05450568  0.15132103 -0.05940906 -0.03670937 -0.05725386
  -0.09715579 -0.03194948 -0.04049444  0.13341392 -0.08479044 -0.05294733
   0.76230483 -1.22043876 -1.07554581 -0.86662121]]
```

## 3.18. SEPARACIÓN DE LOS DATOS

Antes de la implementación de nuestros algoritmos de clasificación, procederemos a separar los datos en entrenamiento y prueba. Para esto utilizaremos la función train_test_split de la librería scikit-learn y emplearemos un 25% del conjunto de nuestros datos como datos de prueba.

```
#Separación de los datos de entrenamiento y prueba
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_bal, y_bal, test_size = 0.25, random_state = 0)
```

## 3.19. APRENDIZAJE SUPERVISADO

El Aprendizaje Supervisado se define como una parte del Aprendizaje Automático, donde a partir de datos anteriores etiquetados se construyen modelos para predecir los resultados de salida.

### 3.19.1 TIPOS DE ALGORITMOS SUPERVISADO

**Algoritmos de regresión**

Los algoritmos de regresión se caracterizan por ser capaces de predecir un valor. Consisten en generar un modelo que sea capaz de predecir el valor de una variable continúa a partir de algunas características de entrada. Los algoritmos de regresión son:

- Regresión lineal
- Regresión polinomial
- Vectores de soporte regresión
- Árboles de decisión regresión
- Bosques aleatorios regresión

Al tener nuestra variable objetivo como una variable categórica no utilizaremos ninguno de los algoritmos de regresión nombrados anteriormente.

**Algoritmos de clasificación**

Los algoritmos de clasificación tienen la particularidad de predecir una variable categórica, que puede ser una variable binaria si solo tiene dos clases o una variable multiclase cuando tiene más de dos clases. El funcionamiento de este tipo de algoritmos consiste en a partir de los datos de entrada etiquetados crear un modelo que permita etiquetar nuevos datos de entrada por si solo.

Cuando creamos un proyecto no sabemos de antemano qué algoritmo de clasificación va a ser más adecuado para este, por lo que debemos probar varios métodos y observar cuales son mejores para nuestro proyecto. Por eso intentaremos implementar varios algoritmos de clasificación para ver cúal se adapta mejor a nuestros datos:

- Regresión Logística
- K Vecinos más cercanos
- Maquinas de vectores de soporte
- Naive bayes
- Arboles de decisión clasificación
- Bosques aleatorios clasificación

## 3.20. MÉTRICAS DE RENDIMIENTO PARA ALGORITMOS DE CLASIFICACIÓN

Para los problemas de clasificación contamos con una gran cantidad de métricas de evaluación que se utilizan para evaluar las predicciones en este tipo de problemas. Nosotros utilizaremos principalmente las siguientes:

- Matriz de confusión
- Reporte de clasificación

**Matriz de confusión**

La matriz de confusión es una de las métricas más fáciles y sencillas de utilizar, se emplean para encontrar la precisión y la exactitud del modelo. Se aplica en problemas de clasificación donde la salida puede ser tanto de dos, como de más clases.

La matriz de confusión es una matriz con dos dimensiones ‘actual’ y ‘predicción’ y los conjuntos de clases en ambas dimensiones. Las filas indican la clase real y las columnas hacen referencia a la clase predicha.

La matriz de confusión no es una medida de rendimiento como tal, pero casi todas las métricas de evaluación se basan en ella.

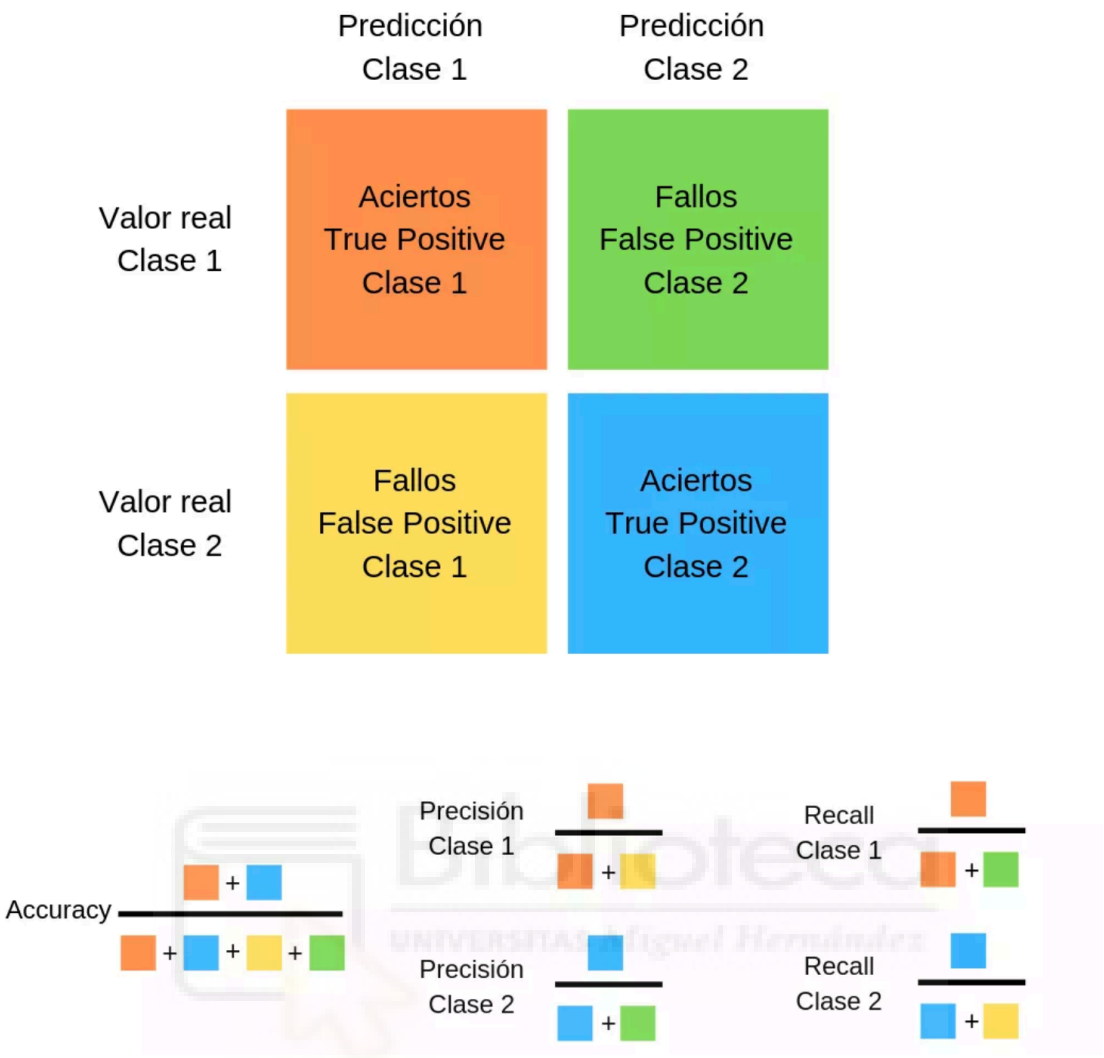



Ilustración 9: Matriz de confusión y métricas de evaluación.

**La exactitud (Accuracy)**

Es la relación entre las predicciones correctas y el número total de predicciones. O más simplemente, con qué frecuencia es correcto el clasificador.

**La precisión**

Es la relación entre las predicciones correctas y el número total de predicciones correctas previstas. Esto mide la precisión del clasificador a la hora de predecir casos positivos.

**La sensibilidad (Recall)**

Es la relación entre las predicciones positivas correctas y el número total de predicciones positivas. O más simplemente, cuán sensible es el clasificador para detectar instancias positivas. Esto también se conoce como la tasa verdadera positiva.

**El puntaje F1**

Es la medida armónica de la memoria y la precisión, con una puntuación más alta como mejor modelo.

**Reporte de clasificación**

El reporte de clasificación da una idea rápida de los conceptos que acabamos de citar anteriormente, precisión, sensibilidad, la puntuación F1 y el soporte de cada clase. El soporte de cada clase hace referencia al número de objetos que seleccionaremos en nuestro test o prueba de cada clase.

Ejemplo:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.68 | 0.66 | 0.67 | 1307 |
| 1 | 0.68 | 0.70 | 0.69 | 1344 |
| accuracy | | | 0.68 | 2651 |
| macro avg | 0.68 | 0.68 | 0.68 | 2651 |
| weighted avg | 0.68 | 0.68 | 0.68 | 2651 |

El promedio macro es el promedio de la media no ponderada por etiqueta.

El promedio ponderado es el promedio de la media ponderada por soporte por etiqueta.

**Área bajo la curva**

La curva AUC-ROC es una de la métrica de evaluación más utilizada para medir el rendimiento de evaluación de un algoritmo, sobre todo cuando nuestros datos se encuentran desbalanceados. ROC significa, características de funcionamiento del receptor y AUC área baja la curva.

## 3.21. APLICACIÓN DE LOS ALGORITMOS DE CLASIFICACIÓN

Aplicaremos los dos últimos apartados a nuestro proyecto, para determinar con que algoritmo se obtiene un mejor rendimiento de nuestros datos.

### 3.21.1. REGRESIÓN LOGÍSTICA

La Regresión Logística es un método utilizado normalmente para predecir clases binarias. Es decir, cuando el resultado o variable objetivo es de naturaleza dicotómica. Por ejemplo, detecciones de enfermedades o calcular la probabilidad de que ocurra un evento.

La Regresión Logística es uno de los algoritmos más simples y de los más empleados para la clasificación de dos clases. Es fácil su implementación y se puede usar como una base para cualquier problema de clasificación binaria. La Regresión Logística describe y hace una estimación de la relación entre una variable binaria dependiente y las variables independientes.

Lo primero que haremos para implementar este método será importar LogisticRegresion que está dentro de la librería de linear_model de sklearn. A continuación, entrenaremos el modelo utilizando las instrucciones fit y los datos tanto de 'X_train' como de 'y_train'. Por último, realizaremos una predicción, utilizando la instrucción predict con los datos de test.

```
#Regresión Logistica
from sklearn.linear_model import LogisticRegression
modelo = LogisticRegression()
modelo.fit(X_train, y_train)
y_pred = modelo.predict (X_test)
print('Datos de entrenamiento:')
print(y_test)
print('Datos de obtenidos en la predicción:')
print(y_pred)
```

Resultados obtenidos una vez ejecutado el código anterior:

```
Datos de entrenamiento:
[0 0 0 ... 0 1 1]
Datos obtenidos en la predicción:
[1 1 1 ... 1 1 1]
```

Efectivamente vemos que se ha ejecutado correctamente nuestro código, pero para ver las variables más detalladamente podemos acceder al variable

explorer y clicar sobre las variables que hemos creado ‘y_test’ e ‘y_pred’ que nos mostrará los siguientes resultados:


Ilustración 10: Comparación de datos reales vs predicciones.

Para observar donde coincide y donde no, nuestra variable ‘y_test’ con nuestra variable ‘y_pred’ nos podría llevar horas, por tanto, para evaluar nuestro algoritmo de clasificación utilizaremos las métricas de rendimiento.

La primera que vamos a observar y que suele ser ideal para hacernos una idea de cómo se está comportando nuestra predicción frente a nuestros datos de prueba, será la matriz de confusión. Utilizaremos la librería metrics de scikit learn y el módulo confusion_matrix para su implementación.

```
#Matriz de confusión
from sklearn.metrics import confusion_matrix
matriz = confusion_matrix(y_test, y_pred)
print('Matriz de confusión:')
print(matriz)
```

Resultado una vez ejecutado el código anterior:

```
Matriz de confusión:
[[10849  7286]
 [ 7312 10592]]
```

La matriz indica que tiene 10849 datos verdaderos positivos, hace referencia a los datos reales que eran 1 y el modelo los predijo correctamente. La matriz indica que hay 10592 datos verdaderos negativos, que hace referencia a los datos que eran 0 y el modelo los predijo correctamente. Podemos observar la cantidad de datos que nuestro algoritmo predijo erróneamente, en este caso 14598, de los que 7312 son falsos positivos, es decir, que el algoritmo predijo que eran 1 cuando en realidad eran 0, y 7286 falsos negativos cuando el algoritmo predijo que eran 0 cuando en realidad eran 1.

Ahora procederemos a calcular todas las métricas que existen para nuestro algoritmo de clasificación. Solamente con el cálculo de una de las métricas de evaluación sería suficiente para obtener el resultado del rendimiento de nuestro algoritmo, porque las demás métricas de evaluación se aproximarán o serán casi idénticas entre ellas, debido a que el dataset se encuentra balanceado. Para la implementación de dichas métricas de evaluación utilizaremos la librería scikit-learn dentro de metrics e iremos importando los módulos precisión_score, accuracy_score, recall_score y roc_auc_score según la métrica que queramos obtener en cada momento.

```
#Calculo de la precision del modelo
from sklearn.metrics import precision_score
precision = precision_score (y_test, y_pred)
print ('Precisión del modelo:')
print(precision)

#Exactitud del modelo
from sklearn.metrics import accuracy_score
exactitud = accuracy_score (y_test, y_pred)
print('Exactitud del modelo:')
print(exactitud)

#Calculo la sensibilidad del modelo
from sklearn.metrics import recall_score
sensibilidad = recall_score(y_test, y_pred)
print('Sensibilidad del modelo:')
print(sensibilidad)

#Calculo el puntaje F1
from sklearn.metrics import roc_auc_score
roc_auc = roc_auc_score(y_test, y_pred)
print('Curva ROC - AUC del modelo:')
print(roc_auc)
```

Resultado una vez ejecutado el código anterior:

```
Precisión del modelo:
0.5924600067121601
Exactitud del modelo:
0.5949388162823608
Sensibilidad del modelo:
0.5915996425379804
Curva ROC - AUC del modelo:
0.5949175494189765
```

Como comentábamos antes, se observa que no existe un cambio radical en la evaluación de nuestro algoritmo de clasificación. Todos los valores oscilan entorno al 59% por lo que se encuentran dentro de la dinámica de la que estábamos hablando.

### 3.21.2. K VECINOS MÁS CERCANOS

Es un algoritmo de aprendizaje no paramétrico, esto significa que no hace suposiciones explícitas sobre la forma funcional de nuestros datos, en otras palabras, dice que si nuestros datos no tienen una distribución normal y nuestro algoritmo tiende a modelar los datos de forma gaussiana, las predicciones del algoritmo serán malas. Por eso es tan importante estandarizar nuestros datos antes de aplicar ciertos algoritmos.

El algoritmo precisa de todos los datos de entrenamiento, que serán utilizados posteriormente en la fase de prueba. Esto provoca que la captación sea más rápida y la fase de prueba más lenta.

```python
#K Vecinos mas Cercanos
from sklearn.neighbors import KNeighborsClassifier
modelo = KNeighborsClassifier()
modelo.fit(X_train, y_train)
y_pred = modelo.predict (X_test)
print('Datos de entrenamiento:')
print(y_test)
print('Datos obtenidos en la predicción:')
print(y_pred)
```

Resultado una vez ejecutado el código anterior:

```
Datos de entrenamiento:
[0 0 0 ... 0 1 1]
Datos obtenidos en la predicción:
[1 0 1 ... 0 0 1]
```

Podemos observar los datos en la regresión logística del variable explorer, esto solo lo hacemos para comprobar que nuestro algoritmo se ha ejecutado correctamente.

Procedemos a la implementación de las métricas igual que en el apartado anterior:

```
#Matriz de confusión
from sklearn.metrics import confusion_matrix
matriz = confusion_matrix(y_test, y_pred)
print('Matriz de confusión:')
print(matriz)
```

Resultado una vez ejecutado el código anterior:

```
Matriz de confusión:
[[10407  7728]
 [ 7899 10005]]
```

Observamos que tenemos un mayor valor de errores en nuestra matriz de confusión que en la regresión logística, por tanto, tendremos peores resultados en nuestras métricas de evaluación.

Ahora evaluaremos los resultados. Utilizaremos esta vez el reporte de clasificación disponible en la librería scikit-learn.

```
#Reporte de clasificación
from sklearn.metrics import classification_report
reporte = classification_report(y_test, y_pred)
print(reporte)
```

Resultado una vez ejecutado el código anterior:

```
              precision    recall  f1-score   support

           0       0.57      0.57      0.57     18135
           1       0.56      0.56      0.56     17904

    accuracy                           0.57     36039
   macro avg       0.57      0.57      0.57     36039
weighted avg       0.57      0.57      0.57     36039
```

Se observa cómo obtenemos peores resultados que en el algoritmo anterior, las métricas de evaluación están entorno al 57% de valores acertados en nuestra predicción, iremos más rápido aplicando el reporte de clasificación, dado que nuestros datos están balanceados, no será necesario aplicar

métricas como la curva roc-auc que como hemos comentado será similar a las demás.

### 3.21.3. MÁQUINAS DE VECTORES DE SOPORTE

Las máquinas de vectores de soporte buscan la línea que mejor separa dos clases. Las características de datos que están más cerca de la línea que mejor separa las clases se denomina vectores de soporte, que influyen en la ubicación de la línea. Este tipo de algoritmos ofrecen una alta precisión comparándolo con otros algoritmos de clasificación.

```
#Maquinas de Vectores de Soporte
from sklearn.svm import SVC
modelo = SVC()
modelo.fit(X_train, y_train)
y_pred=modelo.predict(X_test)
print('Datos de entrenamiento:')
print(y_test)
print('Datos obtenidos en la predicción:')
print(y_pred)
```

Resultado una vez ejecutado el código anterior:

```
Datos de entrenamiento:
[0 0 0 ... 0 1 1]
Datos obtenidos en la predicción:
[1 1 1 ... 1 1 1]
```

Igual que anteriormente evaluamos nuestro modelo mediante la matriz de confusión y el reporte de clasificación.

```
#Matriz de confusión
from sklearn.metrics import confusion_matrix
matriz = confusion_matrix(y_test, y_pred)
print('Matriz de confusión:')
print(matriz)
```

Resultado una vez ejecutado el código anterior:

```
Matriz de confusión:
[[11326  6809]
 [ 7038 10866]]
```

De momento es la matriz de confusión la que más datos ha acertado en la predicción, superando a la regresión logística.

```
#Reporte de clasificación
from sklearn.metrics import classification_report
reporte = classification_report(y_test, y_pred)
print(reporte)
```

Resultado una vez ejecutado el código anterior:

```
              precision    recall  f1-score   support

           0       0.62      0.62      0.62     18135
           1       0.61      0.61      0.61     17904

    accuracy                           0.62     36039
   macro avg       0.62      0.62      0.62     36039
weighted avg       0.62      0.62      0.62     36039
```

Nuestros datos de precisión, exactitud y f1-score, muestran un 62% de acierto en nuestras predicciones. Por el momento, éste es el algoritmo con el que mejor resultados estamos obteniendo.



### 3.21.4. NAIVE BAYES

Es una técnica de clasificación estadística que se basa en el teorema de bayes. Es un algoritmo simple, rápido, preciso y fiable, además proporciona una alta precisión y velocidad cuando trabajamos con grandes volúmenes de datos.

```
#Naive Bayes
from sklearn.naive_bayes import GaussianNB
modelo = GaussianNB()
modelo.fit(X_train, y_train)
y_pred=modelo.predict(X_test)
print('Datos de entrenamiento:')
print(y_test)
print('Datos obtenidos en la predicción:')
print(y_pred)
```

Resultado una vez ejecutado el código anterior:

```
Datos de entrenamiento:
[0 0 0 ... 0 1 1]
Datos obtenidos en la predicción:
[1 1 0 ... 1 1 1]
```

Vamos a proceder a verificar el error de nuestro modelo con la matriz de confusión y el reporte de clasificación.

```
#Matriz de confusión
from sklearn.metrics import confusion_matrix
matriz = confusion_matrix(y_test, y_pred)
print('Matriz de confusión:')
print(matriz)
```

Resultado una vez ejecutado el código anterior:

```
Matriz de confusión:
[[ 4658 13477]
 [ 3266 14638]]
```

Los resultados obtenidos con este algoritmo en la matriz de confusión, como podemos observar, no son muy buenos, ya que predice la mayoría de resultados como si fueran 0, acertando casi todos los resultados que dan 0, pero fallado en casi todos los que deberían de dar 1.

```
#Reporte de clasificación
from sklearn.metrics import classification_report
reporte = classification_report(y_test, y_pred)
print(reporte)
```

Resultado una vez ejecutado el código anterior:

```
              precision    recall  f1-score   support

           0       0.59      0.26      0.36     18135
           1       0.52      0.82      0.64     17904

    accuracy                           0.54     36039
   macro avg       0.55      0.54      0.50     36039
weighted avg       0.55      0.54      0.50     36039
```

Vemos que las demás métricas de evaluación confirman lo que estábamos viendo en la matriz de confusión, arrojando valores que no superan el 55% de aciertos en ninguno de los casos, por lo que determinamos que este algoritmo de clasificación no es adecuado para nuestro conjunto de datos.

### 3.21.5. ÁRBOLES DE DECISIÓN. CLASIFICACIÓN

Los árboles de decisión son un tipo de algoritmo de Aprendizaje Supervisado que se utilizan normalmente en problemas de clasificación, aunque también los podemos utilizar en problemas de regresión. La técnica consiste en subdividir

en conjuntos homogéneos los datos basándose en las variables de entradas más significativas. El árbol se encarga de seleccionar las características más relevantes, para formar así los mejores grupos posibles de nuestra muestra. Se evaluarán todas las variables de entrada y todos los puntos de división, y se eligirá el mejor resultado obtenido.

```
#Árboles de clasificación
from sklearn.tree import DecisionTreeClassifier
modelo = DecisionTreeClassifier()
modelo.fit(X_train, y_train)
y_pred=modelo.predict(X_test)
print('Datos de entrenamiento:')
print(y_test)
print('Datos obtenidos en la predicción:')
print(y_pred)
```

Resultado una vez ejecutado el código anterior:

```
Datos de entrenamiento:
[0 0 0 ... 0 1 1]
Datos obtenidos en la predicción:
[1 0 0 ... 1 1 1]
```

Evaluamos el error implementando la matriz de confusión y el reporte de clasificación:

```
#Matriz de confusión
from sklearn.metrics import confusion_matrix
matriz = confusion_matrix(y_test, y_pred)
print('Matriz de confusión:')
print(matriz)
```

Resultado una vez ejecutado el código anterior:

```
Matriz de confusión:
[[10855  7280]
 [ 7416 10488]]
```

Vemos que la matriz de confusión sigue la dinámica de los demás algoritmos de clasificación, exceptuando el algoritmo anterior de Naive Bayes con el que obteníamos valores dispares.

```
#Reporte de clasificación
from sklearn.metrics import classification_report
reporte = classification_report(y_test, y_pred)
print(reporte)
```

Resultado una vez ejecutado el código anterior:

```
              precision    recall  f1-score   support

           0       0.59      0.60      0.60     18135
           1       0.59      0.59      0.59     17904

    accuracy                           0.59     36039
   macro avg       0.59      0.59      0.59     36039
weighted avg       0.59      0.59      0.59     36039
```

Obtenemos un resultado entorno al 59% de acierto, en las predicciones realizadas en las métricas de evaluación.

### 3.21.6. BOSQUES ALEATORIOS. CLASIFICACIÓN

Los bosques aleatorios funcionan mediante métodos de reducción dimensional, tratan valores perdidos, valores atípicos, y hacen un trabajo bastante eficaz. Se basan principalmente en un método de aprendizaje por conjuntos, donde varios modelos débiles se combinan para formar un modelo poderoso. Se denomina bosque porque a diferencia del apartado anterior, de árboles aleatorios, aquí se utilizarán varios árboles en lugar de un solo árbol. Para clasificar un nuevo objeto, cada árbol da una clasificación y se dice que el árbol vota por esa clase. El bosque elige la clasificación con más votos sobre los demás árboles en el bosque.

```
#Bosques aleatorios clasificación
from sklearn.ensemble import RandomForestClassifier
modelo = RandomForestClassifier()
modelo.fit(X_train, y_train)
y_pred=modelo.predict(X_test)
print('Datos de entrenamiento:')
print(y_test)
print('Datos obtenidos en la predicción:')
print(y_pred)
```

Resultado una vez ejecutado el código anterior:

```
Datos de entrenamiento:
[0 0 0 ... 0 1 1]
Datos obtenidos en la predicción:
[1 1 0 ... 1 1 1]
```

Procedemos a implementar las funciones para la evaluación de nuestro algoritmo, la matriz de confusión y el reporte de clasificación.

```
#Matriz de confusión
from sklearn.metrics import confusion_matrix
matriz = confusion_matrix(y_test, y_pred)
print('Matriz de confusión:')
print(matriz)
```

Resultado una vez ejecutado el código anterior:

```
Matriz de confusión:
[[11468  6667]
 [ 6669 11235]]
```

Observamos que hemos obtenido mejores resultados en nuestra matriz de confusión, que en las matrices de los demás algoritmos de clasificación.

```
#Reporte de clasificación
from sklearn.metrics import classification_report
reporte = classification_report(y_test, y_pred)
print(reporte)
```

Resultado una vez ejecutado el código anterior:

```
              precision    recall  f1-score   support

           0       0.63      0.63      0.63     18135
           1       0.63      0.63      0.63     17904

    accuracy                           0.63     36039
   macro avg       0.63      0.63      0.63     36039
weighted avg       0.63      0.63      0.63     36039
```

Se puede ver que las demás métricas de evaluación arrojan valores muy buenos, como estábamos viendo en la matriz de confusión, por lo que, el bosque aleatorio de clasificación es el algoritmo que mejor resultado nos ha dado en las predicciones, 63% de valores predichos correctamente.

## 3.22. OPTIMIZACIÓN DE LOS HIPER-PARÁMETROS

Hasta ahora hemos implementado todos nuestros algoritmos sin ningún parámetro, simplemente los hemos dejado por defecto, pero existe la posibilidad de añadir y ajustar los parámetros existentes para cada uno de nuestros algoritmos, además es una buena manera de mejorar el rendimiento de dichos algoritmos.

Existen muchos valores que podemos dar a nuestros parámetros, por lo que no sabremos a ciencia cierta con qué parámetros funcionara mejor nuestro algoritmo, para esto, podemos implementar la búsqueda de cuadrículas de la clase GridSearchCV de scikit-learn, que consiste en un paradigma de búsqueda exhaustiva de fuerza bruta, donde especificamos una lista de valores para diferentes parámetros. El ordenador evaluará la combinación de valores que obtengan un mejor rendimiento en nuestro algoritmo.

Aunque la búsqueda de cuadrículas es una buena manera de encontrar cuáles serán los mejores conjuntos de parámetros para nuestro modelo, en ocasiones resulta computacionalmente muy costosa. Por ello, nosotros utilizaremos un enfoque alternativo, denominado búsqueda aleatoria que pertenece a la clase RandomizedSearchCV de scikit-learn, con la cual podemos realizar combinaciones aleatorias de parámetros, a partir del muestreo de distribuciones con una dotación específica.

Vamos a proceder a la implementación de la búsqueda aleatoria en el algoritmo que mejor resultados nos ha ofrecido (Bosques aleatorios clasificación). Podríamos aplicarlo en cualquiera de los explicados anteriormente. Comenzaremos descubriendo cuantos parámetros tiene nuestro algoritmo y que valores vienen asignados por defecto mediante la función get_params().

```
modelo = RandomForestClassifier ()
print (modelo.get_params ())
```

Resultado una vez ejecutado el código anterior:

```
{'bootstrap': True, 'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'gini', 'max_depth':
None, 'max_features': 'auto', 'max_leaf_nodes': None, 'max_samples': None,
'min_impurity_decrease': 0.0, 'min_impurity_split': None, 'min_samples_leaf': 1,
'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100, 'n_jobs': None,
'oob_score': False, 'random_state': None, 'verbose': 0, 'warm_start': False}
```

Vemos que aparecen bastantes parámetros que podemos configurar, pero debemos configurar aquellos que sean más relevantes y produzcan un mayor impacto en nuestro rendimiento, dicho lo cual, vamos a editar los que la página de scikit-learn considera los más importantes:

- n_estimadores: Número de árboles en el bosque.

- max_features: Número máximo de características consideradas para dividir un nodo.
- max_depth: Número máximo de niveles en cada árbol de decisión.
- min_samples_split: Número mínimo de puntos de datos colocados en un nodo antes de que el nodo se divida.
- min_samples_leaf: Número mínimo de puntos de datos permitidos en un nodo hoja.
- bootstrap: Método para muestrear puntos de datos (con o sin reemplazo).

Vamos a crear una cuadrícula de parámetros para muestrear durante el ajuste, porque es necesario para la utilización de RandomizedSearchCV.

```
#Número de árboles en bosque aleatorio
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
## Número de características a considerar en cada división
max_features = ['auto', 'sqrt']
#Número máximo de niveles en el árbol
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
#Número mínimo de muestras necesarias para dividir un nodo
min_samples_split = [2, 5, 10]
#Número mínimo de muestras requeridas en cada nodo de hoja
min_samples_leaf = [1, 2, 4]
#Método de selección de muestras para entrenar cada árbol
bootstrap = [True, False]
#Creamos la cuadrícula aleatoria
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
print(random_grid)
```

Resultado una vez ejecutado el código anterior:

```
{'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000], 'max_features': ['auto', 'sqrt'],
'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None], 'min_samples_split': [2, 5, 10],
'min_samples_leaf': [1, 2, 4], 'bootstrap': [True, False]}
```

En cada interacción nuestro algoritmo seleccionará una combinación distinta para cada parámetro, hay 2 * 12 * 2 * 3 * 3 * 10 = 4320 configuraciones posibles, sin embargo, el beneficio de una búsqueda aleatoria es que no estamos probando todas las combinaciones, sino que, seleccionaremos al azar y observaremos todos los valores que toman nuestros parámetros.

Ahora, crearemos la búsqueda aleatoria y la ajustaremos como hemos hecho en cualquier modelo en scikit-learn. Los parámetros más importantes en

RandomizedSearchCV son n_iter, que se encarga de la cantidad de combinaciones diferentes a probar, en nuestro caso serán 10, en donde 9 de esas 10 interacciones servirán para el entrenamiento del modelo y una utilizada como prueba para la evaluación del modelo, el otro parámetro más relevante es la CV (validación cruzada), que indica el número de divisiones aleatorias en las que dividiremos el conjunto de nuestros datos. A estas divisiones se les denomina pliegues, esto lo haremos para evitar el problema del sobreajuste que se puede provocar si entrenamos muchas veces un mismo conjunto de datos, ya que produciría excelentes resultados en los datos de prueba finales, alejándose de la realidad. Esto se debe a que ya tiene asociados los datos de entrada con su respuesta, lo que acaba siendo al final un proceso casi automático. El problema vendría, por ejemplo, al introducir unos valores distintos en la base de datos de prueba, dado que, el problema al no tener información de estos, es incapaz de sacar conclusiones por sí solo y provocaría errores.

La cantidad de pliegues que vamos a usar para nuestra validación cruzada será, 3 pliegues de 10 interacciones cada uno. Otras interaciones cubrirán un espacio de búsqueda más amplio y más pliegues de CV reducen las posibilidades de sobreajuste, pero al aumentar cada uno, aumentará el tiempo de ejecución. El Aprendizaje Automático es un campo de compensaciones, y el rendimiento frente al tiempo es uno de los más fundamentales.

Ejemplo gráfico de visualización de una validación cruzada de k interacciones con k pliegues, siendo el valor de k=10. Esta es una técnica de remuestreo sin reemplazo, la ventaja de esto es que cada punto de muestra se utilizará para entrenar y validar solamente una vez. Utilizaremos k-1 para el entrenamiento y una para la validación.

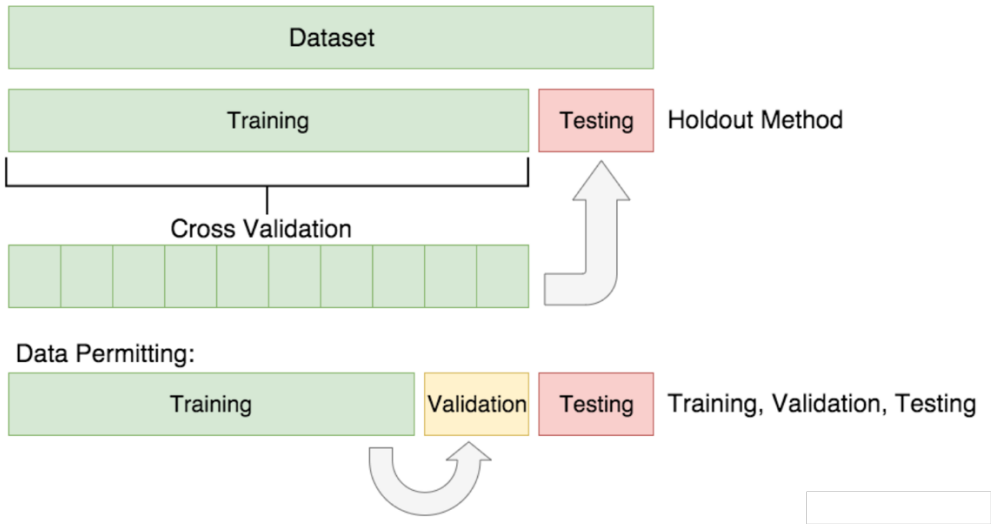



Ilustración 11: División de los datos en la validación cruzada.

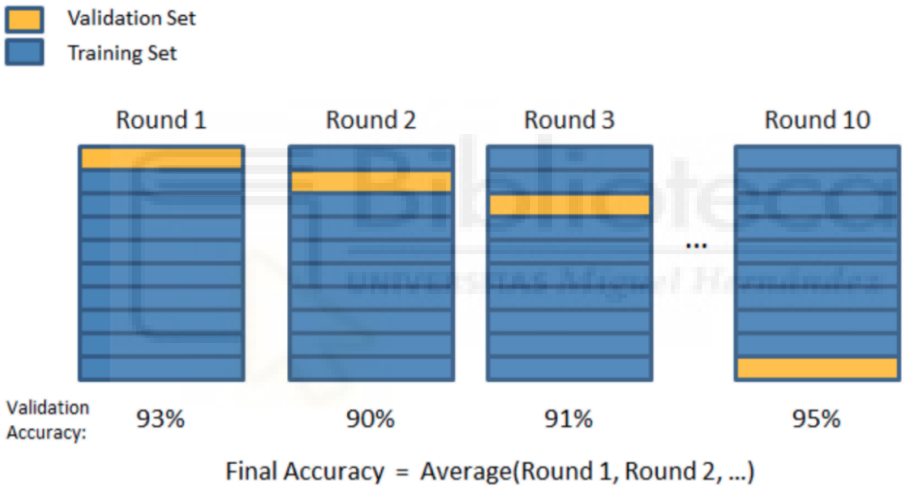



Ilustración 12: Evaluación de las divisiones en la validación cruzada.

```
#Usamos la cuadrícula aleatoria para buscar los mejores hiperparámetros
#Primero creamos el modelo base para ajustar
modelo = RandomForestClassifier()
#Búsqueda aleatoria de parámetros, usando validación cruzada triple
#Búsqueda en 10 combinaciones diferentes y usando todos los núcleos disponibles
rf_random = RandomizedSearchCV(estimator=modelo,
                               param_distributions=random_grid,
                               cv=3, n_iter=10,
                               scoring = 'accuracy',
                               n_jobs = -1, verbose = 2,
                               return_train_score = True,
                               random_state=42)
#Ajustamos el modelo de búsqueda aleatorio
rf_random.fit(X_train, y_train)
print(rf_random.best_params_)
print(rf_random.best_score_)
```

Resultado una vez ejecutado el código anterior:

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[CV] n_estimators=600, min_samples_split=10, min_samples_leaf=4, max_features=sqrt, max_depth=90, bootstrap=False
[CV]  n_estimators=600, min_samples_split=10, min_samples_leaf=4, max_features=sqrt, max_depth=90, bootstrap=False, total= 2.3min
[CV] n_estimators=1400, min_samples_split=5, min_samples_leaf=1, max_features=sqrt, max_depth=30, bootstrap=True
[CV]  n_estimators=1400, min_samples_split=5, min_samples_leaf=1, max_features=sqrt, max_depth=30, bootstrap=True, total= 8.5min
[CV] n_estimators=1200, min_samples_split=2, min_samples_leaf=4, max_features=auto, max_depth=100, bootstrap=True
[CV]  n_estimators=1200, min_samples_split=2, min_samples_leaf=4, max_features=auto, max_depth=100, bootstrap=True, total= 3.2min
[CV] n_estimators=600, min_samples_split=10, min_samples_leaf=4, max_features=sqrt, max_depth=90, bootstrap=False
[CV]  n_estimators=600, min_samples_split=10, min_samples_leaf=4, max_features=sqrt, max_depth=90, bootstrap=False, total= 2.3min
[CV] n_estimators=1000, min_samples_split=10, min_samples_leaf=1, max_features=auto, max_depth=80, bootstrap=False
[CV]  n_estimators=1000, min_samples_split=10, min_samples_leaf=1, max_features=auto, max_depth=80, bootstrap=False, total= 8.6min
[CV] n_estimators=1200, min_samples_split=2, min_samples_leaf=4, max_features=auto, max_depth=100, bootstrap=True
[CV]  n_estimators=1200, min_samples_split=2, min_samples_leaf=4, max_features=auto, max_depth=100, bootstrap=True, total= 3.2min
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed: 24.7min finished
[CV] n_estimators=600, min_samples_split=10, min_samples_leaf=4, max_features=sqrt, max_depth=90, bootstrap=False
[CV]  n_estimators=600, min_samples_split=10, min_samples_leaf=4, max_features=sqrt, max_depth=90, bootstrap=False, total= 2.3min
[CV] n_estimators=1000, min_samples_split=10, min_samples_leaf=1, max_features=auto, max_depth=80, bootstrap=False
[CV]  n_estimators=1000, min_samples_split=10, min_samples_leaf=1, max_features=auto, max_depth=80, bootstrap=False, total= 8.6min
[CV] n_estimators=2000, min_samples_split=5, min_samples_leaf=2, max_features=auto, max_depth=50, bootstrap=True
[CV]  n_estimators=2000, min_samples_split=5, min_samples_leaf=2, max_features=auto, max_depth=50, bootstrap=True, total= 7.0min
[CV] n_estimators=600, min_samples_split=2, min_samples_leaf=2, max_features=auto, max_depth=60, bootstrap=False
[CV]  n_estimators=600, min_samples_split=2, min_samples_leaf=2, max_features=auto, max_depth=60, bootstrap=False, total= 2.7min
[CV] n_estimators=1000, min_samples_split=10, min_samples_leaf=1, max_features=auto, max_depth=80, bootstrap=False
[CV]  n_estimators=1000, min_samples_split=10, min_samples_leaf=1, max_features=auto, max_depth=80, bootstrap=False, total= 9.1min
[CV] n_estimators=2000, min_samples_split=5, min_samples_leaf=2, max_features=auto, max_depth=50, bootstrap=True
[CV]  n_estimators=2000, min_samples_split=5, min_samples_leaf=2, max_features=auto, max_depth=50, bootstrap=True, total= 7.0min
[CV] n_estimators=200, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=50, bootstrap=True
[CV]  n_estimators=200, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=50, bootstrap=True, total=  33.7s
[CV] n_estimators=600, min_samples_split=2, min_samples_leaf=2, max_features=auto, max_depth=60, bootstrap=False
[CV]  n_estimators=600, min_samples_split=2, min_samples_leaf=2, max_features=auto, max_depth=60, bootstrap=False, total= 6.2min
[CV] n_estimators=400, min_samples_split=10, min_samples_leaf=1, max_features=sqrt, max_depth=60, bootstrap=False
[CV]  n_estimators=400, min_samples_split=10, min_samples_leaf=1, max_features=sqrt, max_depth=60, bootstrap=False, total= 2.1min
[CV] n_estimators=2000, min_samples_split=2, min_samples_leaf=2, max_features=auto, max_depth=50, bootstrap=False
[CV]  n_estimators=2000, min_samples_split=2, min_samples_leaf=2, max_features=auto, max_depth=50, bootstrap=False, total= 9.9min
[CV] n_estimators=200, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=50, bootstrap=True
[CV]  n_estimators=200, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=50, bootstrap=True, total=  33.9s
[CV] n_estimators=1400, min_samples_split=5, min_samples_leaf=1, max_features=sqrt, max_depth=30, bootstrap=True
[CV]  n_estimators=1400, min_samples_split=5, min_samples_leaf=1, max_features=sqrt, max_depth=30, bootstrap=True, total= 8.5min
[CV] n_estimators=2000, min_samples_split=2, min_samples_leaf=2, max_features=auto, max_depth=50, bootstrap=False
[CV]  n_estimators=2000, min_samples_split=2, min_samples_leaf=2, max_features=auto, max_depth=50, bootstrap=False, total=11.2min
[CV] n_estimators=200, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=50, bootstrap=True
[CV]  n_estimators=200, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=50, bootstrap=True, total=  33.9s
[CV] n_estimators=1400, min_samples_split=5, min_samples_leaf=1, max_features=sqrt, max_depth=30, bootstrap=True
[CV]  n_estimators=1400, min_samples_split=5, min_samples_leaf=1, max_features=sqrt, max_depth=30, bootstrap=True, total= 8.5min
[CV] n_estimators=2000, min_samples_split=2, min_samples_leaf=2, max_features=auto, max_depth=50, bootstrap=False
[CV]  n_estimators=2000, min_samples_split=2, min_samples_leaf=2, max_features=auto, max_depth=50, bootstrap=False, total=11.2min
[CV] n_estimators=600, min_samples_split=2, min_samples_leaf=2, max_features=auto, max_depth=60, bootstrap=False
[CV]  n_estimators=600, min_samples_split=2, min_samples_leaf=2, max_features=auto, max_depth=60, bootstrap=False, total= 2.7min
[CV] n_estimators=400, min_samples_split=10, min_samples_leaf=1, max_features=sqrt, max_depth=60, bootstrap=False
[CV]  n_estimators=400, min_samples_split=10, min_samples_leaf=1, max_features=sqrt, max_depth=60, bootstrap=False, total= 5.5min
[CV] n_estimators=400, min_samples_split=10, min_samples_leaf=1, max_features=sqrt, max_depth=60, bootstrap=False
[CV]  n_estimators=400, min_samples_split=10, min_samples_leaf=1, max_features=sqrt, max_depth=60, bootstrap=False, total= 1.9min
[CV] n_estimators=200, min_samples_split=5, min_samples_leaf=2, max_features=sqrt, max_depth=10, bootstrap=True
[CV]  n_estimators=200, min_samples_split=5, min_samples_leaf=2, max_features=sqrt, max_depth=10, bootstrap=True, total=  18.4s
[CV] n_estimators=200, min_samples_split=5, min_samples_leaf=2, max_features=sqrt, max_depth=10, bootstrap=True
[CV]  n_estimators=200, min_samples_split=5, min_samples_leaf=2, max_features=sqrt, max_depth=10, bootstrap=True, total=  18.4s
[CV] n_estimators=200, min_samples_split=5, min_samples_leaf=2, max_features=sqrt, max_depth=10, bootstrap=True
[CV]  n_estimators=200, min_samples_split=5, min_samples_leaf=2, max_features=sqrt, max_depth=10, bootstrap=True, total=  18.1s
[CV] n_estimators=1200, min_samples_split=2, min_samples_leaf=4, max_features=auto, max_depth=100, bootstrap=True
[CV]  n_estimators=1200, min_samples_split=2, min_samples_leaf=4, max_features=auto, max_depth=100, bootstrap=True, total= 3.1min
[CV] n_estimators=2000, min_samples_split=5, min_samples_leaf=2, max_features=auto, max_depth=50, bootstrap=True
[CV]  n_estimators=2000, min_samples_split=5, min_samples_leaf=2, max_features=auto, max_depth=50, bootstrap=True, total= 8.7min
```

```
{'n_estimators': 2000, 'min_samples_split': 2, 'min_samples_leaf': 2, 'max_features': 'auto', 'max_depth': 50,
'bootstrap': False}
0.6278476095341158
```

Si nos fijamos en los resultados obtenidos, podemos ver que a pesar de haber implementado un algoritmo de selección de parámetros más rápido computacionalmente hablando, aun así, nos ha llevado 24.7 minutos. El proceso ha determinado que el modelo que mejor precisión de validación cruzada a obtenido ha sido del 0.6278% y los valores de los parámetros de dicho modelo son los siguientes: n_estimators=2000, min_samples_splitz=2, min_samples_leaf=2, max_features='auto', max_depth=50 y bootstrap=False. Por último, vamos a utilizar el conjunto de prueba independiente para estimar el rendimiento del modelo mejor seleccionado, al cual podemos acceder con el atributo best_estimator. Esto lo hacemos para ver el rendimiento del conjunto de pruebas y cómo de bien funcionaría nuestro modelo si se desplegará en el mundo real.

```
#Conjunto de datos de prueba con el mejor modelo seleccionado
clf=rf_random.best_estimator_
clf.fit(X_train,y_train)
print('Test accuracy:%.3f'%clf.score(X_test,y_test))
```

Resultado una vez ejecutado el código anterior:

```
Test accuracy:0.642
```

Como conclusión podemos decir, que es posible mejorar el modelo ajustando los parámetros, por lo menos en el caso del algoritmo del bosque de clasificación aleatorio, ya que los resultados obtenidos en comparación con el rendimiento del algoritmo sin ajustar los hiperparametros tienen una mejora de un 2,2 %, respecto al algoritmo en el que no hemos ajustado los hiperparametros, simplemente los hemos dejado por defecto. De manera que, podemos concluir diciendo que este método ha sido efectivo para obtener mejores resultados.



## 3.23. APRENDIZAJE NO SUPERVISADO

Los algoritmos de Aprendizaje No Supervisados buscan patrones en un conjunto de datos cuyos resultados son desconocidos o lo que es lo mismo son datos que no tienen una salida asignada con los datos de entrada. A diferencia de los algoritmos de Aprendizaje Supervisado, donde se requiere un entrenamiento de los datos antes de proceder a la realización de sus predicciones en este tipo de lenguaje, esto no es necesario, debido a que no hay forma de ver donde se produce el error en nuestras predicciones, porque desconocemos la salida de los datos. En cambio, el aprendizaje sin supervisión puede utilizarse para descubrir una cierta estructura imperceptible en los datos a primera vista.

Los algoritmos se asocian o agrupan para buscar similitudes entre los datos que carecen de una estructura o una clasificación. El término ‘no supervisado’ hace referencia a que el algoritmo carece de una revisión y no se encuentra guiado como ocurre con el Aprendizaje Supervisado.

Tabla 9: Ventajas y desventajas del Aprendizaje No Supervisado.

| VENTAJAS | DESVENTAJAS |
|---|---|
| El Aprendizaje No Supervisado encuentra todo tipo de patrones desconocidos en los datos. | A veces los patrones descubiertos son aproximaciones deficientes de lo que se puede lograr con el Aprendizaje Supervisado. |
| Encuentran características que pueden ser útiles para la categorización. | Es computacionalmente complejo. |
| Es más fácil descubrir datos no etiquetados. | Al no estar etiquetados los datos es difícil saber si los algoritmos están cumpliendo su finalidad. |
| Algoritmos fiables. | Algoritmos poco precisos debido a que los datos de entrada no son conocidos por lo que la máquina tiene que etiquetarlos por sí misma. |
| Se pueden realizar clasificaciones de los datos, aunque el conjunto de los datos sea elevado. | No se puede obtener mucha información sobre la clasificación de dichos datos. |

### 3.23.1. TIPOS DE APRENDIZAJE NO SUPERVISADO

**Agrupamiento**

El agrupamiento consiste en encontrar una estructura o patrón en una colección de datos no categorizados. Los algoritmos de agrupamiento (o clústeres), procesarán los datos buscando grupos o clústeres naturales en

ellos, es posible editar el número de grupos que queremos crear. Existen varios tipos de agrupamientos:

**Exclusivo**

Este método se caracteriza porque los datos de agrupamiento sólo pueden pertenecer a un cierto grupo o cluster. Ejemplo: K Means.

**Aglomerativo**

Esta técnica tiene la particularidad de que cada dato es un clúster. Las uniones iterativas entre los dos clústeres más cercanos reducen el número de clústeres. Ejemplo: Agrupación jerárquica.

**Solapamiento**

Aquí cada punto puede pertenecer a varios grupos, pero es posible medir el grado de implicación que tiene ese punto en cada grupo, asociándolo con lo que denominamos un valor de membresía. Ejemplo: Fuzzy C-Means.

**Probabilístico**

Los grupos se crean en relación a su distribución de probabilidad.

**Asociación**

Las reglas de asociación se utilizan para ver cómo se relacionan nuestras características entre sí y ver qué cosas pueden tener en común y cuáles no. Por ejemplo, si imaginamos las estadísticas de un jugador de baloncesto, y nos fijamos en su porcentaje de triples vemos que este aumenta cuando la característica de lanzamientos de triples sin defensor aumenta, a priori vemos que tiene cierta correlación, y podríamos utilizar las reglas de asociación para ver con más detenimiento esta implicación y sacar más conclusiones.

- Soporte (Support): Mide la frecuencia con la que el antecedente y el consecuente ocurren juntos en todas las instancias de nuestro conjunto de datos, se mide con un porcentaje y los valores del soporte se encuentran entre 0 y 1.

- Confianza (Confidence): Es la probabilidad de que las instancias del antecedente contengan a las del consecuente, su fórmula matemática es la siguiente:

  X=Antecedente

  Y=Consecuente

$$\text{confianza}(X=>Y) = \frac{\text{soporte}(\text{unión}(X,Y))}{\text{soporte}(X)},$$

- Mejora de confianza (lift): El indicador lift es una medida de la relación entre el antecedente cuando este va con el consecuente y cómo esto afecta en nuestro proyecto, su valor oscila entorno a 1.
  Un valor de lift = 1 indica que ese conjunto compuesto por antecedente y consecuente aparece una cantidad de veces acorde a lo esperado bajo condiciones de independencia, es decir, cuando ambos aparecen por separado. Tanto el antecedente como el consecuente no mantienen ninguna correlación son totalmente independientes.
  Un valor de lift > 1 hace referencia a que el conjunto aparece una cantidad de veces superior a lo esperado bajo condiciones de independencia. El antecedente y el consecuente tiene una correlación positiva.
  Un valor de lift < 1 es indicativo de que en el conjunto aparece una cantidad de veces inferior a lo esperado bajo condiciones de independencia. Existe una correlación negativa entre el antecedente y el consecuente.

### 3.23.2 APLICACIÓN DE ALGORITMO DE AGRUPAMIENTO K-MEANS

Vamos a describir un proceso de un algoritmo de agrupamiento k-means para ver cómo podemos realizar un agrupamiento de nuestros datos.

Lo primero que tenemos que tener en cuenta es que vamos a trabajar con un tipo de Aprendizaje No Supervisado, por lo que solo necesitaremos los datos de entrada, es decir nuestra variable ‘X_bal’, y será necesario trabajar con los datos estandarizados. De manera que es necesario tener los datos en una

misma escala, porque la técnica k-means es un tipo de algoritmo que trabaja con distancias, además de que vamos a utilizar el análisis de componentes principales PCA, (del que hemos hablado anteriormente en el apartado de la extracción de características) que trabaja principalmente con la varianza, por lo que nuestras características deben tener la misma importancia para dicha medición, en este apartado partimos con la ventaja de que ya hemos estandarizado los datos anteriormente y los hemos almacenado en la variable 'X_estandar'.

Procedemos a realizar el análisis de componentes principales. Este análisis lo realizamos para poder representar todo el conjunto de datos en un mismo diagrama de dispersión bidimensinal, donde reduciremos la dimensionalidad de nuestro conjunto de datos a 2 solamente, donde en un eje representaremos PC1, que será el análisis con mayor varianza y en el otro eje PC2, que será el análisis con la segunda varianza más alta para tener la mayoría de nuestros datos representados. Empezamos importando el módulo PCA del paquete decomposition de sklearn.

```
#Importar Librerias
from sklearn.decomposition import PCA

#Creamos una instancia de PCA: pca
pca = PCA(n_components=20)
X_pca= pca.fit_transform(X_estandar)

#Graficamos PCA con sus varianzas
features = range(pca.n_components_)
plt.bar(features, pca.explained_variance_ratio_, color='black')
plt.xlabel('PCA')
plt.ylabel('Varianza %')
plt.xticks(features)
```

Resultado una vez ejecutado el código:

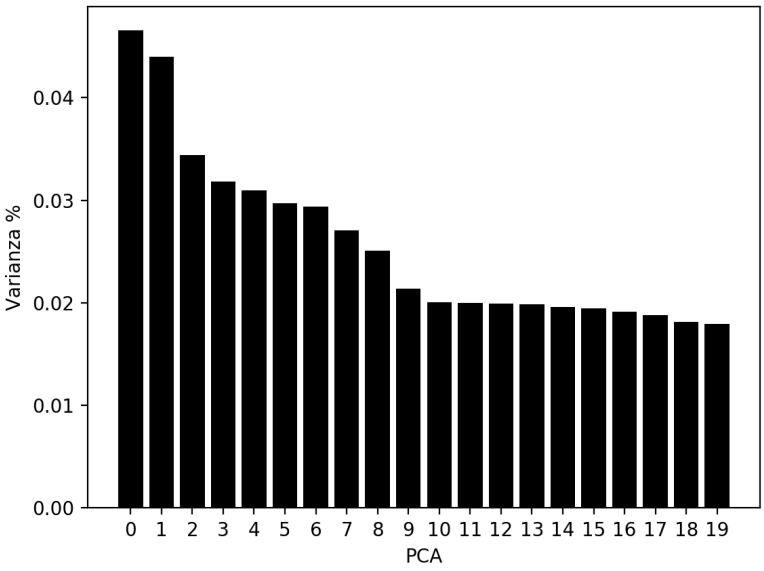



Ilustración 13: Gráfico de barras PCA.

Vamos a proceder a seleccionar nuestros dos componentes principales. Como vemos en la gráfica son los que explican la mayor parte de la varianza de nuestros datos.

**Método del codo**

Antes de aplicar el método, importaremos la librería KMeans del paquete cluster de sklearn y reduciremos el número de componentes principales a dos. Es un método que sirve para cuantificar el número ideal de grupos a crear en nuestro modelo. Utiliza dos parámetros, la inercia, que se puede definir como la suma de errores cuadrados dentro de un mismo grupo, y el número de clústeres a crear. Se puede decir que si el número de clústeres aumenta, la inercia disminuye, esto es así porque las muestras se encuentran más cerca de los centroides a los que han sido asignados. La idea principal de este método consiste en identificar el número de clústeres donde la inercia empieza a aumentar más rápidamente.

```
#Importar Librerias
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans

#Creamos una nueva instancia de PCA: pca
pca = PCA(n_components=2)
X_pca= pca.fit_transform(X_estandar)

ks = range (1, 10)
inercias = []
for k in ks:
    # Cremos una instancia de KMeans con k clústeres: modelo
    modelo = KMeans (n_clusters = k)

    # Ajustamos el modelo a las muestras
    modelo.fit (X_pca)

    # Agregamos la inercia a la lista de inercias
    inercias.append (modelo.inertia_)

plt.plot (ks, inercias, '-o', color = 'black')
plt.xlabel ('Número de grupos, k')
plt.ylabel ('Inercia')
plt.xticks (ks)
plt.show ()
```

Resultado una vez ejecutado el código anterior:

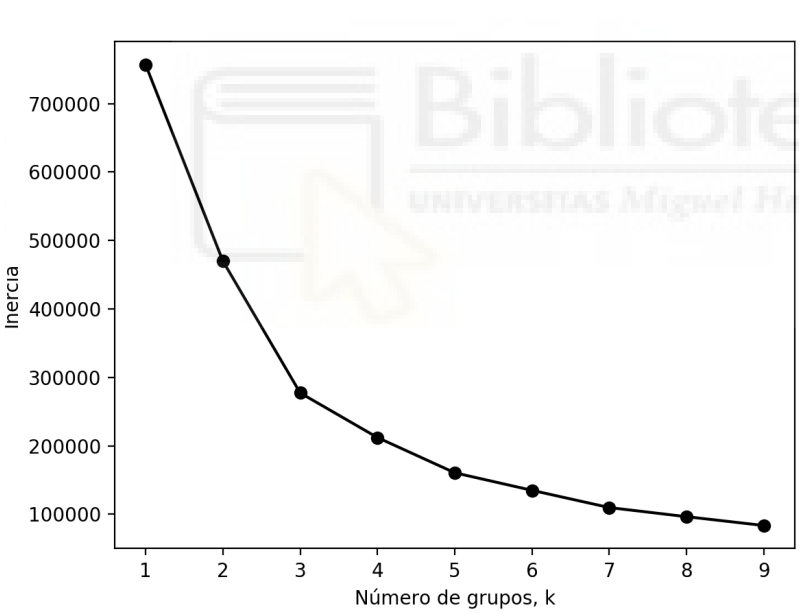



Ilustración 14: Gráfico del método del codo.

Vemos en la gráfica que se produce un punto de inflexión cuando el número de grupos es igual a 3, donde empieza a producirse un crecimiento brusco de la inercia, porque determinamos que ese es el número óptimo de grupos a crear.

Eliminamos el modelo anterior que hemos utilizado para calcular el número idóneo de clústeres y creamos un nuevo modelo k-means que llamaremos km,

pasándole como parámetro el número de grupos que hemos obtenido con el método del codo.

El algoritmo k-means utiliza por defecto el parámetro init=random, donde se forma una disposición aleatoria para colocar los centroides, pero a veces produce malos agrupamientos, por lo que hemos decidido cambiar el parámetro init=k-means++, que realiza múltiples ejecuciones del algoritmo k-means eligiendo el modelo que mejor rendimiento obtenga en términos de suma de errores cuadráticos (inercia). Configuramos también los parámetros n_init=10, max_inter=300, tol=1e-04 y random_state=0.  Nos disponemos a trazar cada clúster en el subespacio de características bidimensional, crearemos cada grupo con un color y una forma diferente para distinguirlos bien.

```
km = KMeans(n_clusters=3,
            init='k-means++',
            n_init=10,
            max_iter=300,
            tol=1e-04,
            random_state=0)
y_km = km.fit_predict(X_pca)

plt.scatter(X_pca[y_km == 0, 0], X_pca[y_km == 0, 1],  s=50, c='lightgreen', marker='s', edgecolor='black', label='cluster 1')
plt.scatter(X_pca[y_km == 1, 0], X_pca[y_km == 1, 1],  s=50, c='orange',  marker='o', edgecolor='black', label='cluster 2')
plt.scatter(X_pca[y_km == 2, 0], X_pca[y_km == 2, 1], s=50, c='lightblue', marker='v', edgecolor='black', label='cluster 3')
plt.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:,1], s=250, marker='*', c='red', label='centroids')
plt.legend(loc='best')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.show()
```

Resultado una vez ejecutado el código anterior:

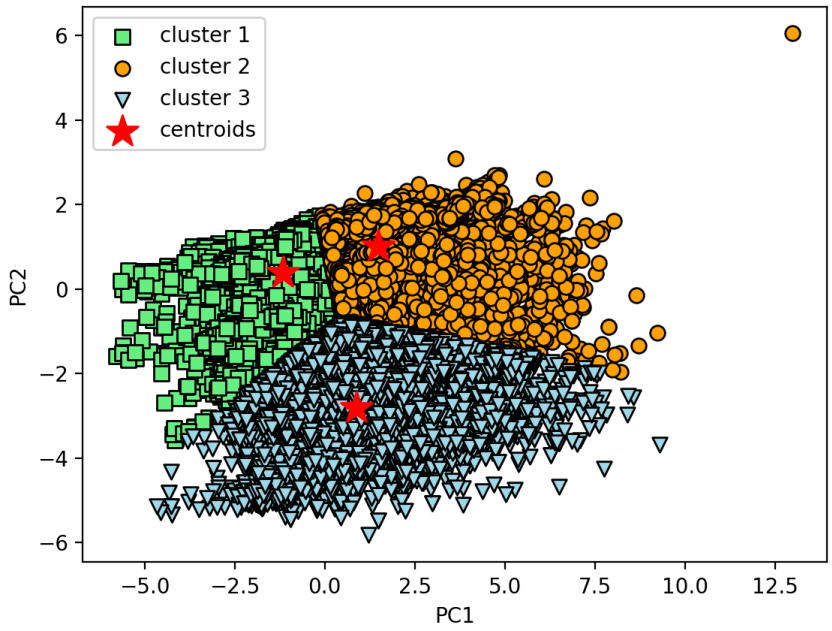



Ilustración 15: Gráfico de agrupación PCA.

Se puede ver los grupos claramente diferenciados con sus centroides bien distribuidos, por lo que podemos esclarecer que se produce una buena distribución en los datos de entrada y se pueden agrupar los datos en tres grupos distintos.

### 3.23.3. APLICACIÓN DE ALGORITMOS DE ASOCIACIÓN

Hasta ahora todos los algoritmos que hemos implementado de Aprendizaje Automático en Python los hemos encontrado en la librería de scikit-learn, pero para utilizar las reglas de asociación esto no es posible y necesitamos añadir una nueva librería llamada mlxtend, creado por Sebastian Raschka donde encontraremos el algoritmo apriori que nos sirve para obtener las reglas de asociación. Para poder utilizar el algoritmo solo necesitamos instalar la librería empleando pip:

```
In [9]: pip install mlxtend
```

Una vez instalado ya podemos utilizar el algoritmo apriori para obtener nuestras reglas de asociación en nuestra base de datos, recordar que al ser un algoritmo de Aprendizaje No Supervisado trabajaremos con los datos sin etiquetar por lo que utilizaremos la variable 'X_bal'.

Lo primero que haremos será convertir la variable 'X_bal' en un DataFrame, ya que necesitaremos tener el nombre de nuestras características para saber a cuál nos estamos refiriendo en cada momento. Una vez convertido nuestro array en un DataFrame, el programa nos asigna valores de 0 en adelante para llamar a nuestras columnas, por lo que es necesario asignar a cada número el nombre que le corresponde, esto lo hacemos rápidamente con el atributo columns de panda y procedemos a agregar los nombres en el mismo orden que están en nuestro DataFrame, de lo contrario asignaremos un nombre con una característica que no es produciendo un error.

```
X_asociacion = pd.DataFrame(X_bal)
X_asociacion.columns=['Number_of_Vehicles','Speed_limit','Year','Day_of_Week_Monday','Day_of_Week_Saturday','Day_of_Week_Sunday','Day_of_Week_Thursday','Day_of_Week_Tuesday',
                      'Day_of_Week_Wednesday','1st_Road_Class_A(M)','1st_Road_Class_B','1st_Road_Class_C','1st_Road_Class_Motorway','1st_Road_Class_Unclassifield',
                      'Road_Type_One way street','Road_Type_Roundabout','Road_Type_Single carriageway','Road_Type_Slip road','Road_Type_Unknown',
                      'Pedestrian_Crossing-Human_Control_Control by school crossing patrol','Pedestrian_Crossing-Human_Control_None within 50 metres',
                      'Pedestrian_Crossing-Physical_Facilities_Footbridge or subway','Pedestrian_Crossing-Physical_Facilities_No physical crossing within 50 meters',
                      'Pedestrian_Crossing-Physical_Facilities_Pedestrian phase at traffic signal junction','Pedestrian_Crossing-Physical_Facilities_Zebra crossing',
                      'Pedestrian_Crossing-Physical_Facilities_non-junction pedestrian crossing','Light_Conditions_Darkness: Street lighting unknown',
                      'Light_Conditions_Darkness: Street lights present and lit','Light_Conditions_Darkness: Street lights present but unlit',
                      'Light_Conditions_Daylight: Street light present','Weather_Conditions_Fine without high winds','Weather_Conditions_Fog or mist','Weather_Conditions_Other',
                      'Weather_Conditions_Raining with high winds','Weather_Conditions_Raining without high winds','Weather_Conditions_Snowing with high winds',
                      'Weather_Conditions_Snowing without high winds','Weather_Conditions_Unknown','Road_Surface_Conditions_Flood (Over 3cm of water)','Road_Surface_Conditions_Frost/Ice',
                      'Road_Surface_Conditions_Snow','Road_Surface_Conditions_Wet/Damp','Special_Conditions_at_Site_Auto traffic singal out','Special_Conditions_at_Site_Mud',
                      'Special_Conditions_at_Site_None','Special_Conditions_at_Site_Oil or diesel','Special_Conditions_at_Site_Permanent sign or marking defective or obscured',
                      'Special_Conditions_at_Site_Road surface defective','Special_Conditions_at_Site_Roadworks','Carriageway_Hazards_Dislodged vehicle load in carriageway',
                      'Carriageway_Hazards_Involvement with previous accident','Carriageway_Hazards_None','Carriageway_Hazards_Other object in carriageway',
                      'Carriageway_Hazards_Pedestrian in carriageway (not injured)','Urban_or_Rural_Area_Urban','Day','Month','Hour']
```

Para estar seguros de que lo hemos realizado correctamente, guardaremos el DataFrame en el escritorio y lo abriremos para comprobar que cada columna corresponde con sus valores. Crearemos una variable con la dirección donde queremos almacenar el nuevo DataFrame y la pasamos como argumento a la función to_csv() de pandas.

```
#Guardamos el nuevo DataFrame
X_asociacion_guardado='/Users/manuelgarciaalfaro/Desktop/X_asociacion.csv'
X_asociacion.to_csv(X_asociacion_guardado)
```

Resultado una vez ejecutado el código anterior:

| | Number_of_Vehicles | Speed_limit | Year | Day_of_Week_Monday | Day_of_Week_Saturday | Day_of_Week_Sunday | ... | Urban_or_Rural_Area_Urban | Day | Month | Hour |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 30 | 2012 | 0 | 0 | 0 | | 1 | 13 | 1 | 18 |
| 1 | 2 | 30 | 2012 | 0 | 0 | 0 | | 1 | 6 | 1 | 12 |
| 2 | 1 | 30 | 2012 | 0 | 0 | 0 | | 1 | 13 | 1 | 12 |
| 3 | 2 | 30 | 2012 | 0 | 1 | 0 | | 1 | 11 | 2 | 19 |

Efectivamente se ha creado el nuevo DataFrame correctamente, por lo que vamos a utilizar la función apriori para obtener las características con un mínimo de soporte.

```
from mlxtend.frequent_patterns import apriori
frequent_itemsets = apriori(X_asociacion > 0, min_support=0.06, use_colnames=True)
print(frequent_itemsets.head())
```

Resultado una vez ejecutado el código:

```
    support              itemsets
0  1.000000    (Number_of_Vehicles)
1  1.000000          (Speed_limit)
2  1.000000                (Year)
3  0.140175    (Day_of_Week_Monday)
4  0.136831  (Day_of_Week_Saturday)
```

A continuación, obtenemos las reglas que superen un mínimo de confianza.

```
from mlxtend.frequent_patterns import association_rules
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.8)
print(rules.head())
```

Resultado una vez ejecutado el código anterior:

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction |
|---|---|---|---|---|---|---|---|---|---|
| 0 | (Speed_limit) | (Number_of_Vehicles) | 1.000000 | 1.0 | 1.000000 | 1.0 | 1.0 | 0.0 | inf |
| 1 | (Number_of_Vehicles) | (Speed_limit) | 1.000000 | 1.0 | 1.000000 | 1.0 | 1.0 | 0.0 | inf |
| 2 | (Year) | (Number_of_Vehicles) | 1.000000 | 1.0 | 1.000000 | 1.0 | 1.0 | 0.0 | inf |
| 3 | (Number_of_Vehicles) | (Year) | 1.000000 | 1.0 | 1.000000 | 1.0 | 1.0 | 0.0 | inf |
| 4 | (Day_of_Week_Monday) | (Number_of_Vehicles) | 0.140175 | 1.0 | 0.140175 | 1.0 | 1.0 | 0.0 | inf |

De esta forma podemos ir ajustando los valores mínimos y máximos de soporte, confianza y mejora de confianza, para así poder buscar las relaciones que nos interesen dentro de nuestra base de datos.

Podemos realizar una interpretación rápida (a modo de ejemplo) fijándonos en la fila número 4, donde se puede ver que el soporte indica que es lunes un 14.01% de las veces de todas las instancias de nuestra base de datos y que como es lógico tiene una confianza de 1 con el número de vehículos. Todas las instancias que son lunes, tienen asociadas un número de vehículos debido a que no existen valores perdidos ni errores en el número de vehículos. La mejora de confianza igual a 1, es indicativo de que el lunes y el número de vehículos aparecen las mismas veces que se esperaría si los dos fueran independientes, por lo que podemos estimar que no tienen ninguna relación entre si.

# CAPITULO 4

# 4. PROBLEMA EN BIGML

## 4.1. CONTEXTO

Analizaremos ahora nuestro problema de otra forma distinta a la anterior para ver si podemos mejorar en nuestra predicción o no. Esta se realizará mediante el uso de una plataforma integral de Machine Learning, denominada BigML. Intentaremos tratar el problema eliminando menos variables que en el caso anterior e incluyendo la localización en el desarrollo de nuestro problema como una variable con más relevancia, para ver cómo esta afecta al grado de severidad de un accidente.

Muchos de los datos preprocesados en el problema anterior en Python nos servirán ahora en nuestro nuevo problema, pero con algunas modificaciones:

```
#Eliminacion de columnas irrelevantes
#Localizacion
data=data.drop(['Location_Easting_OSGR','Location_Northing_OSGR','LSOA_of_Accident_Location'], axis=1)
#Numeración de carreteras
data=data.drop(['1st_Road_Number','2nd_Road_Number'], axis=1)
#Atendió el escenario
data=data.drop(['Police_Force','Did_Police_Officer_Attend_Scene_of_Accident'], axis=1)
#Autoridad local donde se produjo el accidente
data=data.drop(['Local_Authority_(District)','Local_Authority_(Highway)'], axis=1)
#Indece del accidente
data=data.drop(['Accident_Index'], axis=1)
#Número de víctimas
data=data.drop(['Number_of_Casualties'], axis=1)
```

En el problema anterior eliminábamos todas las columnas relacionadas con la localización, pero ahora queremos contar con las de latitude y longitude por lo que las eliminaremos de la función drop, para que no se eliminen de nuestra base de datos.

También anularemos los cambios en los datos categóricos, debido a que BigML tiene su propio sistema para tratar estos datos, por lo que no será necesario hacer ninguna conversión como hacíamos antes para convertir las variables categóricas. Vamos a dejar la conversión de variables numéricas a categóricas, pero anularemos la conversión de todas las variables categóricas a variables tipo dummies.

```
#Reemplazar los valores clase 1st_Road_Class, Day_of_Week, Urban_or_Rural_Area
data['1st_Road_Class'].replace([1,2,3,4,5,6],['Motorway','A(M)','A','B','C','Unclassifield'], inplace=True)
print(data['1st_Road_Class'])

data['Day_of_Week'].replace([1,2,3,4,5,6,7],['Sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday'], inplace=True)
print(data['Day_of_Week'])

data['Urban_or_Rural_Area'].replace([1,2],['Urban','Rural'], inplace=True)
print(data['Urban_or_Rural_Area'])

print('Conversión de variables numéricas a categóricas')
data.head()

'''
#Conversión datos categóricos
data=pd.get_dummies(data, columns = ['Day_of_Week','1st_Road_Class','Road_Type','Pedestrian_Crossing-Human_Control',
'Pedestrian_Crossing-Physical_Facilities','Light_Conditions','Weather_Conditions','Road_Surface_Conditions',
'Special_Conditions_at_Site','Carriageway_Hazards','Urban_or_Rural_Area'], drop_first = True)

print(data.head())
print(data.dtypes)
'''
```

Los demás datos preprocesados anteriormente en Python, los dejaremos igual para nuestro segundo problema, esto nos ahorrará bastante tiempo, más que si decidiéramos hacer todo el preprocesamiento desde BigMl de nuevo.

Ahora procederemos a guardar nuestros cambios en un nuevo archivo csv al que llamaremos ‘Accidents’. Para hacer esto y conseguir que nuestros datos estén balanceados necesitaremos realizar las siguientes operaciones en Python:

Justo a continuación de donde hemos realizado el submuestreo, procederemos primeramente a cambiar nuestros tipos de datos array ‘X_bal’ a un DataFrame que llamaremos ‘X’, al realizar esta operación las columnas tomarán valores por defecto del 0 al 25, por lo que tendremos que cambiar los nombres conforme los teníamos antes. (lo que hacemos con X.columns=[…]).

Después crearemos una columna para introducir los índices que se generen automáticamente al utilizar la función DataFrame, esto lo haremos con la función reset_index (level=0, inplace=True), que nos servirá para crear una columna en común con los datos del DataFrame, que crearemos a partir del array ‘y_bal’ al que denominaremos ‘y’. Para su posterior unión por medio de la función merge(), hay que pasarle un parámetro denominado on y el nombre de

la columna, que hace de solapamiento entre los dos DataFrame ‘X’ e ‘y’. Esto generara un nuevo DataFrame único que denominaremos ‘Accidents’.

Una vez creado ‘Accidents’ nos damos cuenta de que la columna que nos ha servido para la unión es totalmente redundante, por lo que la eliminaremos por medio de la función drop(), también eliminaremos el índice autogenerado por la función DataFrame, esto se hace con la función set_index() pasándole como parámetro, el valor de la primera columna que quieres que aparezca como columna inicial y sirva para sustituir al índice.

Por último, procedemos a guardar los DataFrame creados como archivos csv, creamos una variable con la ruta donde queremos almacenar el archivo y posteriormente pasamos esta variable como argumento a la función to_csv().

```
X=pd.DataFrame(X_bal)
X.columns = ['Longitude','Latitude','Number_of_Vehicles','Day_of_Week','1st_Road_Class','Road_Type','Speed_limit','Pedestrian_Crossing-Human_Control',
             'Pedestrian_Crossing-Physical_Facilities','Light_Conditions','Weather_Conditions','Road_Surface_Conditions','Special_Conditions_at_Site','Carriageway_Hazards',
             'Urban_or_Rural_Area','Year','Day','Month','Hour']
X.reset_index(level=0,inplace=True)


y=pd.DataFrame(y_bal)
y.columns =['Accident_Severity']
y.reset_index(level=0,inplace=True)

#Unimos los DataFrame de X e y en uno solo que llamaremos Accidents
Accidents=pd.merge(X, y, on='index')

#Redundante índece
Accidents=Accidents.drop(['index'], axis=1)
Accidents.set_index('Longitude', inplace=True)

#Guardamos los nuevos DataFrame
X_guardar='/Users/manuelgarciaalfaro/Desktop/TFG/X.csv'
X.to_csv(X_guardar)

y_guardar='/Users/manuelgarciaalfaro/Desktop/TFG/y.csv'
y.to_csv(y_guardar)

Accidents_guardar='/Users/manuelgarciaalfaro/Desktop/TFG/Accidents.csv'
Accidents.to_csv(Accidents_guardar)
```

Resultado una vez ejecutado el código anterior:


Ilustración 16: Localización de ficheros creados.

-X.csv

| | index | Longitude | Latitude | Number_of_Vehicles | Day_of_Week | 1st_Road_Class | Road_Type | Speed_limit |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | -0.200249 | 51.492652 | 1 | Friday | A | Dual carriageway | 30 |
| **1** | 1 | -0.188775 | 51.502278000000004 | 2 | Friday | A | Single carriageway | 30 |
| **2** | 2 | -0.185916 | 51.501694 | 1 | Friday | Unclassifield | Single carriageway | 30 |
| **3** | 3 | -0.213002 | 51.50175 | 2 | Saturday | A | Single carriageway | 30 |

Ilustración 17: Fichero ‘X’.

-y.csv

y

| | index | Accident_Severity |
|---|---|---|
| **0** | 0 | 1 |
| **1** | 1 | 1 |
| **2** | 2 | 1 |
| **3** | 3 | 1 |

Ilustración 18: Fichero ‘y’.

-Accidents

| Longitude | Latitude | Number_of_Vehicles | Day_of_Week | | Month | Hour | Accident_Severity |
|---|---|---|---|---|---|---|---|
| **-0.200249** | 51.492652 | 1 | Friday | | 1 | 18 | 1 |
| **-0.188775** | 51.502278000000004 | 2 | Friday | | 1 | 12 | 1 |
| **-0.185916** | 51.501694 | 1 | Friday | | 1 | 12 | 1 |
| **-0.213002** | 51.50175 | 2 | Saturday | ... | 2 | 19 | 1 |

Ilustración 19: Fichero ‘Accidents’.

Observamos como el archivo tiene la forma deseada y tanto el DataFrame ‘X’ como el ‘y’ fueron unificados correctamente en el archivo ‘Accidents’, ya que incluye la característica Accident_Severity.

Comentar que podíamos simplificar el proceso anterior, utilizando la función column_stack() de la librería numpy, que nos permite la unión de arrays de diferente dimensionalidad, para después realizar su posterior conversión a DataFrame, esto sería algo similar a esto:

```
#Alternativa rápida con union de arrays
Accidents = np.column_stack((X_bal,y_bal))
Accidents = pd.DataFrame(Accidents)
Accidents.columns=['Longitude','Latitude','Number_of_Vehicles','Day_of_Week','1st_Road_Class','Road_Type','Speed_limit','Pedestrian_Crossing-Human_Control',
            'Pedestrian_Crossing-Physical_Facilities','Light_Conditions','Weather_Conditions','Road_Surface_Conditions','Special_Conditions_at_Site','Carriageway_Hazards',
            'Urban_or_Rural_Area','Year','Day','Month','Hour','Accident_Severity']
#Fijar primera columna
Accidents.set_index('Longitude', inplace=True)
#Guardamos los nuevos DataFrame
Accidents_guardar='/Users/manuelgarciaalfaro/Desktop/Accidents.csv'
Accidents.to_csv(Accidents_guardar)
```

Podemos emplear cualquiera de los dos métodos, uno trabaja con la unión de arrays y el otro con la unión de DataFrames.

## 4.2. KEPLER.GL

Es conveniente recordar que hemos realizado el proceso anterior antes del escalamiento, porque queremos obtener una visualización de nuestros datos mediante las coordenadas de localización y de haberlo aplicado hubiéramos alterado todos estos datos. Dicho esto, subiremos nuestros datos a la plataforma Kepler, para observar su comportamiento y si estos se encuentran sesgados.


Accidents Slight Accidents Fatal and Serious

Ilustración 20: Representación geográfica de los accidentes.

Hemos separado los casos en función de la gravedad, en la parte izquierda de la ilustración 20 se encuentran los casos leves ‘Accidents Slight’ = 0 (color naranja) y en la parte derecha de la ilustración los casos graves y mortales ‘Accidents Fatal and Serious’ = 1 (color rojo). Vemos que a simple vista no podemos sacar muchas conclusiones, ya que contamos con una gran cantidad de datos, por lo que será necesario utilizar filtros con los valores de las características que deseamos.

Vamos a empezar filtrando y haciendo animaciones de la concentración de accidentes en función del tiempo.

Primeramente, con la hora, marcando un rango de una hora.

- Hora 03:00am-04:00am.

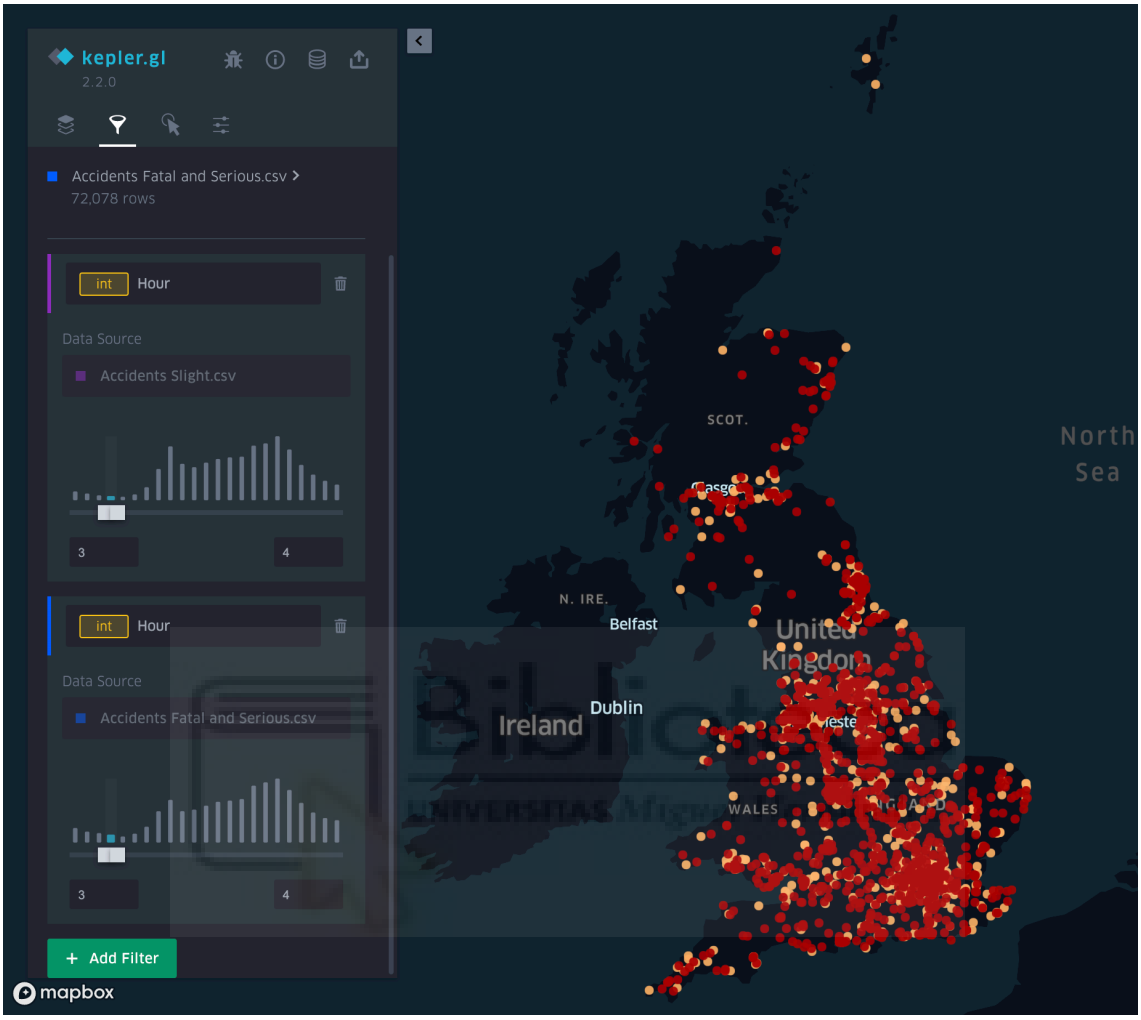



Ilustración 21: Representación geográfica horaria 03:00am-04:00am.

- Hora 17:00pm-18:00pm.

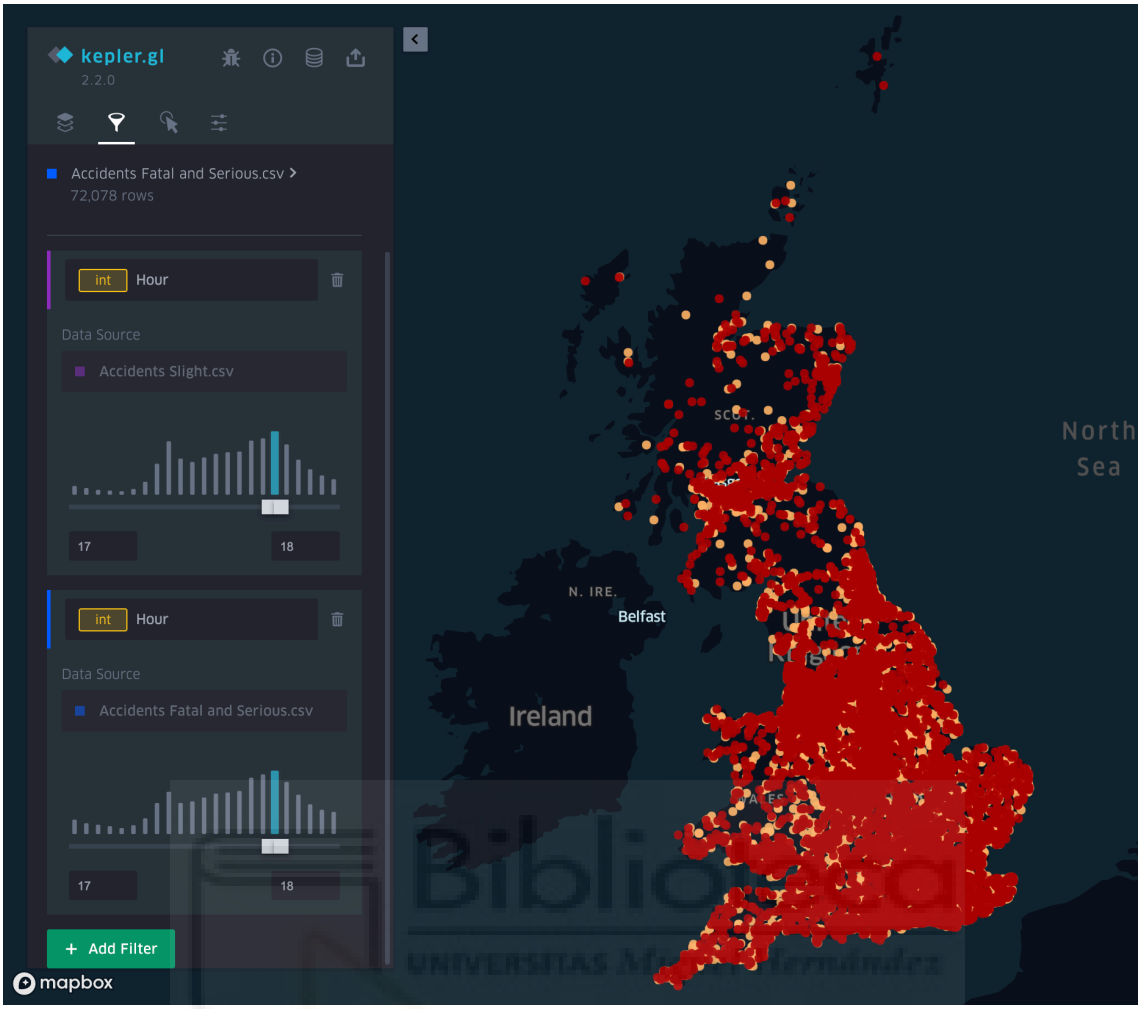



Ilustración 22: Representación geográfica horaria 17:00pm-18:00pm.

Como observamos, en las horas de madrugada 03:00am-04:00am apenas se produjeron accidentes, mientras que en el horario de 17:00pm-18:00pm el número de accidentes es mucho mayor, debido claro está, a la mayor actividad de las personas durante esa franja horaria, por lo que comprobamos que nuestros datos tienen cierta lógica y no se encuentran sesgados.

Otra de las cosas que nos damos cuenta a la hora de filtrar por hora, es que independientemente de la hora filtrada, se observa claramente una mayor concentración de accidentes en ciudades con una gran área metropolitana como son Londres, Manchester, Birmingham o Newcastle, que en zonas rurales, que apenas cuentan con afluencia de tráfico en sus carreteras.

Podemos probar a filtrar más características, por ejemplo, ver la concentración del número de accidentes dependiendo del día de la semana.

- Lunes

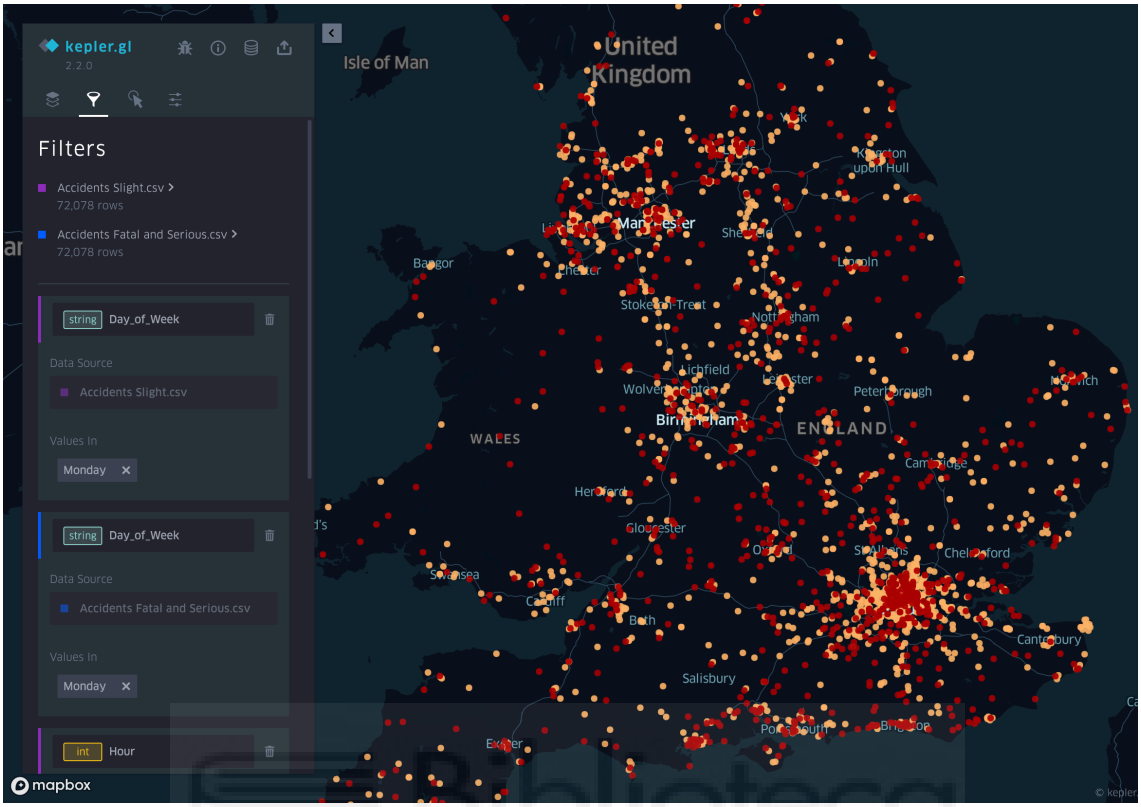



Ilustración 23: Representación geográfica de los Lunes.

- Domingo

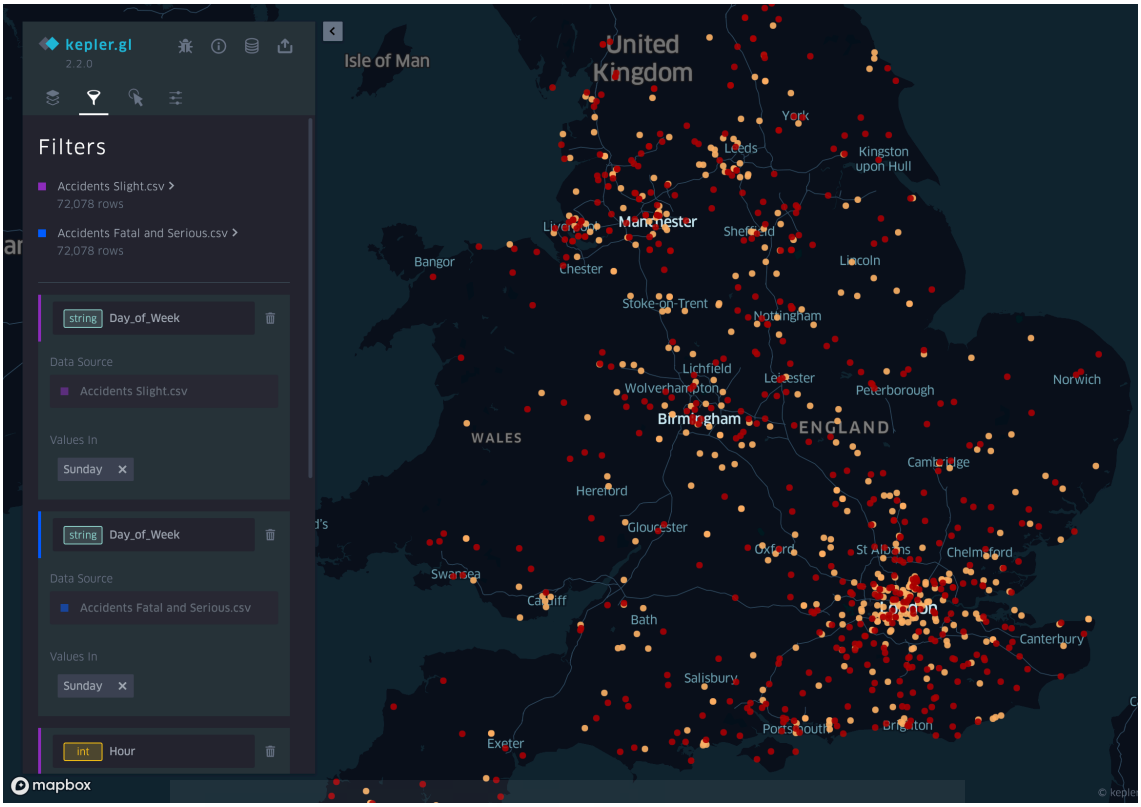


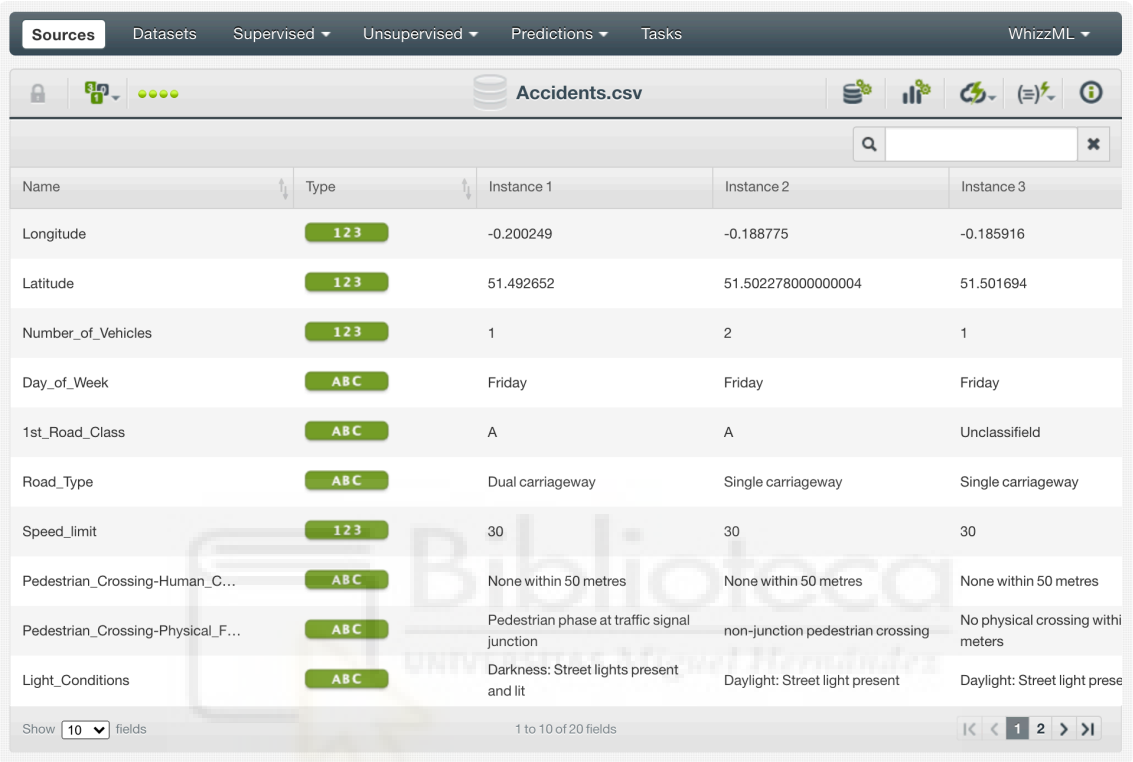

Ilustración 24: Representación geográfica de los Domingos.

Se observa claramente que se producen menos accidentes un Domingo, es decir, un día festivo que por ejemplo un día laboral como podría ser un lunes, donde vemos que los casos producidos son más. Hay que decir que hemos aplicado un filtro horario de 8 a 9 de la mañana, porque la actividad laboral propicia una mayor movilidad de vehículos, provocando más casos de accidentes.

## 4.3. PRIMEROS PASOS EN BIGML

Subiremos nuestro archivo Accidents.csv inicial a la plataforma BigML. Para subir nuestro archivo basta con acceder a la plataforma y arrastrar el fichero a la pantalla de listado de recursos. Una vez cargados los datos veremos que se ha creado un objeto nuevo en la interfaz con el nombre del fichero subido. Llamaremos Source a este tipo de objetos que almacenan las características necesarias para interpretar:

- Los campos que contiene nuestro fichero de datos: La primera fila del fichero es analizada para determinar si contiene los nombres de los campos. De no ser así, se asignan nombres automáticamente.
- El tipo de cada campo: Al lado de cada nombre vemos una imagen que nos indica el tipo de valores que contiene ese campo.




| Name | Type | Instance 1 | Instance 2 | Instance 3 |
|---|---|---|---|---|
| Longitude | 123 | -0.200249 | -0.188775 | -0.185916 |
| Latitude | 123 | 51.492652 | 51.502278000000004 | 51.501694 |
| Number_of_Vehicles | 123 | 1 | 2 | 1 |
| Day_of_Week | ABC | Friday | Friday | Friday |
| 1st_Road_Class | ABC | A | A | Unclassifield |
| Road_Type | ABC | Dual carriageway | Single carriageway | Single carriageway |
| Speed_limit | 123 | 30 | 30 | 30 |
| Pedestrian_Crossing-Human_C… | ABC | None within 50 metres | None within 50 metres | None within 50 metres |
| Pedestrian_Crossing-Physical_F… | ABC | Pedestrian phase at traffic signal junction | non-junction pedestrian crossing | No physical crossing withi meters |
| Light_Conditions | ABC | Darkness: Street lights present and lit | Daylight: Street light present | Daylight: Street light prese |

Show 10 fields 1 to 10 of 20 fields 1 2

Ilustración 25: Vista preliminar de nuestra base de datos.

El tipo de dato que admite esta plataforma puede ser:

- Numérico: Valores que sólo contienen números.
- Categórico: Son un número limitado de valores de tipo texto que se repiten en varias instancias.
- Fecha/hora: Aquellos campos cuyo contenido encaja con alguno de los formatos de fecha/hora más usuales (año, mes, día, hora, minutos, segundos).
- Items: Un campo es del tipo items cuando contiene textos separados por algún carácter separador.
- Texto: Su contenido es de texto libre, es decir, que no está limitado a un subconjunto de etiquetas, o bien cuando el número de etiquetas distintas supera las 1.000.

Usando la opción 1-click dataset del menú de nuestro Source se crea un objeto dataset. Éste es siempre el paso previo al aprendizaje, en él se analiza el contenido de cada uno de los campos y se serializan los valores encontrados.

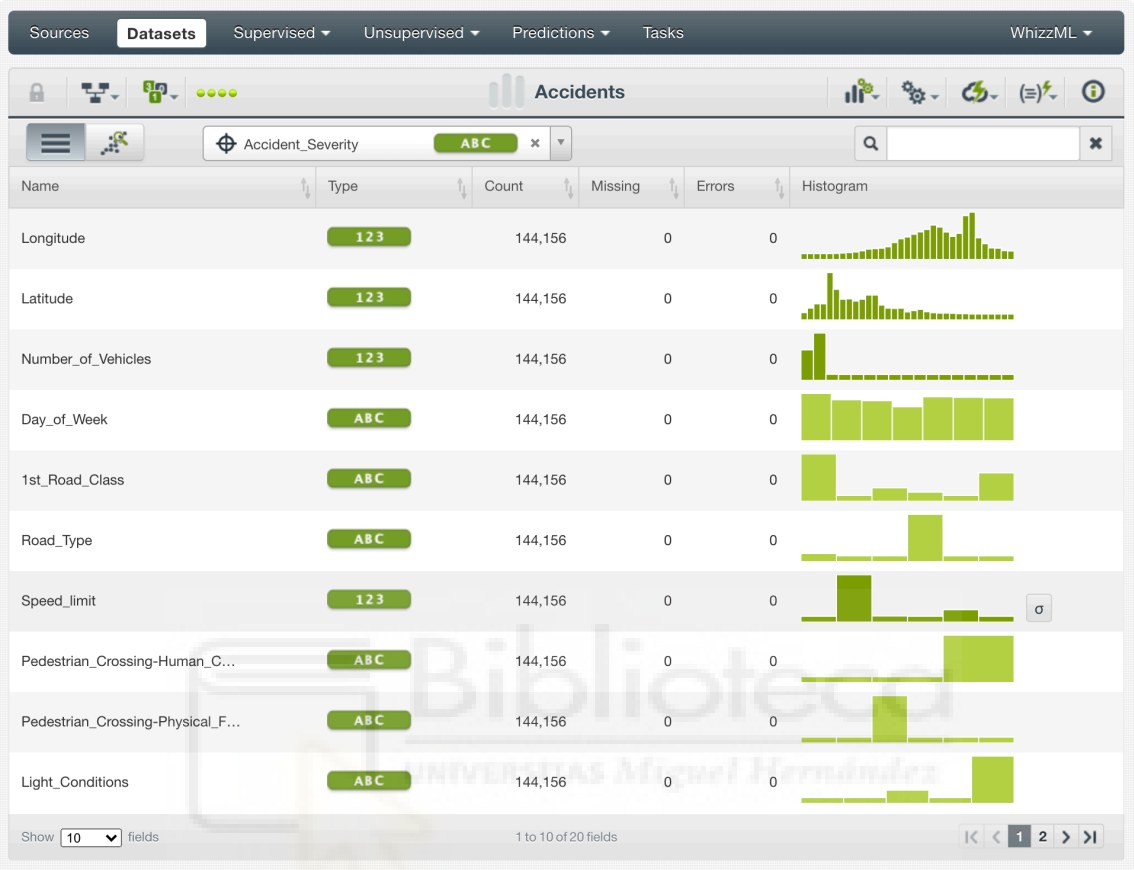



Ilustración 26: Análisis de los datos.

Observamos que aparece una visualización de cómo nuestras características se comportarán. Aparecen las siguientes columnas:

- Count: Hace referencia al número de instancias que hay en el atributo.
- Missing: Indica los valores perdidos en dicha característica.
- Errors: Detecta errores en las características. Por ejemplo, una característica tipo numérico donde encontramos valores de texto.
- Histogram: Muestra una distribución de los datos y se puede observar si sigue una distribución normal.

Otra de las cosas que se puede observar es la información estadística de cada atributo.

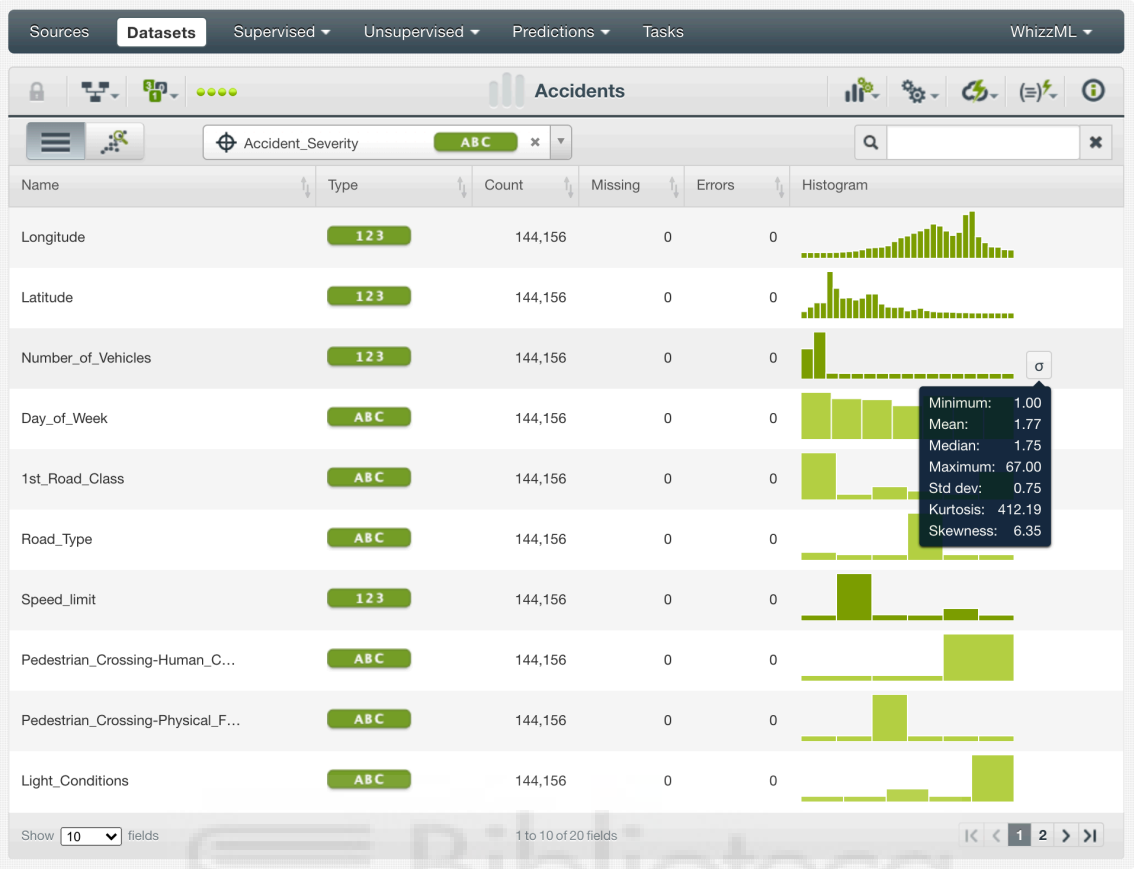



Ilustración 27: Vista estadística de los datos.

## 4.4. APRENDIZAJE SUPERVISADO EN BIGML

Vamos implementar el algoritmo de Aprendizaje Supervisado en el que mejor resultados hayamos obtenido en Python. Al ser pocas las variables que vamos a introducir en nuestra base de datos, no deberían producir un cambio muy significativo en los resultados de rendimiento obtenidos.

### 4.4.1. BOSQUES ALEATORIOS CLASIFICACIÓN

Vamos a proceder a crear un modelo de bosques aleatorios de clasificación (Ensemble), porque ha sido el modelo con mejores resultados hemos obtenido en el ejercicio anterior, así podremos darnos una idea de como BigML trata los

datos y si los resultados de predicción cambian al añadir las variables de localización.

Después dividiremos nuestro dataset en dos conjuntos dataset de entrenamiento y dataset de test, con la opción que nos muestra BigML.

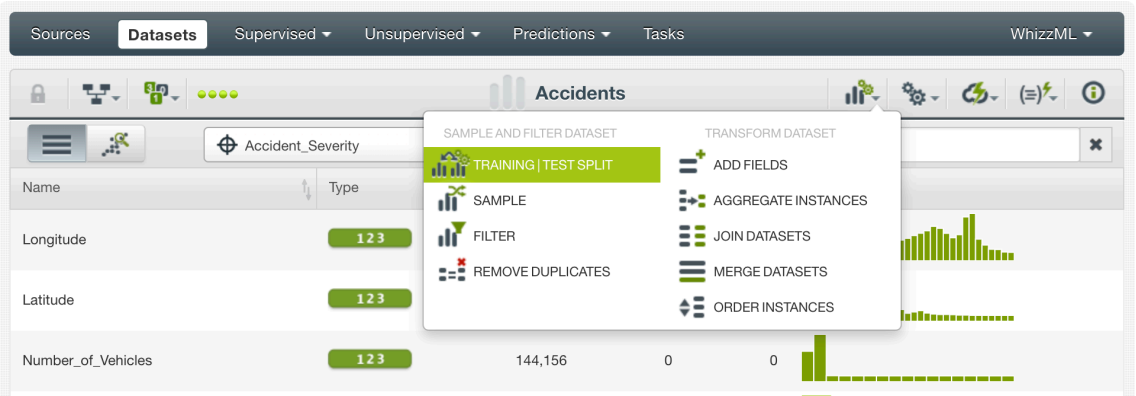



Ilustración 28: División de los datos en entrenamiento y test.

Procederemos a seleccionar el porcentaje de datos con los que vamos a realizar el entrenamiento de nuestro algoritmo y el porcentaje restante lo reservaremos para realizar la evaluación de dicho algoritmo con la matriz de confusión. Vamos a seleccionar el 75% de los datos totales para el entrenamiento y el 25% para la evaluación, lo haremos con los mismos porcentajes que en nuestro algoritmo en Python. Seleccionaremos estos porcentajes e intentaremos semejar las condiciones en las que hemos predicho los resultados con el algoritmo de bosques aleatorios clasificación en Python, para después comparar los resultados de una y otra matriz de confusión y así poder ver si existe una similitud entre los datos predichos al añadir la localización en BigML y los datos predichos sin localización en Python.

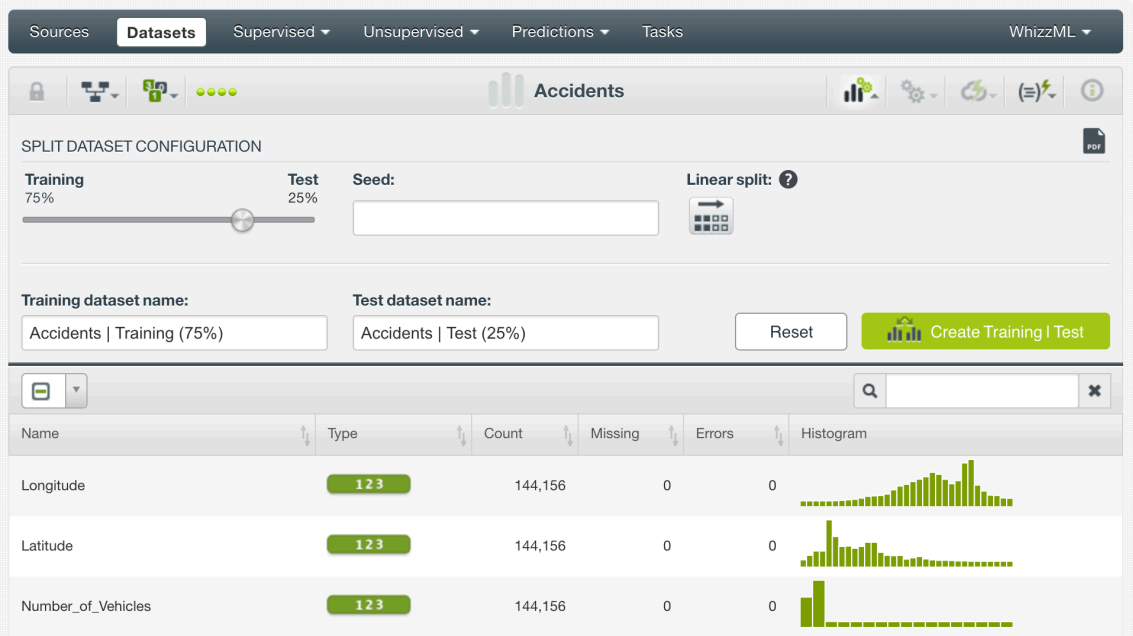



Ilustración 29: Selección porcentaje en la división de los datos.

Como vemos aparece una barra para seleccionar el porcentaje de datos que queremos seleccionar para entrenar nuestro modelo y cuál reservar para hacer el test, una vez seleccionado pulsamos la casilla Create Training|Test, y obtendremos nuestros dos dataset.

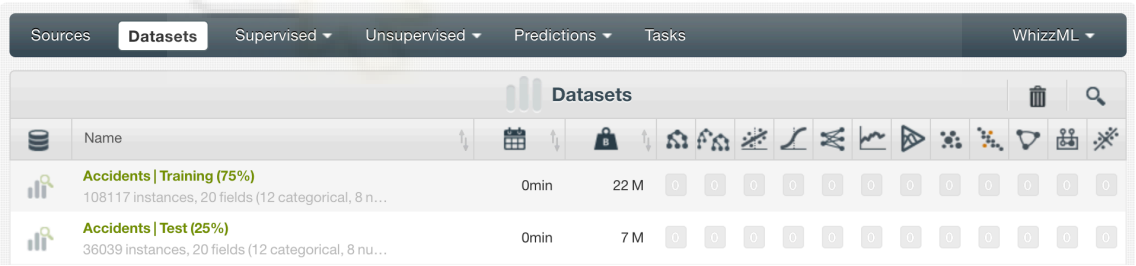



Ilustración 30: Vista de los datasets generados de entrenamiento y test.

Accedemos a nuestros datasets y vemos como efectivamente se ha creado correctamente, a continuación, clicamos en Accidents|Training(75%).

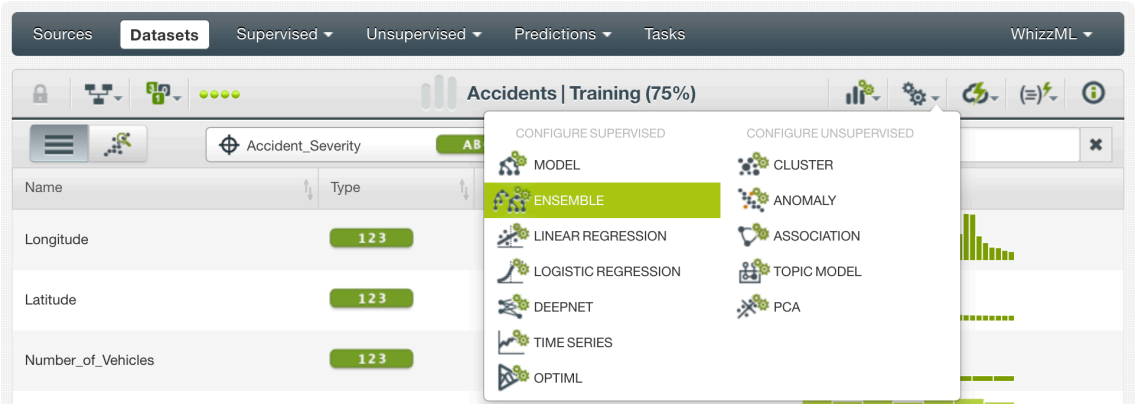



Ilustración 31: Cómo crear un bosque aleatorio clasificación.

En configure vemos como salen todos los modelos que podemos crear, tanto los modelos de Aprendizaje Supervisado como los de Aprendizaje No Supervisado, y marcamos el modelo que queremos crear, en nuestro caso ensemble (bosque aleatorio).

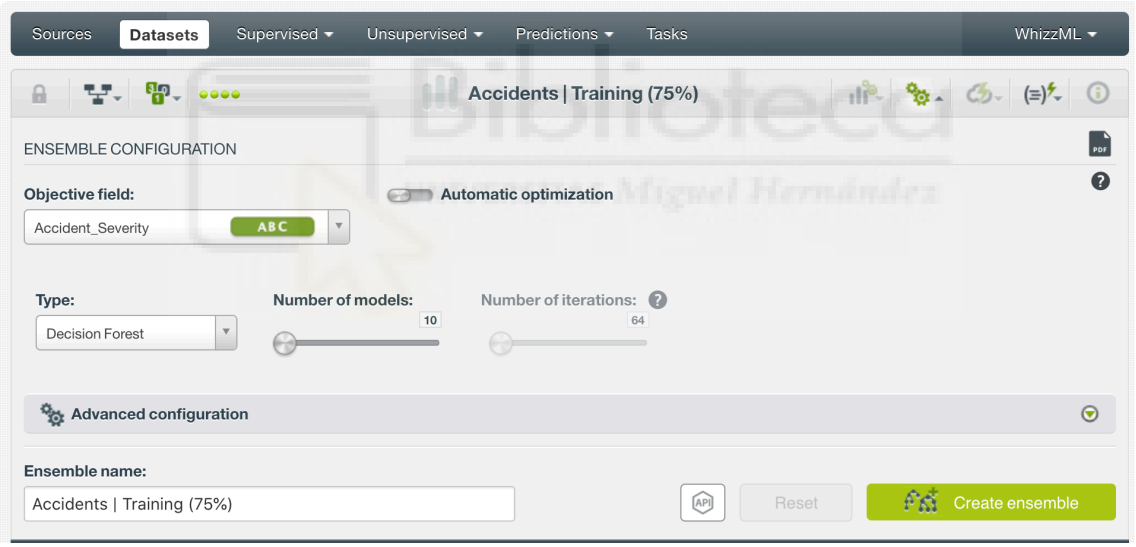



Ilustración 32: Ajustar parámetros en un bosque aleatorio clasificación.

Como observamos al seleccionar ensemble podemos ajustar algunos hiperparámetros como el tipo, el número de modelos, o el número de interacciones, lo dejaremos por defecto y marcamos la casilla Create ensemble.

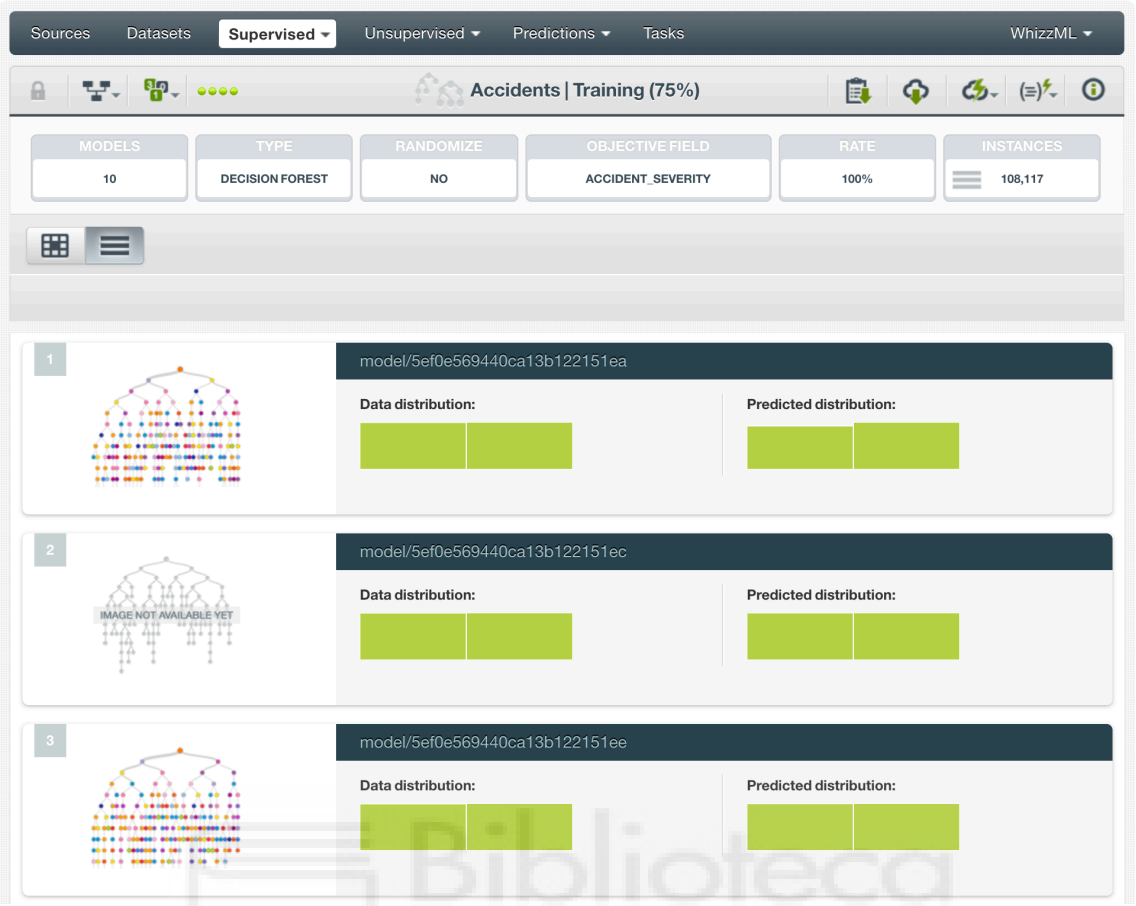



Ilustración 33: Vista general de bosque aleatorio clasificación.

Efectivamente vemos como se han generado 10 modelos, a los cuales podemos acceder y ver cómo se comporta cada uno, esta es una de las ventajas que nos ofrece BigML para ayudarnos a comprender mejor el comportamiento de nuestros datos.

Por ejemplo, si accedemos al modelo numero 1:

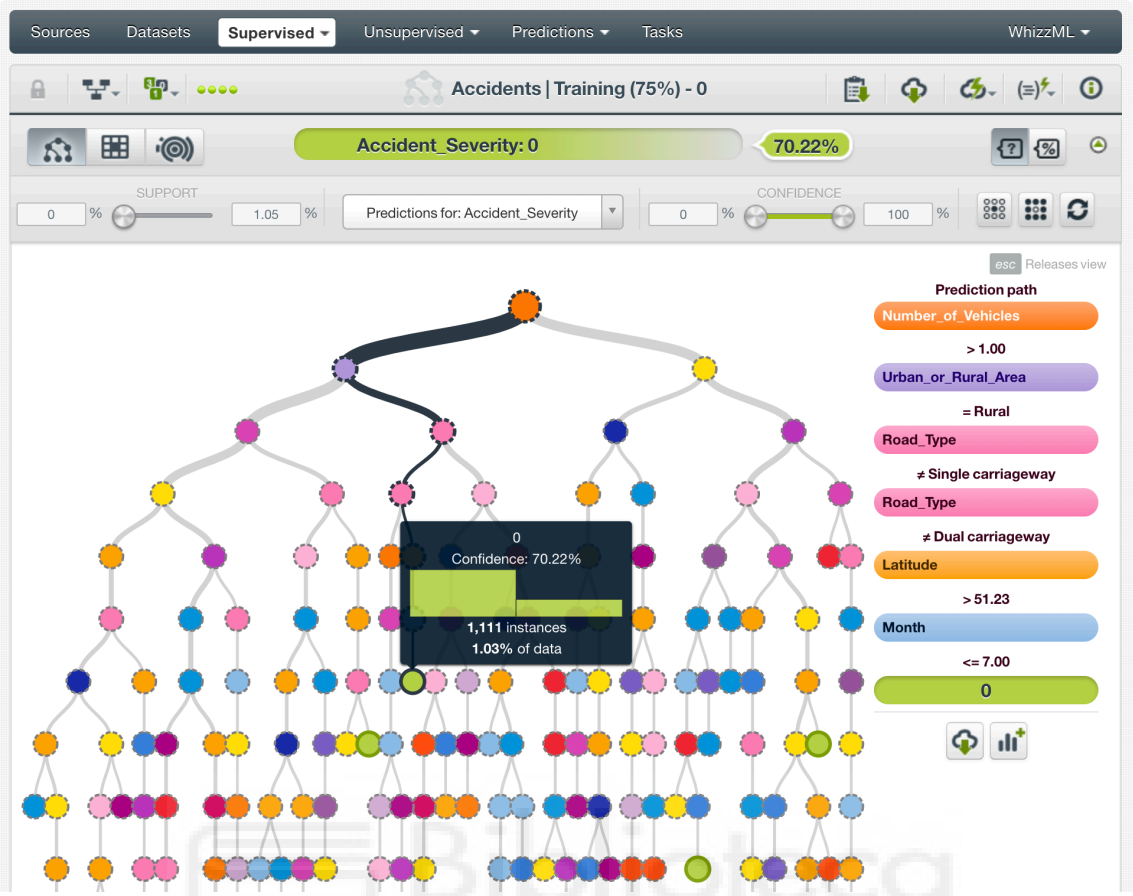



Ilustración 34: Árbol generado en el bosque aleatorio clasificación.

Podemos ir seleccionando los nodos y ver como se comporta nuestro modelo, qué atributos intervienen y qué valores adquieren para determinar la gravedad del accidente. Por ejemplo, el nodo que hemos seleccionado en la imagen anterior nos dice que el 70.22% de los casos han resultado adquirir el valor de 0, mientras que un 29.78% ha resultado ser 1 para las condiciones mostradas a la derecha de la imagen, es decir, en donde se ha producido un accidente en el que intervienen más de un vehículo, se produce en una zona urbana, donde el tipo de la vía es distinta a la características de un solo carril y dos carriles, para una latitud mayor de 51.23 y para todos los meses iguales o anteriores a Julio.

Para el modelo que hemos generado podemos observar la importancia de las variables de dicho modelo clicando sobre Ensemble Summary Report.

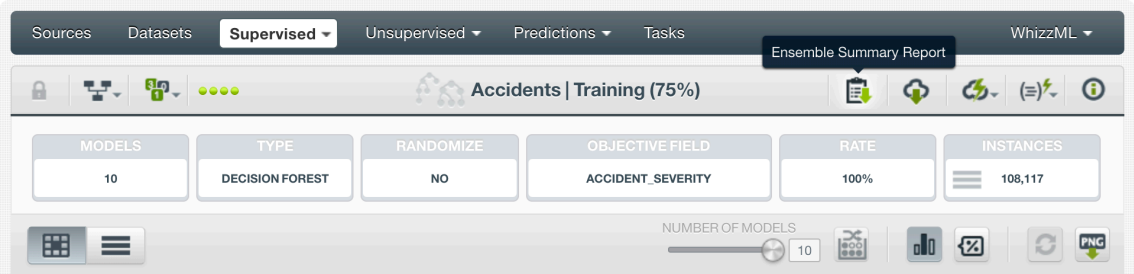



Ilustración 35: Cómo visualizar porcentajes de relevancia de las características en nuestro modelo.

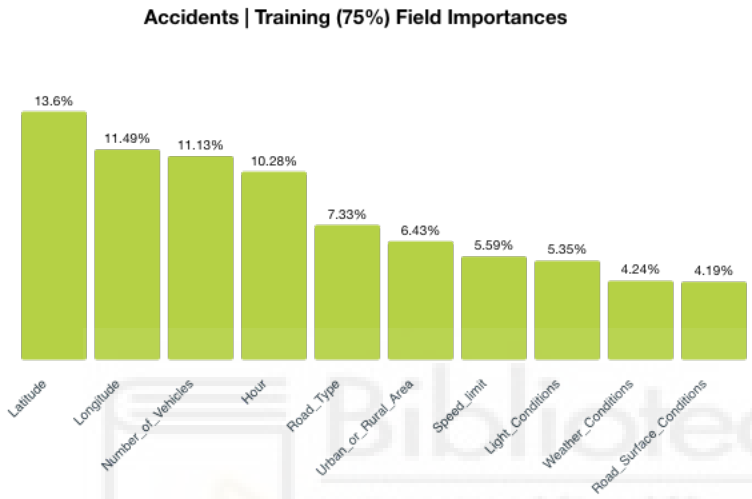



Ilustración 36: Visualización porcentajes de relevancia de las características en nuestro modelo.

Vemos que las características más importantes para nuestro modelo son las de latitud y longitud añadidas especialmente en este problema para compararlo con el problema anterior resuelto en Python. En la ilustración 36, podemos observar como se produce un gráfico de barras con el porcentaje de importancia de cada característica en el problema.

## 4.5. GRÁFICO DE INDEPENDENCIA PARCIAL (O PCP)

El gráfico de independencia parcial sirve para visualizar la importancia que nuestros campos tienen en las predicciones y como variarán éstas al variar los valores de dichos campos.

El gráfico es una representación de mapa de calor de sus predicciones en función de diferentes valores de los dos campos seleccionados en los ejes, sin tener en cuenta el resto de campos del dataset. Los ejes admiten tanto campos categóricos como numéricos, otra de las cosas que se pueden hacer es configurar los valores para el resto de los campos de entrada utilizando el panel del inspector de campos que aparece a la derecha.

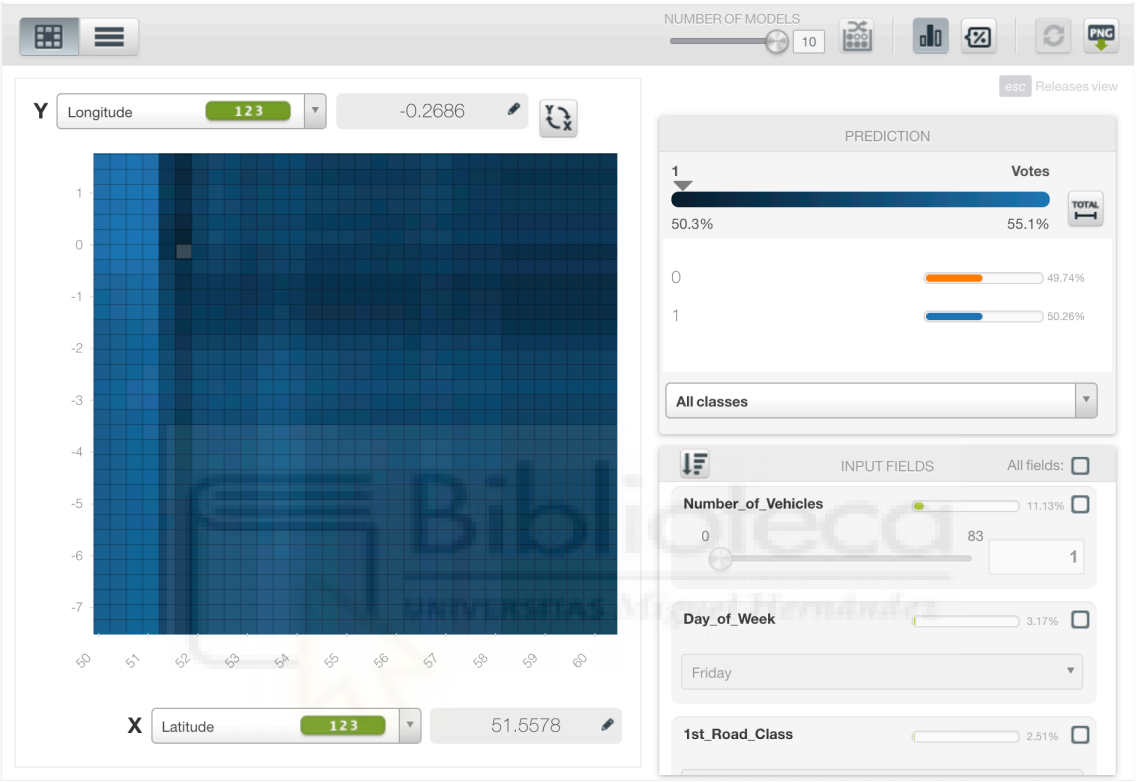



Ilustración 37: PCP de las características más importantes de nuestro modelo.

Si observamos por ejemplo los dos atributos más influyentes en nuestro modelo (latitude y longitude) y deshabilitamos los demás atributos de modo que no se tengan en cuenta para predecir el grado de gravedad del accidente, se aprecia un claro predominio de la clase 1(color azul), pero vemos que todos los valores se encuentran contenidos en el intervalo de 50.3% a 55.1% esto indica que la localización de un caso tiene una ligera tendencia a determinar que un caso es grave o mortal, pero al ser el margen de valores tan reducido y oscilar en el 50%, determinamos que la localización de un caso por si sola no puede ser determinante para clasificar la gravedad de dicho accidente ya que apenas se diferencia del azar.

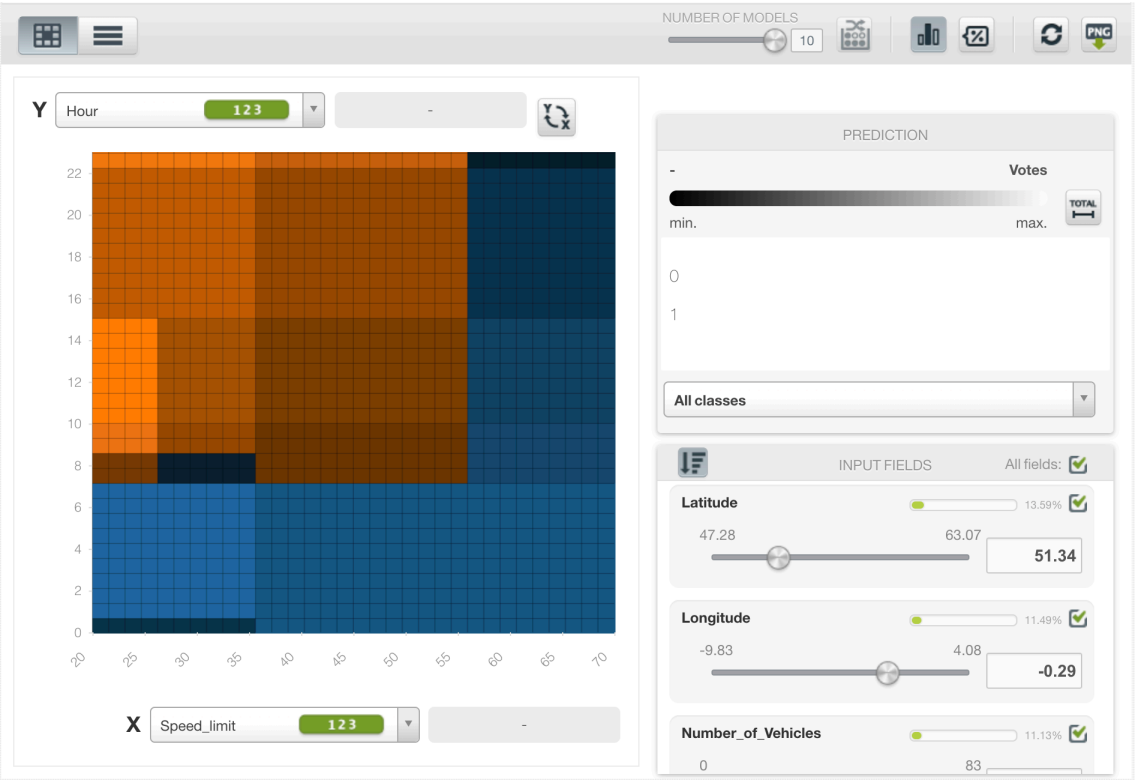



Ilustración 38: PCP con todas las características en la variable independiente.

Si habilitamos los demás campos de entrada en orden de importancia y vamos cambiando los valores de estos podemos ver como nuestro modelo se comportará en ciertas situaciones, lo que nos ayudará a sacar mejores conclusiones, por ejemplo en el caso que tomemos como variable dependiente la hora y la variable independiente el límite de velocidad, si asignamos los valores siguientes a las variables independientes más relevantes en nuestro modelo como son  Latitude=51.34 y Longitude=-0.29 (un punto cercano a Londres), Number_of_Vehicules=2, road_Type=single carriageway podemos observar como aparece una zona en color naranja que nos indica que la gravedad del accidente es leve entre el rango de velocidades de 20mph a 55mph y el rango horario de 08:00am a 22:00pm, lo que nos lleva a determinar que para vías de un sentido con una concentración de tráfico mayor, (en esas condiciones de localización cercana a Londres y en esa franja horaria la actividad laboral de las personas es mayor), es más probable reducir el riesgo de sufrir un accidente grave al circular entre unos límites de velocidad moderados, porque se puede ver claramente como al sobrepasar esos límites

se genera una segunda zona azul que es indicativo de que el accidente producido tiene consecuencias más severas.

## 4.6. PREDICCIÓN INDIVIDUAL

En BigML existen varias formas de realizar predicciones, en este apartado veremos la predicción individual donde podemos definir individualmente los valores de cada una de las características e ir variándolos para observar al instante cómo varía nuestra predicción, y poder sacar conclusiones.

Nosotros nos centraremos explícitamente en los campos de latitud y longitud porque son los campos sobre los que estamos centrando nuestras investigaciones como ya hemos comentado.

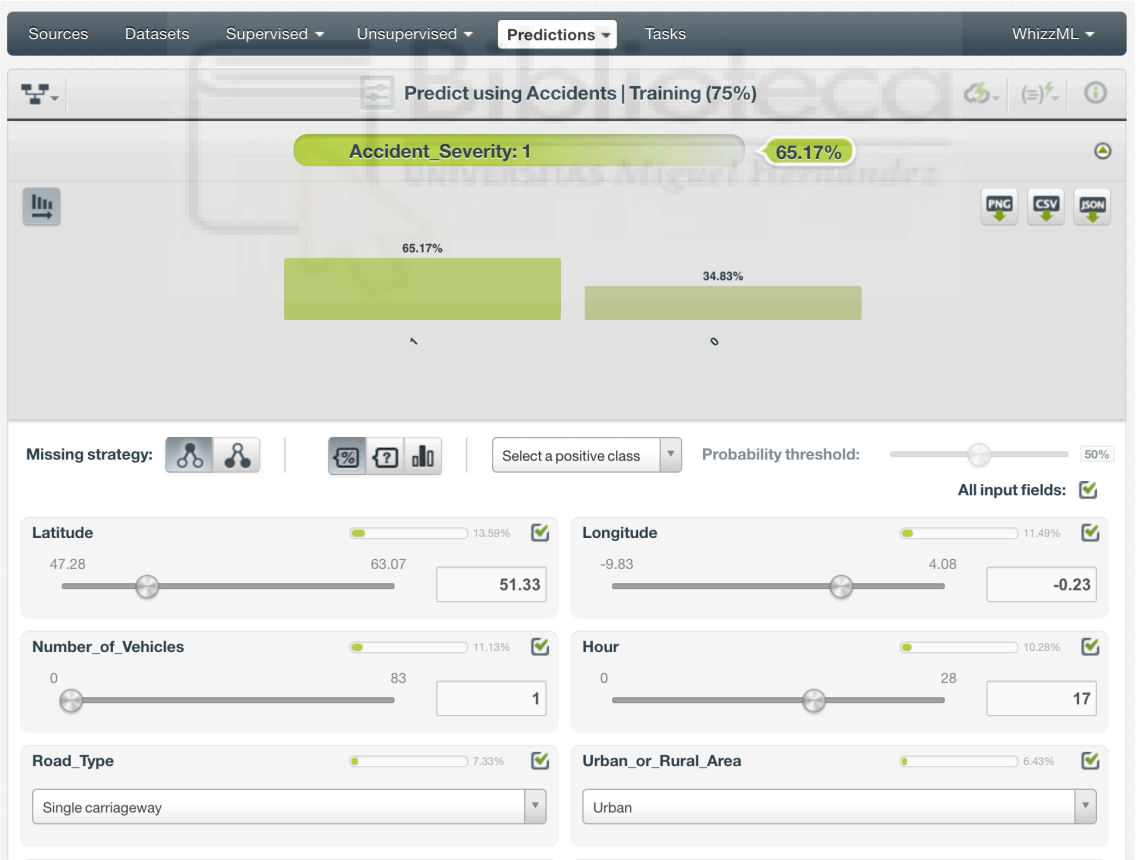



Ilustración 39: Predicción individual del modelo en Londres.

Si, por ejemplo, elegimos un punto cercano a Londres (Latitude=51.33 y Longitude=-0.23) en condiciones en que un único vehículo circula a alta

velocidad en una hora donde la concentración de tráfico es mayor y la vía es de un único sentido, observamos que la probabilidad de que se produzca un accidente con un grado de severidad alto es mayor, exactamente de un 65.17% frente al 34.83% de que el accidente sea leve.

En las mismas condiciones que antes pero cambiando la localización del accidente a un punto de Wales (Latitude=51.78 y Longitude=-4.24) que cuenta con una población de 3 millones de habitantes en todo el país y son sobretodo zonas rurales (donde el tráfico es reducido) podemos cerciorarnos que el porcentaje se reduce al 44.22%, de que se produzca un accidente grave y a un 55.78% de que el individuo sufra un accidente leve. El porcentaje para que se produzca un accidente grave es inferior, debido claro está, a que el vehículo que circula a gran velocidad, no tiene tantos obstáculos al ser esta una zona de poca afluencia de vehículos y podría ejecutar ciertas maniobras que no podría realizar en una zona urbana sin colisionar.

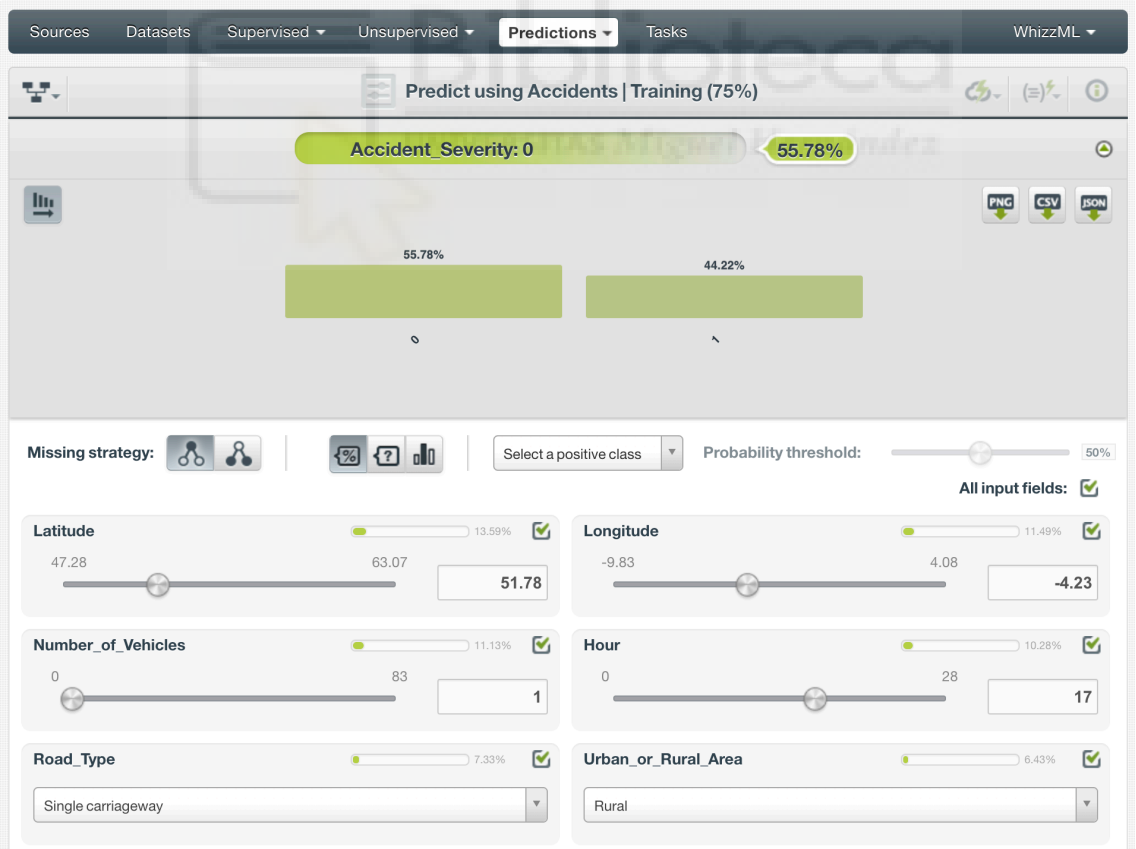



Ilustración 40: Predicción individual del modelo en Wales.

## 4.7. MATRIZ DE CONFUSIÓN

Vamos a proceder a obtener la matriz de confusión de nuestro bosque de árboles aleatorios para poder comparar nuestro problema en BigML que incluye la localización con nuestro problema en Python sin localización y ver si existe mucha diferencia en los resultados de uno a otro.

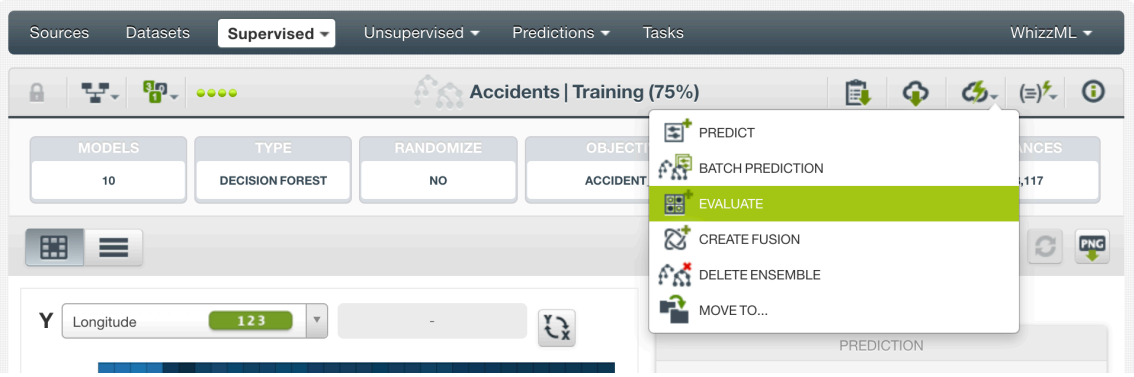



Ilustración 41: Cómo generar matriz de confusión del modelo.

Se selecciona la opción evalúate dentro de las opciones de predicción que nos ofrece BigML, después se procede a añadir el dataset con el 25% de las muestras que habíamos reservado.

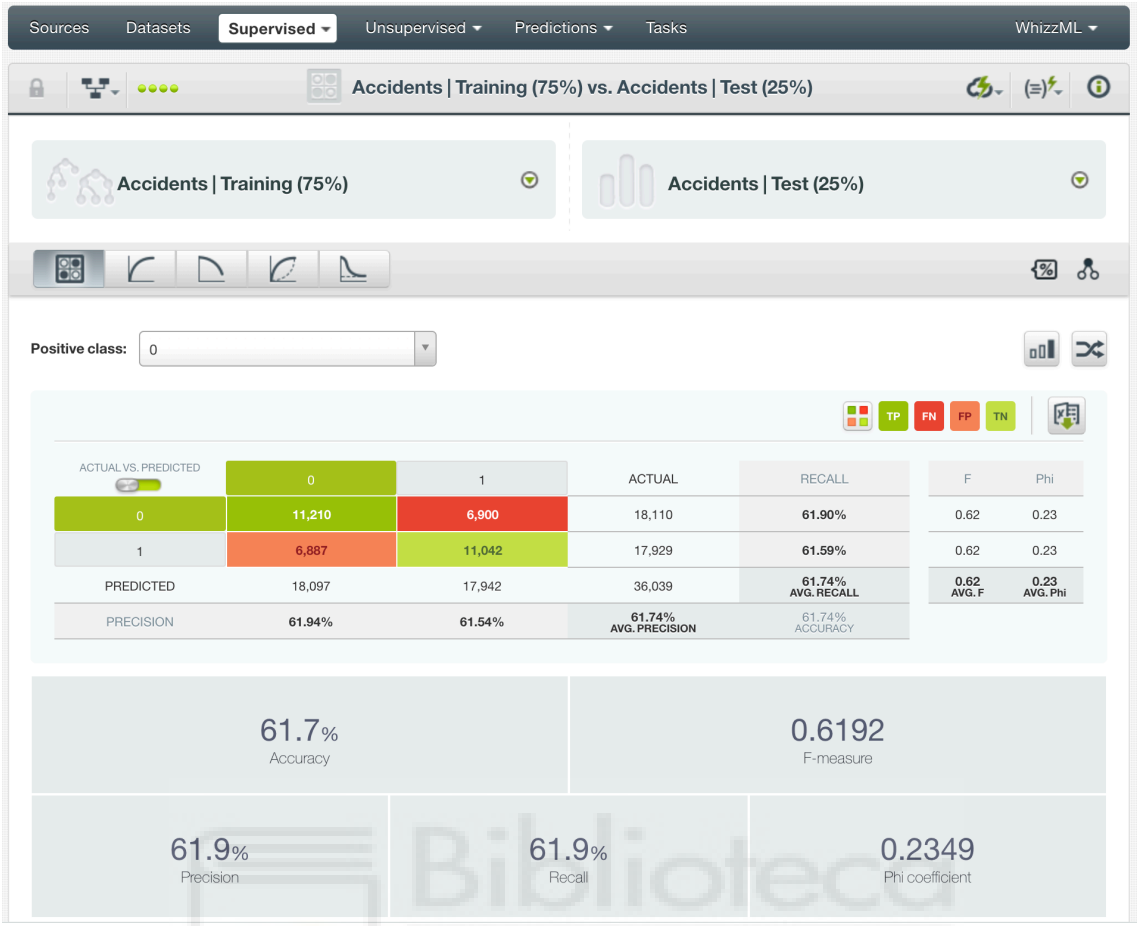



Ilustración 42: Matriz de confusión.

Se puede ver que el rendimiento de nuestro algoritmo desciende un poco con el generado en Python, pero no de forma significativa, por lo que es indiferente añadir la localización para calcular nuestras predicciones, además se puede determinar que es recomendable añadirlas porque nos aportan gran información y son de gran ayuda para saber donde se producen los casos y en qué condiciones y así poder tomar las medidas adecuadas para intentar reducir la gravedad de estos.

## 4.8. APRENDIZAJE NO SUPERVISADO EN BIGML

Igual que hicimos en Python vamos a proceder a implementar un algoritmo de Aprendizaje No Supervisado de agrupación y otro de asociación y ver que conclusiones podemos obtener de ellos.

### 4.8.1. ALGORITMO DE AGRUPACIÓN K-MEANS

Vamos a proceder a crear un análisis de cluster para agrupar los accidentes con características similares. Lo primero que vamos a hacer va a ser eliminar la columna de Accident_Severity. En el Aprendizaje No Supervisado se trabaja con datos sin etiquetar, BigMl nos ofrece esta posibilidad simplemente clicando en el lápiz que aparece justo al lado de la característica que queremos eliminar, en la imagen se puede apreciar mejor.


Ilustración 43: Cómo cancelar una característica de nuestro dataset.

Una vez eliminada la columna se apreciará una interrogación de color naranja.

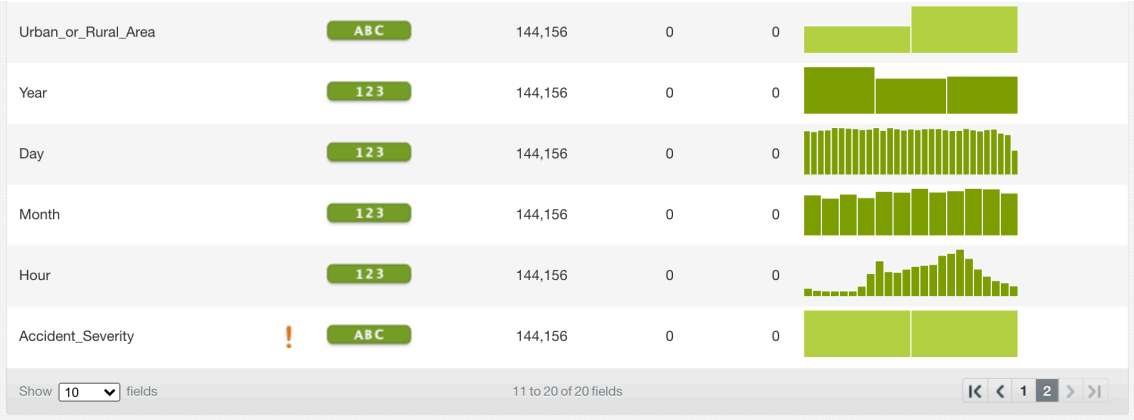



Ilustración 44: Resultado una vez cancelada.

Ahora podemos pasar a la implementación de nuestro algoritmo k-means, clicando en la pestaña de Aprendizaje No Supervisado cluster procedemos a su creación.

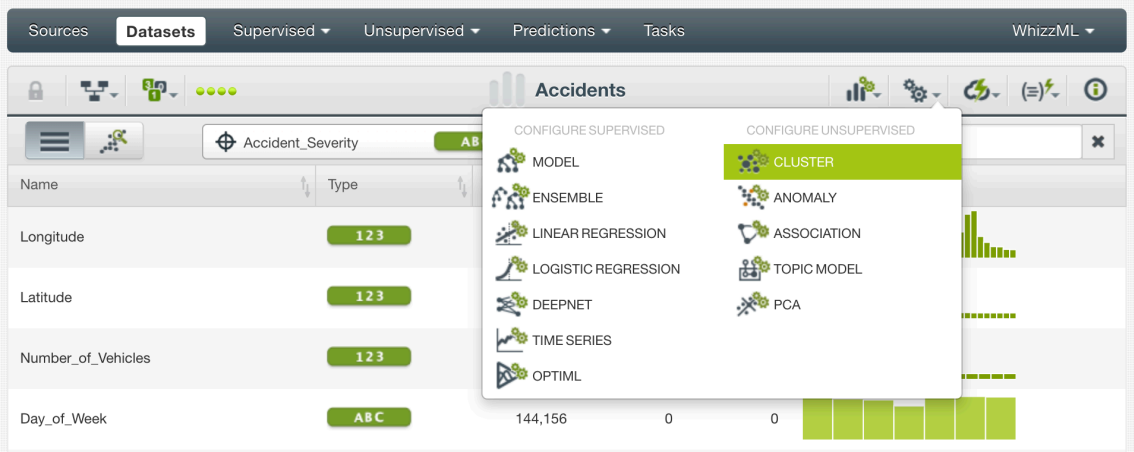



Ilustración 45: Cómo generar un cluster.

Podemos ajustar los parámetros de la forma que queramos o dejarlos por defecto. En nuestro caso como el algoritmo de clustering que queremos utilizar es el mismo que utilizamos en el lenguaje en Python, es decir, K-means y sabemos el número idóneo de clusters que queremos crear (como determinamos en nuestro algoritmo en Python, el número idóneo de clusters para nuestro modelo es 3) recordamos que este fue obtenido por medio del método del codo, por lo que en Number of clusters (K) seleccionaremos el número 3.

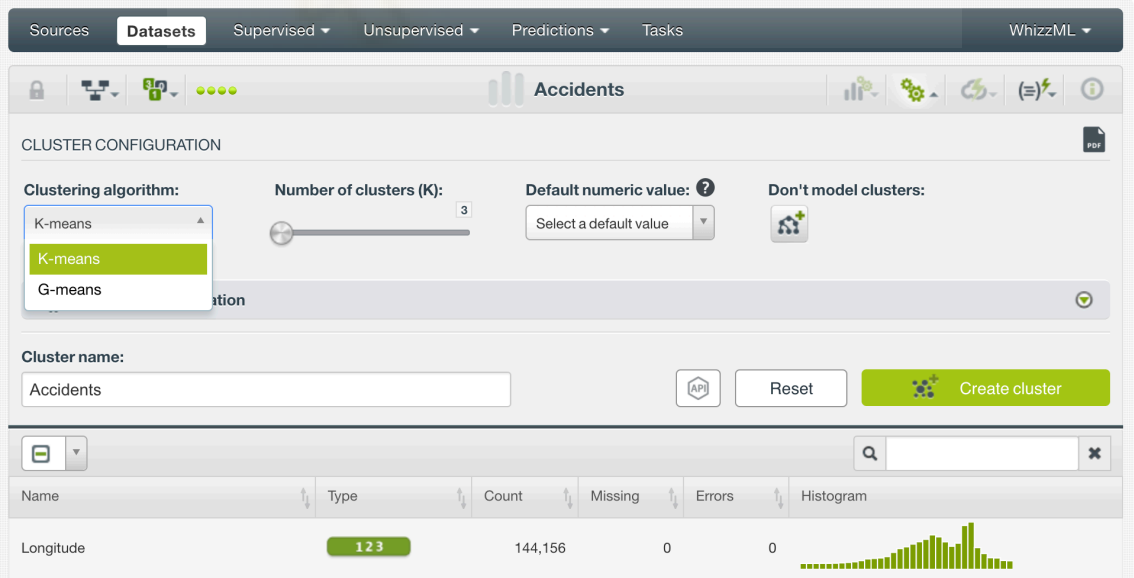



Ilustración 46: Cómo ajustar los parámetros en el cluster.

Resultado:

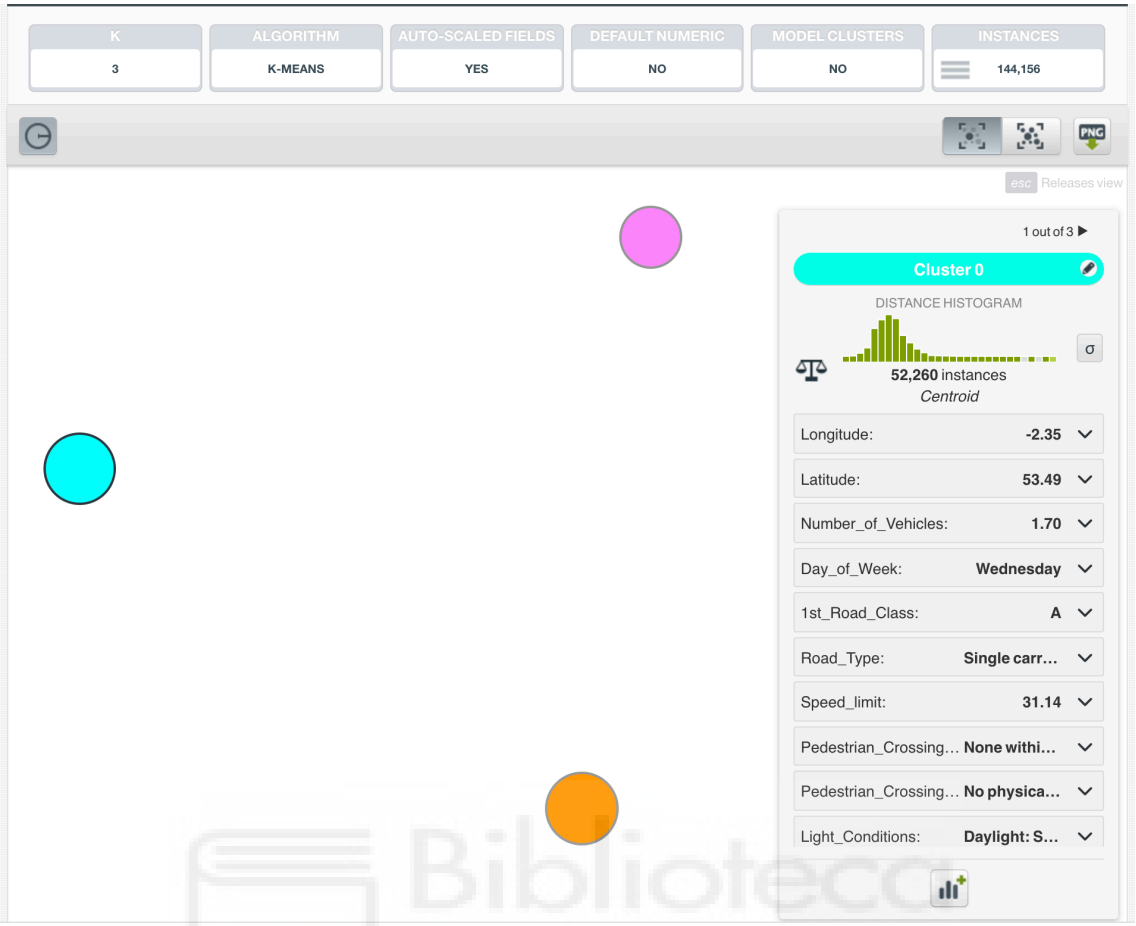



Ilustración 47: Visualización del algoritmo k-means.

Una vez generado nuestro algoritmo podemos apreciar cómo se agrupan las características y que valores adquieren en cada cluster, para poder determinar si alguna de estas agrupaciones es útil para nuestras conclusiones finales.

También nos ofrece la posibilidad mediante el icono  de poder crear un nuevo dataset a partir de cada agrupación obtenida, que puede resultar bastante útil para encontrar patrones en nuestros datos.

### 4.8.2. ALGORITMO DE ASOCIACIÓN

Vamos a emplear las reglas de asociación para ver como se comportan nuestras características. Buscaremos principalmente las reglas de asociación que incluyan la localización.

Como hemos anulado en el algoritmo anterior la columna de Accident_Severity ya no será necesario volverlo hacer, simplemente accedemos al dataset y creamos el modelo de Aprendizaje No Supervisado cliclando esta vez en association.

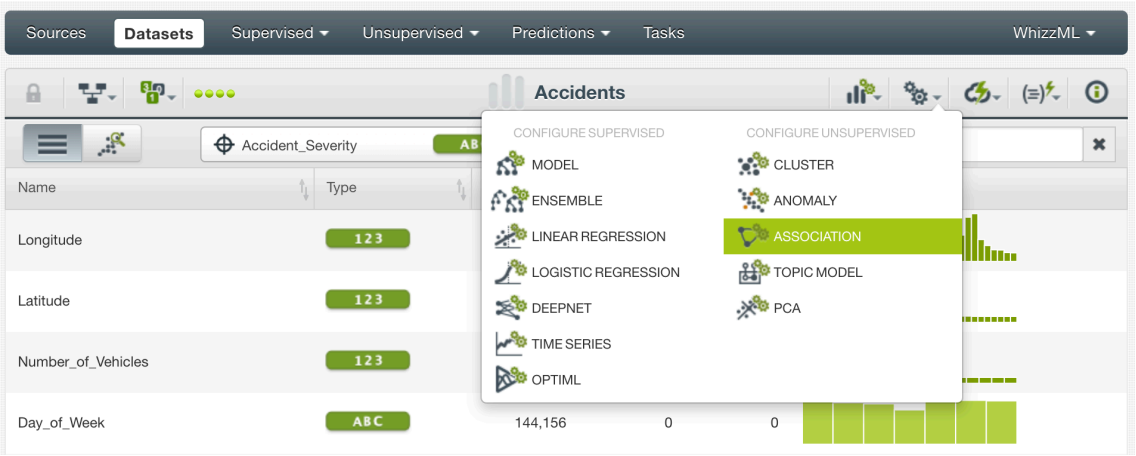



Ilustración 48: Cómo crear un descubrimiento de asociaciones.

Podemos ajustar los parámetros como queramos o dejarlos por defecto. Nosotros en nuestro caso los dejaremos por defecto.

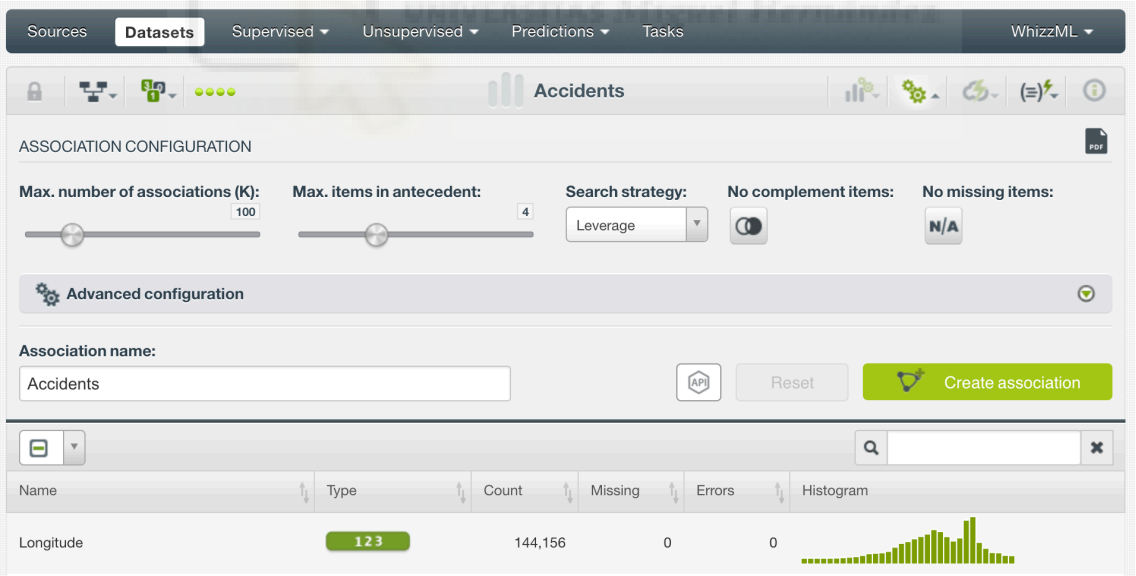
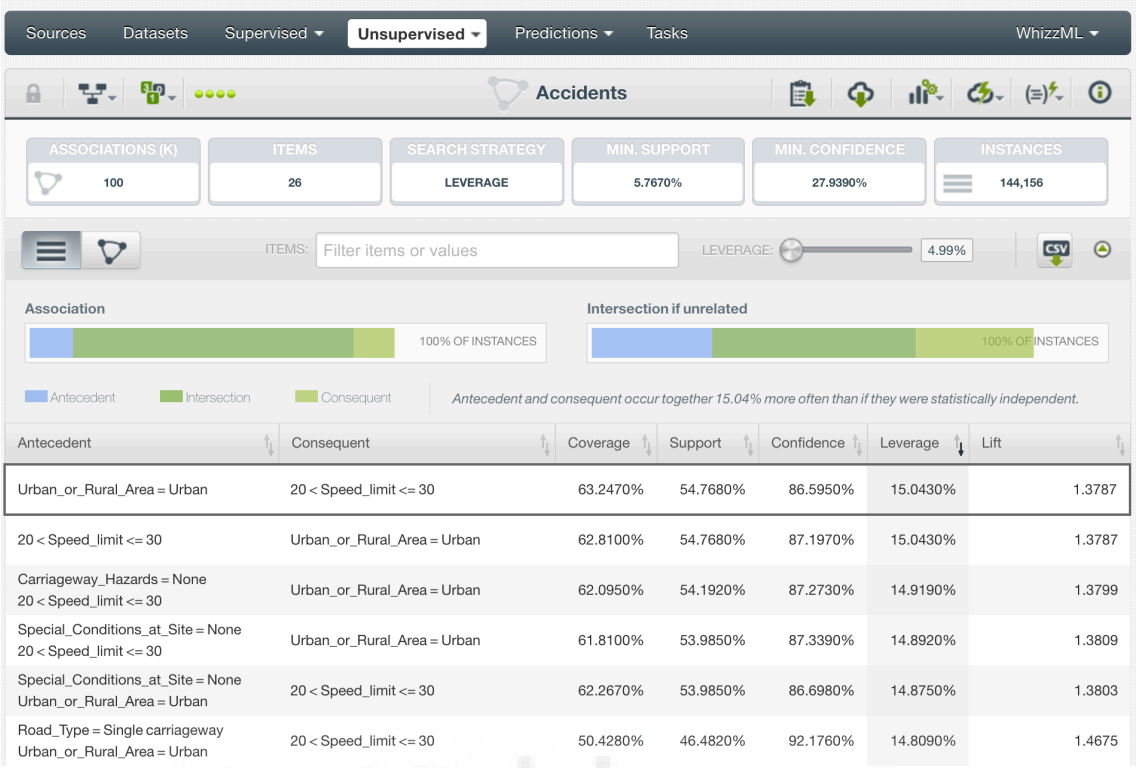



Ilustración 49: Ajustar los parámetros del descubrimiento de asociaciones.

Resultado:




| Antecedent | Consequent | Coverage | Support | Confidence | Leverage | Lift |
|---|---|---|---|---|---|---|
| Urban_or_Rural_Area = Urban | 20 < Speed_limit <= 30 | 63.2470% | 54.7680% | 86.5950% | 15.0430% | 1.3787 |
| 20 < Speed_limit <= 30 | Urban_or_Rural_Area = Urban | 62.8100% | 54.7680% | 87.1970% | 15.0430% | 1.3787 |
| Carriageway_Hazards = None<br>20 < Speed_limit <= 30 | Urban_or_Rural_Area = Urban | 62.0950% | 54.1920% | 87.2730% | 14.9190% | 1.3799 |
| Special_Conditions_at_Site = None<br>20 < Speed_limit <= 30 | Urban_or_Rural_Area = Urban | 61.8100% | 53.9850% | 87.3390% | 14.8920% | 1.3809 |
| Special_Conditions_at_Site = None<br>Urban_or_Rural_Area = Urban | 20 < Speed_limit <= 30 | 62.2670% | 53.9850% | 86.6980% | 14.8750% | 1.3803 |
| Road_Type = Single carriageway<br>Urban_or_Rural_Area = Urban | 20 < Speed_limit <= 30 | 50.4280% | 46.4820% | 92.1760% | 14.8090% | 1.4675 |

Ilustración 50: Descubrimiento de asociaciones.

Vemos que las asociaciones que se producen tienen cierta lógica. Si nos fijamos en el primer ejemplo, vemos que es normal que en las zonas urbanas la velocidad se encuentre limitada entre 20mph y 30 mph, porque dentro de las ciudades siempre se produce un descenso en la limitación de la velocidad, debido al tránsito de peatones y al aumento de tráfico. Esto se traduce en que la confianza entre el antecedente y el consecuente sea del 86.59%, es decir, las veces que aparecen juntos el antecedente y el consecuente.

Siguiendo la línea anterior hemos buscado asociaciones que incluyeran la localización para ratificar lo dicho anteriormente y hemos encontrado la siguiente asociación.

| | | | | | | |
|---|---|---|---|---|---|---|
| Urban_or_Rural_Area = Urban<br>20 < Speed_limit <= 30<br>Longitude > -0.13 | 51.415 < Latitude <= 51.685 | 12.3420% | 7.7740% | 62.9830% | 5.1820% | 3.0002 |

Ilustración 51: Ejemplo de reglas de asociación para la localización.

Donde se puede observar que esa localización pertenece a Londres y sigue los patrones de limitación de velocidad que hemos comentado.

Podemos sacar más conclusiones con las reglas de asociación, pero nosotros nos estamos centrando en determinar cómo afecta la localización a nuestro problema.

# CAPÍTULO 5

# 5. CONCLUSIONES GENERALES

Durante este trabajo hemos intentado ver qué técnicas funcionan mejor para la prevención de la gravedad de un accidente y ver si ciertas características como son la localización, el tiempo, o la velocidad son determinantes para detectar patrones que nos ayuden a identificar mejor dicha gravedad.

Me ha sorprendido gratamente los resultados obtenidos en todos los programas, destacaría especialmente el uso de Kepler.gl, donde este nos ha permitido darnos una idea visual de donde sea han producido nuestros accidentes más graves y si se pueden sacar conclusiones entorno a ello, que como ya vimos en el apartado de Kepler.gl se ha demostrado que sí, además permite una rápida visualización de los resultados.

Hemos utilizado un enfoque de aplicación, y valoración del resultado, porque al ser tantos los algoritmos aplicados hemos decidido que esta sería la mejor opción. Realizaremos una conclusión mucho más general de los resultados, dividiendo estos en Aprendizaje Supervisado y Aprendizaje No Supervisado, como hemos venido haciendo durante todo el problema.

Los mejores resultados obtenidos en las técnicas de Aprendizaje Supervisado se acercan a pronosticar el grado de la severidad del accidente, pero no hemos encontrado un modelo ideal que lo consiga al 100 por cien o se acerque a esa cifra, quizás en gran medida a una base de datos carente de algunas características, cómo podrían haber sido el tipo de vehículo, años del vehículo, si el conductor iba ebrio o drogado, si en el momento del accidente el conductor llevaba medidas de seguridad y otros factores y características que hubieran conseguido acercarse más a la predicción de un modelo ideal. Siendo honestos, considero que hubiera supuesto un aumento considerable en el

coste computacional. En mi opinión, hemos establecido un buen modelo predictivo para las herramientas de las que se disponía.

En cuanto a las técnicas de Aprendizaje No Supervisado hemos conseguido establecer importantes relaciones entre variables que nos permiten saber, por ejemplo, que es posible la agrupación en clases del conjunto de datos entrantes y observar el grado de asociación que se obtiene entre las características que más nos interesen. Concluir diciendo que los algoritmos de Aprendizaje No Supervisado son totalmente complementarios con los de Aprendizaje Supervisado y la suma de estos dos nos da una fantástica idea de como nuestros datos se comportan.

## 5.1 TRABAJO FUTURO

Vivimos en una época en la que se avecina una revolución tecnológica y este tipo de herramientas de inteligencia artificial se postulan como una de las principales bazas de este cambio. Esto se debe en gran medida, a que los ordenadores y dispositivos digitales hoy en día pueden considerarse como una extensión de nuestro cuerpo. Móviles, tablets y ordenadores todos puestos a nuestra disposición y alcance, ya sea para utilizarlos en forma de ocio o por necesidades de trabajo. También hay que tener en cuenta que los ordenadores hoy en día controlan las centrales nucleares, el suministro de luz, los misiles armados y muchas otras cosas. Por este motivo nos surgen preguntas e inquietudes como a los grandes genios de nuestro tiempo, desde Bill Gates o Elon Musk y también al añorado Stephen Hawking, todos ellos han mostrados sus dudas e inquietudes sobre este tema, argumentando que quizás las máquinas al tener tal control sobre la sociedad humana no deberían ser capaces de pensar por sí mismas ¿y si decidieran que los humanos somos prescindibles para este mundo un día?. Bueno lejos de dramatizar y de teorizar como en películas de ciencia ficción, en mi humilde opinión deberíamos seguir trabajando y cooperando como lo veníamos haciendo, porque como hemos visto en este trabajo fin de grado, también se pueden utilizar para salvar vidas, ayudar en la prevención de enfermedades, detectar fraudes fiscales y multitud

de aplicaciones más, que solo buscan la mejora de las áreas o campos donde se apliquen.

En un futuro, metodologías similares a las aplicadas en este problema junto con una mejora en el proceso de la recolección de datos y una mayor capacidad computacional que nos ayude a procesar modelos mucho más complejos en menos tiempo, puedan ayudar a pronosticar o realizar simulaciones mucho más avanzadas y cercanas a la realidad, pero por el momento nos conformaremos con decir que vamos en la dirección correcta para que esto suceda.

# BIBLIOGRAFÍA


10 datos sobre la seguridad vial en el mundo.

https://www.who.int/features/factfiles/roadsafety/es/

Análisis k-means con PCA

https://medium.com/@dmitriy.kavyazin/principal-component-analysis-and-k-means-clustering-to-visualize-a-high-dimensional-dataset-577b2a7a5fe2

Anexo: Países por tasa de muertes por siniestros de tránsito.

https://es.wikipedia.org/wiki/Anexo:Pa%C3%ADses_por_tasa_de_muertes_por_siniestros_de_tr%C3%A1nsito

Aprendizaje Automático: predicciones basadas en datos con BigML

https://cleverdata.io/en/bigdata-predictions-bigml/

BigML: Videos educativos de BigML.

https://bigml.com/education/videos

Cómo convertir la columna del DataFrame a Datetime en Pandas.

https://www.delftstack.com/es/howto/python-pandas/how-to-convert-dataframe-column-to-datetime-in-pandas/

Entrenamiento, validación y test con Scikit-learn.

https://www.analyticslane.com/2020/04/20/entrenamiento-validacion-y-test-con-scikit-learn/

Estos son los países más seguros y los más peligrosos para conducir.

https://www.autopista.es/noticias-motor/articulo/paises-mas-menos-seguros-circular-mundo-accidentes

Exploring 250,000+ Movies with Association Discovery.

https://blog.bigml.com/2015/12/17/exploring-250000-movies-with-association-discovery/

Géron, Aurélien. (2017). Hands-On Machine Learning with Scikit-Learn and TensorFlow. Grid Search, pp. 102-108.

Guía del usuario en pandas.

https://pandas.pydata.org/pandas-docs/stable/user_guide/index.html

Implementación de algoritmos de Aprendizaje Automático.

https://scikit-learn.org/stable/user_guide.html

Información de librería matplotlib.

https://es.wikipedia.org/wiki/Matplotlib

Información de librería numpy.

https://es.wikipedia.org/wiki/NumPy

Información de librería pandas.

https://es.wikipedia.org/wiki/Pandas

kaggle: 1.6 million UK traffic accidents.

https://www.kaggle.com/daveianhickey/2000-16-traffic-flow-england-scotland-wales

Kepler.gl de Mapbox: mapas interactivos y visualización de datos.

http://www.geomapik.com/webmapping-gis/kepler-gl-mapbox-mapas-interactivos-spatial-data/

Librerías de Python para Machine Learning.

https://iartificial.net/librerias-de-python-para-machine-learning/

Los apocalípticos que han contagiado a Musk y Gates su miedo a la IA.

https://www.eldiario.es/hojaderouter/ciencia/inteligencia_artificial-robots-Elon_Musk-Bill_Gates-Steve_Wozniak_0_405959588.html

Mapa de calor.

https://gmaclenn.github.io/articles/airport-pca-analysis/

Mirjalili, Vahid., & Raschka, Sebastian. Python Machine Learning. (2019). Tratar con el desequilibrio de clases, pp. 214-220.

Pandas en Python, con ejemplos -Parte II- Lectura-Escritura, Merge & GroupBy

https://jarroba.com/pandas-python-ejemplos-parte-ii-io-merge-groupby/

Principal Component Analysis & Clustering with Airport Delay Data.

https://gmaclenn.github.io/articles/airport-pca-analysis/

Probabilidad de lesiones en accidente de automóvil en BigML.

https://bigml.com/user/czuriaga/gallery/model/50f981e53b56354d2a00063a

Prediciendo los ganadores del Oscar 2018 en BigML.

https://blog.bigml.com/2018/03/01/predicting-the-2018-oscar-winners/

Python slice ().

https://www.programiz.com/python-programming/methods/built-in/slice

Reglas de asociación en python.

https://www.analyticslane.com/2018/08/31/reglas-de-asociacion-market-basket-analysis/

Spyder: Visión general.

https://www.spyder-ide.org/

Time deltas.

https://pandas.pydata.org/pandas-docs/stable/user_guide/timedeltas.html

Titanic: Love in Data Analytics - - Aprendizaje Automático – 2020.

https://es.sciencewal.com/27891-titanic-love-in-data-analytics-1b38b40d7247-94

Theobald, Oliver. (2017). Machine Learning For Absolute Beginners. Building a model in python, pp. 119-129.

Unir y combinar DataFrames con pandas en Python.

https://www.analyticslane.com/2018/09/10/unir-y-combinar-dataframes-con-pandas-en-python/

Validación cruzada.

https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6