Escuela Politécnica Superior

Generación de datos estructurados para problemas de clasificación en entornos desequilibrados

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:

Pedro Juan Baeza Gómez

Tutor/es:

Jorge Calvo Zaragoza

Juan Ramón Rico Juan

José Javier Valero Más

Septiembre 2016

Universitat d'Alacant
Universidad de Alicante

Generación de datos estructurados para problemas de clasificación en entornos desequilibrados

Adaptación del algoritmo de generación de prototipos SMOTE para datos estructurados del tipo cadena

Autor
Pedro Juan Baeza Gómez

Directores
Jorge Calvo Zaragoza
Departamento de Lenguajes y Sistemas Informáticos
Juan Ramón Rico Juan
Departamento de Lenguajes y Sistemas Informáticos
José Javier Valero Más
Departamento de Lenguajes y Sistemas Informáticos

Grado en Ingeniería Informática

Escuela Politécnica Superior

ALICANTE, 11 de septiembre de 2016

Preámbulo

"Hoy en día multitud de investigaciones se llevan a cabo en el campo de la inteligencia artificial, avances que nos otorgan grandes facilidades en nuestra vida cotidiana, que ayudan a otras ramas a predecir problemas que perjudicaran al ser humano en un futuro y sobretodo obtener técnicas que permitan mejorar la vida de unos pocos, ofreciéndoles de nuevo la posibilidad de realizar tareas que son poco valoradas para muchos. El poder aportar algo que ayude a estas tareas es mi principal motivación a seguir con toda mi formación.

Este proyecto refleja mis ganas de seguir estudiando sobre estos temas, ya que todavía existen muchas cosas por descubrir y mejorar. Me gustaría agradecer a mis tutores Jorge Calvo Zaragoza, Juan Ramón Rico Juan y José Javier Valero Más por guiarme a la hora de desarrollar este proyecto y querer transmitirme sus conocimientos. A mis compañeros y en especial a Francisco José Castellanos los cuales me han prestado su ayuda en todo momento y han sabido aguantarme en estos cuatro años de carrera.

Por último aprovechar para mencionar a todas aquellas personas que me han formado y apoyado a seguir hacia delante en mi camino durante estos años. En especial agradecimiento a mis padres *Isidora Gómez* y *Pedro González*, así como a mis hermanos *Brahis Baeza* y *Nerea Baeza*, por el apoyo incondicional que siempre me habéis transmitido así como toda vuestra ayuda a la hora de esquivar los obstáculos con los que me he podido encontrar, sin vosotros ahora no estaría aquí sentado escribiendo estas frases.

Y ya para acabar agradecerte a ti todo lo que hoy soy, gracias por enseñarme a luchar por mis objetivos, a no rendirme nunca y sobretodo a no consentir que nadie me frene a lograr mis metas, por todo esto y más cosas dedicarte este trabajo y todos mis futuros logros a ti mi abuelo *Agustín Gómez García*"

A mis padres, hermanos y amigos,
sin los cuales esto no hubiera sido posible.

El destino mezcla las cartas,
y nosotros las jugamos.

Arthur Schopenhauer.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

Desde hace años, la idea de conseguir que una máquina tenga comportamiento automático e inteligente ha sido la causa de una amplia investigación en multitud de disciplinas. El interés por idear sistemas capaces de sustituir a los seres humanos en determinadas tareas ha provocado el desarrollo de diferentes estrategias para llevar a cabo el aprendizaje de las máquinas.

En el campo de la informática, la rama de la inteligencia artificial encargada de este proceso es la denominada *machine learning*. El principal objetivo es desarrollar técnicas capaces de conseguir que un computador aprenda una serie de comportamientos generales obtenidos de un conjunto de datos suministrados en forma de ejemplos. Dependiendo del conjunto de datos que se le suministren a la máquina se podrá realizar un entrenamiento supervisado, donde los datos de entrada llevan a su vez su salida correspondiente, o un entrenamiento no supervisado donde los datos no aparecen etiquetados. *Pattern Recognition* es la rama de *machine learning* encargada de llevar a cabo esta tarea.

En *Pattern Recognition* nos podemos encontrar con que la representación de los datos puede ser mediante estructuras como cadenas, arboles, grafos; o espacios de vectores, donde la representación de la información se basa en vectores de características numéricas las cuales permiten representar a las diferentes clases existentes en los datos.

Que existan estos dos tipos de representación de la información es debido a las ventajas que puede aportar una con respecto a la otra. Para los datos de tipo estructurado nos encontramos con que nos permite una amplia representación de la información de los datos, ya que son estructuras más complejas, cosa que otorga robustez. Pero con este modo de representación de la información tenemos que existe un menor número de algoritmos, ya que es complejo tratar con ellos. Por otro lado, la representación de los datos mediante un espacio de vectores nos encontramos con que ocurre todo lo contrario, ya que tiene un amplio rango de algoritmos que los tratan por su facilidad de manejo, pero con el inconveniente de que la representación de la información se encuentra más limitada.

Recientemente se publicó una técnica [Calvo-Zaragoza et al., 2016] capaz de trabajar con datos de tipo estructurado como si de datos de espacio de vectores se tratasen. Consiste en un mapeo de la información de datos estructurados del tipo cadenas a un espacio de vectores de características, para posteriormente generar muestras sintéticas

de datos del segundo tipo. Para realizar este proceso del mapeo se extrae un subconjunto $R = \{r_1, \ldots, r_n\}$ del conjunto total de los datos, los cuales van a ser los pivotes sobre los cuales se va a realizar el proceso de mapeo de los datos. La transformación de los datos consiste en que por cada muestra, x del conjunto total de datos se traducirá en un vector $(v_1, v_2, \ldots, v_m)$, $m = |R|$ donde cada componente v_i será la distancia entre x y el pivote r_i. Con esta técnica se consigue una primera aproximación de como trabajar con datos estructurados en algoritmos que solo permitían datos del tipo vectores de características.

En este campo de la informática nos podemos encontrar con distintos problemas a la hora de trabajar con el conjunto de datos suministrado, los cuales ocasionan deficiencias en la clasificación de nuevos prototipos enviados al sistema. En muchas aplicaciones de aprendizaje supervisado, hay una diferencia significativa entre las probabilidades de pertenecer a cada una de las distintas clases de los datos suministrados, debido a que en el conjunto de datos una clase contiene un menor número de muestras (Clase minoritaria) con respecto al resto (Clase mayoritaria). Esta situación se conoce como el problema de desequilibrio de clase.

Hoy en día nos encontramos con diferentes métodos que permiten paliar este problema. Existen algoritmos que por un lado se centran en la disminución de la clase mayoritaria mediante técnicas de selección de prototipos como *Editing* [Wilson, 1972], *Condensing* [Hart, 1968] o *FN* y *NE* [Rico-Juan and Iñesta, 2012], su procedimiento consiste en realizar una extracción reducida del conjunto de datos total de la clase, a la vez que se conservan los parámetros de clasificación de la misma. Por otro lado, nos encontramos con algoritmos que pretenden realizar un sobremuestreo de la clase con el menor número de prototipos como *Reduction by Space Partitioning* (RSP) [Sánchez, 2004], *Evolutionary Nearest Prototype Classifier* (ENCP) [Fernández and Isasi, 2004] o *Mean Squared Error* (MSE) [Decaestecker, 1997], su método se centra en la generación de nuevas muestras artificiales a partir del conjunto de datos original, en ambos casos la finalidad es conseguir que ambas clases mantengan un equilibrio en el número de prototipos, para que las probabilidades de pertenecer a una clase no se vea afectada por los datos suministrados.

Todo este conjunto de algoritmos consiguen llevar a cabo mejoras en los resultados de clasificación, pero todo dependiendo del modelo en el que se representen las muestras. Entre todo este conjunto de métodos destaca *Shyntetic Minority Over-sampling Technique* (SMOTE) [Chawla et al., 2002] como uno de los algoritmos más conocidos para llevar a cabo el sobremuestreo de la clase minoritaria, este método se encarga de generar muestras sintéticas de la clase con menos prototipos hasta conseguir igualarla a la clase mayoritaria, el principal problema que puede tener este algoritmo es que solo trabaja con datos representados mediante espacios de vectores, lo cual limita su gran funcionalidad.

Este proyecto se centra en solucionar el problema del desequilibrio de los datos, directamente sobre el espacio de datos de estructurados, en este caso trabajando sobre el dato estructurado de tipo cadena. Con el fin de estudiar si el trabajar directamente con ellos

aporta la mismas ventajas que nos ofrecen los datos estadísticos a la hora de realizar técnicas de sobremuestreo de los prototipos sin la necesidad de realizar transformaciones intermedias de un tipo de dato al otro.

1.1. Objetivos

Este documento presenta una investigación realizada en el Departamento de Lenguajes y Sistemas Informáticos que pretende alcanzar los objetivos que se definen a continuación.

- Llevar a cabo un estudio del algoritmo de generación de prototipos SMOTE, y sus dos variantes posteriores que pretenden sobremuestrear los bordes, *BorderLine - SMOTE* [Han et al., 2005].

- Investigación de técnicas que permitan realizar la generación de datos sintéticos trabajando directamente con datos estructurados del tipo cadenas, con el fin de extrapolar dichos métodos al nuevo algoritmo.

- Implementación de los algoritmos actuales, que trabajan con datos estadísticos, con el fin de tener una comprensión clara de los algoritmos y su funcionamiento.

- Desarrollo del algoritmo de clasificación supervisada kNN [Cover and Hart, 1967] y del algoritmo de validación cruzada [Refaeilzadeh et al., 2009] para llevar a cabo las pruebas.

- Preparación de los datos para llevar a cabo una correcta experimentación. Implementación del problema teniendo en cuenta las técnicas estudiadas.

- Evaluación de los resultados y extracción de las conclusiones con respecto al algoritmo desarrollado.

- Estudio de posibles trabajos futuros en consecuencia a los resultados obtenidos del sistema.

1.2. Metodología

Con el fin de lograr un correcto desarrollo del proyecto se distinguieron un conjunto de fases las cuales permitirán llevar a cabo una metodología de trabajo definida por la siguiente estructura.

- *Fase de búsqueda y estudio*: Antes de comenzar a llevar a cabo una implementación del proyecto que resuelva el problema propuesto sera necesaria una fase de búsqueda y estudio previa la cual nos otorgue información de lo que existe actualmente en el campo de trabajo del problema a resolver. La finalidad de esta fase es tener una comprensión clara de los conceptos del ámbito en el que se mueve el problema, así como conocer posibles formas de resolución del problema en base a lo ya existente. Toda esta búsqueda y estudio se detalla en la sección 2.

- *Fase de preparación de datos y diseño del sistema*: Plantearemos el sistema a desarrollar indicado en la sección 3.3, teniendo en cuenta todos los conocimientos obtenidos en la fase previa, además llevaremos a cabo la preparación de los datos, de la forma que se detalla en la sección 3.2 de tal forma que puedan ser empleados en el nuevo sistema y realizaremos una selección de aquellos que realmente aportaran información útil a la investigación del problema.

- *Fase de implementación y depuración*: En esta fase se procederá a decidir el entorno de trabajo en el que se desarrollara el problema, así como el lenguaje que será empleado para la programación. Tras la implementación tendremos que llevar a cabo un proceso de depuración del código el cual nos permita detectar errores en la implementación del problema y evitar obtener resultados no correctos. Todo el proceso de implementación y sus problemas lo podemos observar en la sección 3.4.

- *Fase de experimentación y análisis de resultados*: Realizaremos la experimentación sobre los datos seleccionados anteriormente y consigo llevará el análisis de los resultados obtenidos, los cuales nos permitirán observar si el problema ha sido resuelto o por el contrario no se obtiene ninguna mejora. Todos estos resultados y conclusiones se pueden observar en la sección 4.

- *Fase de reflexión*: Tras estudiar los resultados obtenidos, en esta fase reflexionaremos sobre posibles trabajos futuros que podría otorgar este proyecto, los cuales detallaremos en la sección 5, así como si seria realmente útil continuar realizando el sobremuestreo de datos de tipo estructurado.

1.3. Estructura del trabajo

Para facilitar su lectura, el contenido de este documento se encuentra organizado en capítulos y secciones. A continuación se describe la estructura que vamos a seguir.

- ***Capítulo 1: Introducción*** ⇒ Capítulo introductorio que abarca los fundamentos del proyecto y describe los objetivos que se pretenden alcanzar.

- ***Capítulo 2: Estado del arte*** ⇒ Dedicada a proporcionar una base teórica general del ámbito en el que se mueve el problema a resolver, así como los objetivos a abordar.

- ***Capítulo 3: Desarrollo*** ⇒ Refleja los detalles del desarrollo del problema, entre los cuales se encuentran la preparación de datos, diseño del sistema y la preparación de las pruebas.

- ***Capítulo 4: Experimentación*** ⇒ Muestra los resultados de la experimentación y un análisis de ellos.

- ***Capítulo 5: Conclusiones y perspectivas de futuro*** ⇒ Detalla el conjunto de reflexiones obtenidas en cuanto al estudio de los resultados de las pruebas del proyecto, así como de posibles trabajos futuros a abordar.

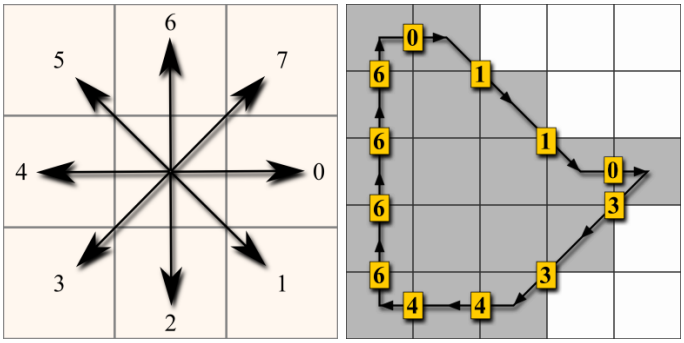
2. Estado del arte

2.1. Conceptos previos

Para poder situarnos bien en contexto del tema a tratar es necesario tener claro una serie de conceptos con los que luego trabajaremos a lo largo de la memoria, por ello en este apartado situaremos al lector un poco en contexto detallando términos que posteriormente ayudaran en la comprensión.

2.1.1. Código de Freeman

El método conocido como el Código de Freeman [Freeman, 1961] permite realizar la codificación de imágenes monocromáticas mediante cadenas de dígitos comprendidos entre 0 y 7. El proceso que lleva a cabo este método consiste en realizar la selección de un pixel de partida de la imagen indicando su posición y una vez este se encuentra seleccionado el algoritmo de codificación se mueve a lo largo del borde de la imagen y, en cada paso, registra el dígito que representa la dirección de su movimiento (Fig. 2.1a). Esto continua hasta que el codificador retorna a la posición inicial desde la cual parte, obteniendo al final del proceso una secuencia de dígitos los cuales nos codifican el borde de una imagen. En la fig. 2.1b tenemos que partiendo del pixel situado más a la derecha el codificador comienza a recorrer el contorno de la mancha obteniendo al final la cadena "334466660110" la cual codifica la imagen.



(a) Codificación del movimiento (b) Codificación contorno

Figura 2.1.: Código de Freeman

2.1.2. Distancia de edición

La distancia de edición o distancia de Levenshtein, calcula el número mínimo de operaciones que se deben de realizar para transformar una cadena de caracteres en otra. Entre las operaciones que se mencionan nos encontramos con:

- Inserción: consiste en añadir un nuevo carácter a la cadena que se pretende transformar y para el algoritmo esto vendría a conllevar una operación de coste 1.

- Borrado: consiste en suprimir un carácter ya existente en la cadena que queremos transformar y al igual que la inserción tendría un coste de 1 esta operación.

- Sustitución: consiste en sustituir un carácter de la cadena a transformar, con uno de la cadena a la que se quiere llegar a obtener. En cuanto al coste de esta operación tendremos dos situaciones, una en la que realizaremos la sustitución de dos caracteres diferentes por lo que tendrá un coste de 1 y por otro lado la sustitución de dos caracteres iguales por lo que el coste de esta operación sería 0.

Este algoritmo se ayuda de la programación dinámica para llevar a cabo su cálculo. El algoritmo ira rellenando una matriz de dimensiones $(n+1) \cdot (m+1)$, donde n y m son la longitud de las dos cadenas a comparar. En cada iteración del proceso se ira comprobando de las tres posibles operaciones cual de ellas otorga el menor coste de transformación, hasta conseguir rellenar la matriz al completo y obtener el resultado de la distancia en la posición $(n+1) \cdot (m+1)$ de la matriz.

2.1.3. Técnica de clasificación de los vecinos más cercanos (kNN)

El algoritmo kNN es un clasificador supervisado que compara un conjunto de datos etiquetados frente a una muestra que se quiere clasificar. Para ello, previamente se entrena al clasificador con un conjunto de entrenamiento, que consiste en una base de datos con ejemplos de los cuáles se conoce su pertenencia a determinada clase o tipo de dato.

kNN compara todo el conjunto de entrenamiento con la muestra a clasificar mediante el cálculo de la distancia o el grado de disimilitud, pudiendo ser obtenidos a través de diferentes métodos. Una vez calculado el grado de disimilitud o la distancia del elemento a clasificar frente a todo el conjunto de entrenamiento, se ordenan los resultados de menor a mayor, y en ese momento entra en juego el parámetro k del algoritmo kNN, que representa el número de vecinos que deben tenerse en cuenta en la clasificación.

Se emplean los primeros k ejemplos de la lista ordenada y se genera un histograma (vector cuyos elementos contabilizan el número de ocurrencias de un tipo de dato concreto) para detectar cuál es la etiqueta que más se repite en ese subconjunto de k ejemplos, siendo la que finalmente determinará el resultado de la clasificación. Este algoritmo de clasificación es el que emplearemos para llevar a cabo la implementación de nuestro sistema, así como las pruebas que se le realizarán.

2.1.4. Medida de exactitud F1

En análisis estadísticos de clasificación binaria, la F1 es una medida de la exactitud de una prueba. Está especialmente indicada en experimentos donde el número de muestras por clase están desequilibrados. En esta medida se tienen en cuenta los valores de precisión (p) y de exhaustividad (r), donde la p seria una medida que reflejaria que de todos los valores que el sistema ha dicho que un prototipo pertenece a la clase X cuantas veces realmente lo era y por otro lado la r mediría que para todas las veces que el sistema tendría que haber dicho que el prototipo pertenece a la clase X cuantas veces lo ha hecho correctamente. La medida F1 puede ser interpretada como un promedio ponderado de la precisión y la exhaustividad, en donde la máxima puntuación alcanzada será un valor de 1 y el mínimo será 0.

$$F1 = 2 \cdot \frac{precision \cdot exhaustividad}{precision + exhaustividad} \tag{2.1}$$

2.1.5. Validación cruzada

La validación cruzada o cross-validation es una técnica utilizada para evaluar los resultados de un análisis estadístico. Consiste en dividir el conjunto total de datos en n particiones y posteriormente ejecutar el algoritmo tantas veces como particiones se hayan generado, teniendo que para cada ejecución del sistema una de las particiones formara el conjunto de prueba y el resto formara el conjunto de entrenamiento del sistema (Fig. 2.2). Tras finalizar el proceso se recogerán todos los resultados generados y se realizara un promedio entre ellos. Lo mas común es dividir la BD en 10 particiones y en algunos casos en 5 particiones.

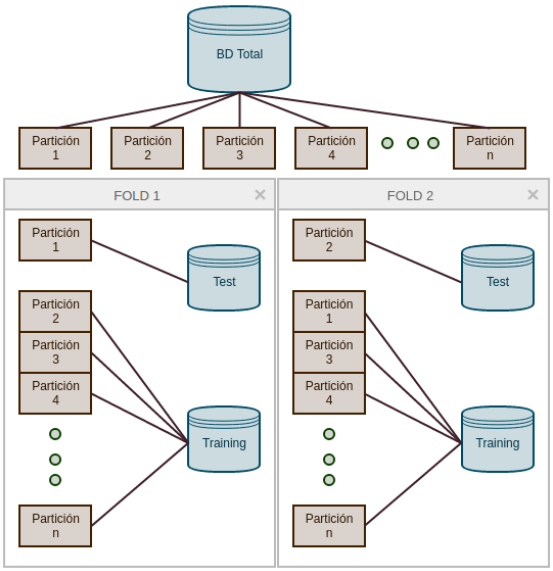


Figura 2.2.: Esquema validación cruzada para 2 iteraciones.

2.1.6. Test de suma de rangos Wilcoxon

La prueba de suma de rangos de Wilcoxon [Demšar, 2006] se trata de una prueba no paramétrica que sirve para aceptar o rechazar la hipótesis nula (H_0), es decir, compara los valores de dos distribuciones dadas para afirmar si son o no similares con cierto grado de confianza. En este proyecto los test de significancia se usarán para comparar los resultados obtenidos por el clasificador kNN ejecutando el nuevo sistema y determinar si otorga mejores resultados que los del mismo clasificador pero sin ejecutar el programa de generación de prototipos. Como los resultados obtenidos no tienen porqué restringirse a una distribución normal, lo más conveniente es usar un tipo de test no paramétrico como éste.

2.2. Estudios previos

En una fase previa al desarrollo del proyecto se llevó a cabo la búsqueda y estudio de las diferentes técnicas existentes que pretenden solucionar el problema de desequilibrio de los datos. Además fue necesaria la búsqueda de métodos que permitieran llevar el manejo de datos estructurados a algoritmos que manejaban datos de tipo sintético.

2.2.1. Algoritmo "Synthetic Minority Over-sampling Technique" (SMOTE)

Una de las técnicas más empleadas para el sobremuestreo de prototipos es SMOTE. Esta técnica surgió en el año 2002, su objetivo principal es el de realizar un sobremuestreo de la clase que tenga el menor número de prototipos, llevando a cabo la generación de muestras sintéticas en base a los datos originales proporcionados al sistema.

El proceso de sobremuestreo consiste en ir seleccionando cada prototipo de la clase minoritaria e ir introduciendo muestras sintéticas a lo largo de los segmentos que se generan entre la muestra escogida y todos sus k-Vecinos (Sec. 2.1.3) más cercanos seleccionados de forma aleatoria, este proceso se lleva a cabo hasta que se consiga igualar el número de prototipos de la clase minoritaria con el de la mayoritaria. Dependiendo del sobremuestreo necesario para igualar las dos clases, tendremos que por cada prototipo se podrán generar una o más muestras sintéticas a los largo de los segmentos generados. La implementación que se empleo usaba una k de 5 vecinos por cada muestra.

En el caso de tener por un lado 12 muestras de la clase mayoritaria y 7 de la clase minoritaria, el sistema realizaría un sobremuestreo del 100 %, es decir que por cada muestra de la clase minoritaria generara unicamente un prototipo sintético hasta que se consigan igualar ambas clases. El ejemplo se muestra en la Fig. 2.3 donde los cuadrados rojos vendrían a ser la clase mayoritaria, los círculos azules la clase minoritaria y los círculos verdes serian los prototipos generados sintéticamente.

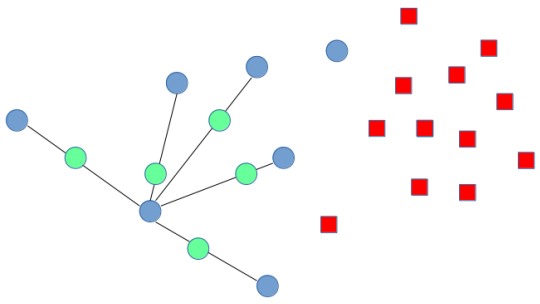


Figura 2.3.: Algoritmo SMOTE original

2.2.2. Algoritmo BorderLine - SMOTE

En 2005 surgió una variante del algoritmo SMOTE, en el cual a diferencia del original se centra en llevar a cabo el sobremuestreo de los prototipos que conforman el borde que separan a una clase de la otra.

La selección de este subconjunto que conforma el borde de la clase minoritaria se realiza como se muestra en el proceso indicado en el algoritmo 1. Una vez se detectan los prototipos que conforman el borde, se realiza la técnica del SMOTE original sobre dicho conjunto. Esta es una de las dos técnicas de sobremuestrear el borde que se propusieron, denominada *BorderLine - SMOTE1*.

La segunda técnica que se propuso llevaba a cabo la selección del borde de la misma forma, la diferencia con respecto a la anterior es que en *BorderLine - SMOTE2* no solo se infla la clase minoritaria teniendo el cuenta los prototipos de ella misma, sino que también se realiza el proceso sobre los prototipos de la clase mayoritaria, teniendo en cuenta que sobre los vecinos de su misma clase genera los prototipos a lo largo de todo el segmento entre la muestra seleccionada y su vecino, mientras que para los de la clase contraria lo realiza sobre la mitad más cercana a la muestra del segmento generado entre la misma y el vecino más cercano de la clase contraria. Los estudios realizados sobre estos dos variaciones demostraron que el sobremuestrear los bordes obtenían un mejor rendimiento en la clasificación que con el SMOTE original y otros métodos de sobremuestreo aleatorios.

Algoritmo 1 Algoritmo que obtiene el borde de la clase minoritaria.

Entrada:

Conjunto total $T = \{t_1, \ldots, t_(pnum + nnum)\}$
Conjunto clase minoritaria $P = \{p_1, \ldots, p_pnum\}$, donde $P \subseteq T$.
Conjunto clase mayoritaria $N = \{n_1, \ldots, n_nnum\}$, donde $N \subseteq T$.

Salida:

Conjunto $BORDE = \{\emptyset\}$ inicialmente vació, donde $BORDE \subseteq P$

1: **para** $i = 0$ hasta p_pnum **hacer**
2: Calcular los k vecinos más cercanos de p_i sobre el conjunto T.
3: **si** Todos los k vecinos pertenecen a N **entonces**
4: Se considera que la muestra es ruido y no se añade al conjunto.
5: **si no**, **si** Más de la mitad de los k vecinos pertenecen a P **entonces**
6: Se considera que la muestra no forma parte del borde entre ambas clases y no se añade al conjunto.
7: **si no**
8: La muestra se considera como parte del borde y se añade al conjunto $BORDE$.
9: **fin si**
10: **fin para**

2.2.3. Cadena media a dos cadenas

Uno de los principales problemas con los que nos encontramos a la hora de trabajar con datos de tipo estructurado, es que es complejo entender el espacio en el que trabajan. En el caso del proyecto uno de los principales problemas que se deben afrontar es la generación de cadenas intermedias, entre dos cadenas seleccionadas. En datos de tipo estadístico es sencillo obtener un vector central a dos vectores, cosa que con grafos, árboles o cadenas no sucede. Centrándonos en abordar el proyecto con datos estructurados del tipo cadenas existe un algoritmo eficiente que permite obtener la cadena media a dos cadenas, el cual fue desarrollado por [Abreu and Rico-Juan, 2013].

Este método consiste en que a partir de dos cadenas, se realiza el cálculo de la distancia de edición entre ambas y se obtiene el conjunto de transformaciones de inserción, borrado y sustitución que se deben llevar a cabo para transformar una cadena en la otra. A partir de todas las transformaciones y la distancia entre ambas se procede a ir desplegando el árbol de combinaciones, de tal forma que la cadena que se va formando se encuentre a una distancia inferior a la distancia que separan a las dos cadenas.

Teniendo la cadena "223697" y la cadena "246985" se representa $W(e, X)$ como una inserción, $W(X, e)$ como un borrado y $W(X, Y)$ como una sustitución, las transformaciones para pasar de una cadena a la otra serán $W(2, e)$ $W(2, 2)$ $W(3, 4)$ $W(6, 6)$ $W(9, 9)$ $W(7, 8)$ $W(e, 5)$ y la distancia entre ambas sera de 4. Al ir desplegando el árbol, mostrado en la figura 2.4, nos encontramos con que es necesario ir podando ramas que no nos llevan a ninguna parte ya que pasaríamos a repetir combinaciones como ocurre con $W(7, 8)$.

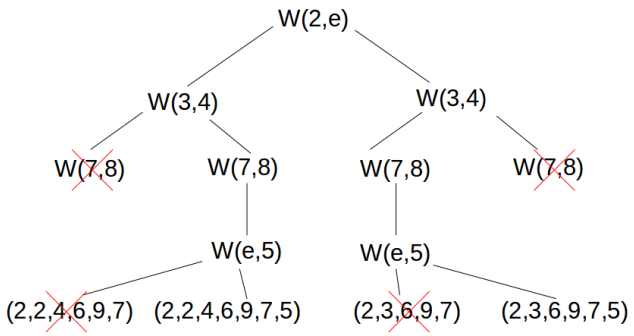


Figura 2.4.: Árbol de transformaciones para la obtención de la cadena media

Finalmente de las cadenas generadas seleccionaríamos aquellas que se sitúan a una distancia media entre las dos cadenas, en este caso las cadenas "2246975" y "236975" se encuentran a una distancia de 2 entre las dos cadenas.

3. Desarrollo

3.1. Toma de contacto

En una primera fase de toma de contacto se optó por realizar la implementación del sistema con la representación de los datos mediante vectores de características. Este proceso se llevo a cabo con la finalidad de tener una compresión mayor de los algoritmos ya existentes de los cuales nos vamos a partir para desarrollar el nuevo sistema. Para poder visualizar las pruebas que mostraron que la implementación estaba bien realizada se opto por usar el programa de representación de gráficos y funciones *GNUPlot*. Las pruebas se llevaron a cabo sobre un conjunto de datos generado de manera artificial, donde cada muestra del los datos se correspondía con un vector de dos dimensiones (X, Y) y la clase a la que este pertenecía.

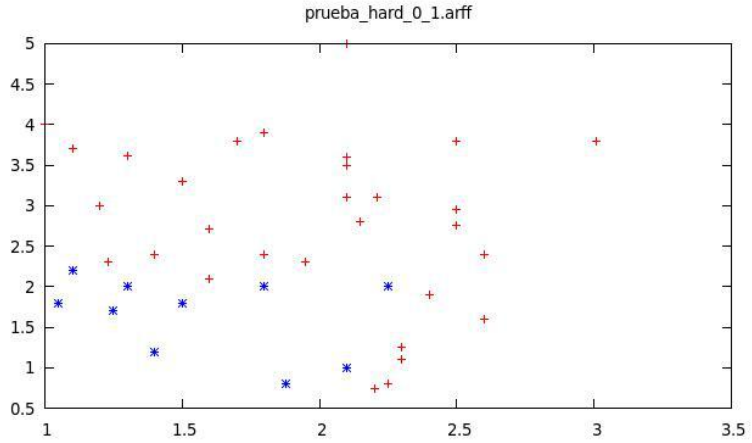


Figura 3.1.: Conjunto de datos desequilibrado sobre el que trabajaremos. En dicha imagen se muestran como '+' los prototipos pertenecientes a la clase mayoritaria y '*' los pertenecientes a la clase minoritaria.

El conjunto de datos constaba de 10 prototipos para la clase que tenía el menor número de muestras y 28 para la clase mayoritaria del conjunto de datos (Fig. 3.1).

Teniendo el conjunto de datos ya definido, se procedió a realizar la implementación del algoritmo de sobremuestreo de la clase minoritaria SMOTE (Sec. 2.2.1). Los resultados mostraban que la clase minoritaria sufre una generación de 18 muestras sintéticas consiguiendo que esta se iguale a la mayoritaria (Fig. 3.2). En este algoritmo se puede apreciar como las muestras que se producen se encuentran cercadas en su propia región.

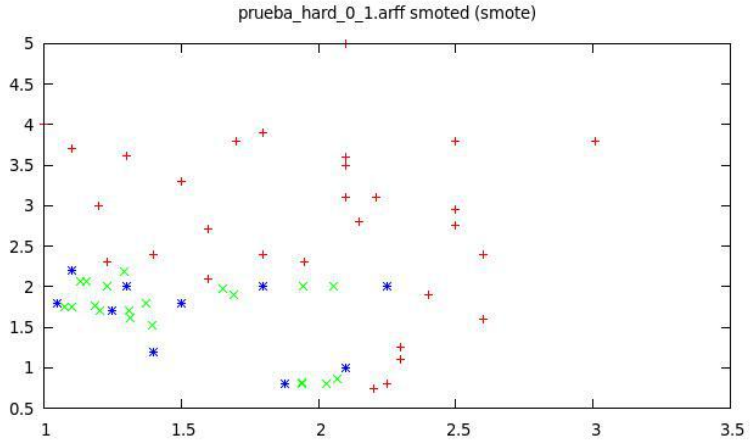


Figura 3.2.: Conjunto de datos resultante de ejecutar SMOTE original para solucionar el desequilibrio. En dicha imagen se muestran como '+' los prototipos pertenecientes a la clase mayoritaria, '*' los pertenecientes a la clase minoritaria y 'x' como las muestras sintéticas generadas.

El siguiente algoritmo del cual se realizo la implementación fue el BorderLine-SMOTE1 (Sec. 2.2.2). Al igual que el algoritmo anterior este también produce el número de muestras necesarias como para igualar ambas clases, con la única diferencia que este trabaja sobre el borde de la clase minoritaria. Tras detectar el borde de todo el conjunto la generación se realiza sobre el mismo (Fig. 3.3).

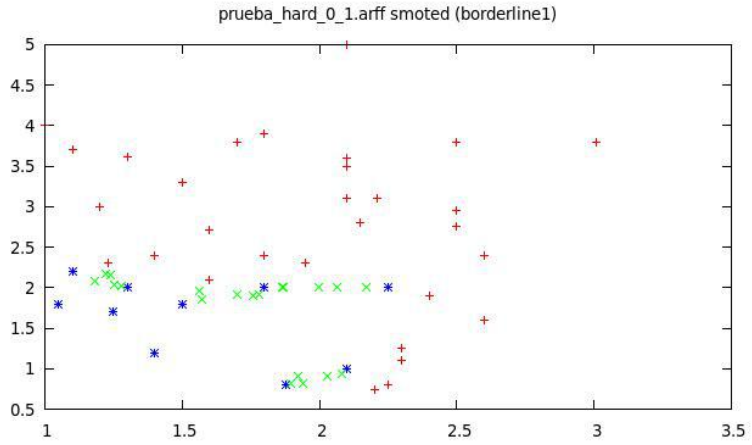


Figura 3.3.: Conjunto de datos resultante de ejecutar BorderLine - SMOTE1 para solucionar el desequilibrio. En dicha imagen se muestran como '+' los prototipos pertenecientes a la clase mayoritaria, '*' los pertenecientes a la clase minoritaria y 'x' como las muestras sintéticas generadas.

Y por último, el algoritmo BorderLine-SMOTE2 (Sec. 2.2.2) es la otra variante del SMOTE original y este como el anterior tambien lleva a cabo el sobre muestreo del borde, con la diferencia de que en este caso tambien se tiene en cuenta la clase mayoritaria para la generación de muestras. Esto ocasiona que las muestras se aproximen más, que en el caso anterior, a la clase de mayor número de prototipos (Fig. 3.4).

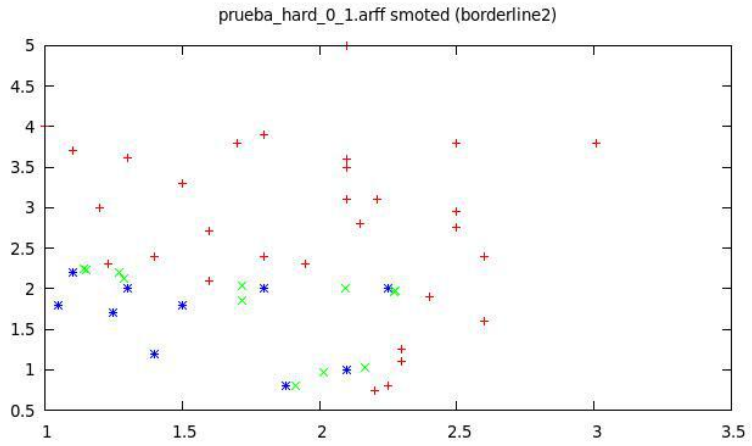


Figura 3.4.: Conjunto de datos resultante de ejecutar BorderLine - SMOTE2 para solucionar el desequilibrio. En dicha imagen se muestran como '+' los prototipos pertenecientes a la clase mayoritaria, '*' los pertenecientes a la clase minoritaria y 'x' como las muestras sintéticas generadas.

3.2. Preparación de datos

Para poder llevar a cabo las pruebas de forma correctamente, primeramente fue necesario llevar a cabo una fase de preparación de datos, la cual consistía en transformar el contenido de las bases de datos con las que vamos a trabajas, así como prepararlas para posteriormente ejecutarlas con el algoritmo de validación cruzada (Sec 2.1.5). Las bases de datos con las cuales realizaremos las pruebas son las siguientes:

- *National Institute of Standards and Technology* (NIST), es una colección de imágenes de caracteres escritos a mano. Los datos que alberga son un total de 800.000 imágenes de caracteres escritos a mano por un total de 3.600 escritores. Esta compuesta por 26 clases donde cada clase es una letra del abecedario.

- *Modified National Institute of Standards and Technology* (MNIST), formada por una colección de imágenes de dígitos escritos a mano. Se encuentra dividida en dos partes, ambas con 10 tipos de datos donde cada uno corresponde a un dígito del 0 al 9. Tendremos por un lado un conjunto de entrenamiento compuesto por 60.000 imágenes, el cual descartaremos para las pruebas debido a su gran multitud de datos, y por el otro un conjunto de test compuesto por 10.000 imágenes que será el que empleemos para las pruebas.

- *U.S. Postal Service* (USPS), esta compuesta por datos numéricos obtenidos a partir de la digitalización de dígitos escritos a mano en los sobres del Servicio Postal de Estados Unidos. Los dígitos originales escaneados son binarios, de diferentes tamaños y orientaciones, por lo que tras realizar un normalizado de las imágenes lo que resulta son imágenes de 16 x 16 en escala de grises. Esta formada por un total de 9298 dígitos escritos a mano, los cuales se dividen en un conjunto de entrenamiento compuesto por 7291 imágenes y por un conjunto de test compuesto por 2007 imágenes.

Para poder emplear estas bases de datos fue necesaria realizar la conversión de la información de las imágenes a un espacio de datos estructurado. Para ello se empleo la técnica del código de Freeman (Sec. 2.1.1) a partir del cual conseguimos codificar el contorno de las imágenes a través de un espacio de cadenas. Las bases de datos quedaron con el tamaño y la información que podemos ver en la tabla 3.1.

Una vez teníamos las bases de datos preparadas para poder emplearlas en un espacio de cadenas, nos encontramos con que el algoritmo SMOTE (Sec. 2.2.1) y sus dos variantes (Sec. 2.2.2) emplean tan solo dos clases para resolver el problema del desequilibrio de clase. Por ese mismo motivo se optó por realizar bases de datos a partir de las indicadas en la tabla 3.1 pero cogiendo únicamente 2 clases para cada base de datos. Para poder realizar la selección de estas dos clases que formarían las nuevas base de datos, se realizó

Base de datos	Tamaño	Número de clases	Descripción
NIST	5.200	26	Colección de imágenes con caracteres del abecedario manuscritos.
MNIST	10.000	10	Colección de imágenes con dígitos escritos a mano.
USPS	9.298	10	Colección de imágenes con dígitos manuscritos obtenidos de los sobres del servicio postal.

Tabla 3.1.: Resumen de las bases de datos que utilizaremos

un sondeo previo que consistía en ver entre que clases existía una mayor confusión, o lo que es lo mismo, para que clases sus datos estaban lo suficientemente solapados como para originar fallos en la clasificación de una forma más sencilla. A través de los resultados del sondeo realizados se optó por generar bases en las cuales los datos estuvieran muy solapados y bases de datos donde los datos fuesen más sencillos de clasificar, o lo que es lo mismo unos datos poco solapados. Teniendo en cuenta los resultados se generaron dos bases de datos muy solapadas y dos más poco solapadas, con la finalidad de observar como se comportaban los datos en cada caso.

En los gráficos mostrados (Gráficos 3.5, 3.6 y 3.7) tendremos que en valor de las ocurrencias se indica lo siguiente, dado un par de clases $X - Y$ donde pretendemos clasificar la muestra X, el número de ocurrencias refleja las veces en las que el algoritmo de clasificación $1 - NN$ clasifica la muestra X como la muestra Y. Con esto extraemos de información que cantidad de veces ambas clases se confunden y por lo tanto podremos concluir que con un valor alto en el número de ocurrencias obtenemos que ambas clases son difíciles de distinguir y por lo tanto deducir que su información esta solapada. Para cada base de datos que se detalla a continuación tendremos que los conjuntos de datos generados como muy solapados o poco solapados fueron los siguientes:

* NIST (Gráfico. 3.5): se generaron dos bases de datos muy solapadas, en este caso las pertenecientes a las clases C-G y V-Y, y dos más poco solapadas, en este caso I-Q y I-U.

* MNIST (Gráfico. 3.6): se generaron dos bases de datos muy solapadas, en este caso las pertenecientes a las clases 3-5 y 7-9, y dos más poco solapadas, en este caso 1-5 y 3-6.

* USPS (Gráfico. 3.7): se generaron dos bases de datos muy solapadas, en este caso las pertenecientes a las clases 3-5 y 7-9, y dos más poco solapadas, en este caso 1-2 y 3-4.

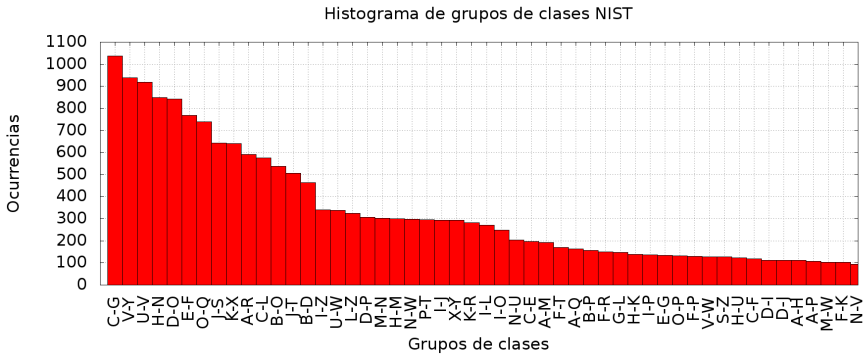


Figura 3.5.: Gráfica de sondeo NIST

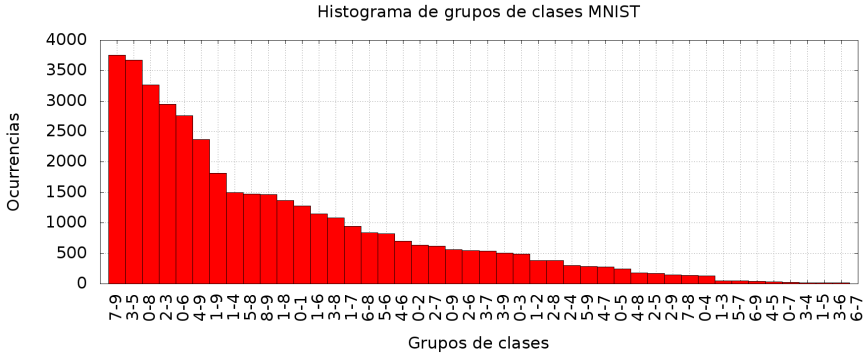


Figura 3.6.: Gráfica de sondeo MNIST

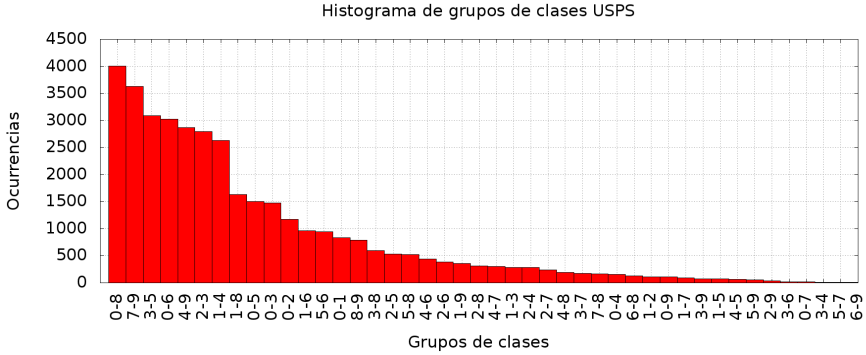


Figura 3.7.: Gráfica de sondeo USPS

Tras generar las bases de datos para 2 clases se continuó con la preparación de estos conjuntos de datos para su posterior ejecución en el algoritmo de validación cruzada. En

este caso se prepararon los datos para realizar un 10 *Fold Cross-Validation*. Para ello se debe coger todo el conjunto total de los datos y hacer la división en 10 particiones, por lo que a la hora de generar las pruebas, en dicho algoritmo se tendrá que por cada iteración un 90 % de los datos formará el conjunto de entrenamiento del clasificador y el otro 10 % formara el conjunto de test.

Por último, nos encontramos con que los datos se encuentran equilibrados en este momento por lo que deberemos generar un desequilibrio artificial para poder observar como ha afectado el desequilibrio a esa base de datos y tras este paso llamar al nuevo sistema que me vuelva a equilibrar los datos y así poder observar si conseguimos ganar en la clasificación. El desequilibrio que se generó fue a través de una reducción del 20 %, 40 %, 60 % y 80 % sobre una de las dos clases de cada base de datos, se pueden ver los resultados del contenido de las bases de datos tras estos procesos en la tabla 3.2. Los directorios de trabajo por base de datos son los representados en la figura 3.8.

BBDD	**Instancias por clase (antes de la reducción)**	**Instancias (tras reducción)**			
		20 %	40 %	60 %	80 %
NIST	520	416	312	208	104
MNIST	1000	800	600	400	200
USPS	928	742	556	372	186

Tabla 3.2.: La tabla muestra la cantidad de muestras por clase que existen en las bases de datos originales y la cantidad tras aplicarles la reducción indicada en % sobre una de las dos clases.

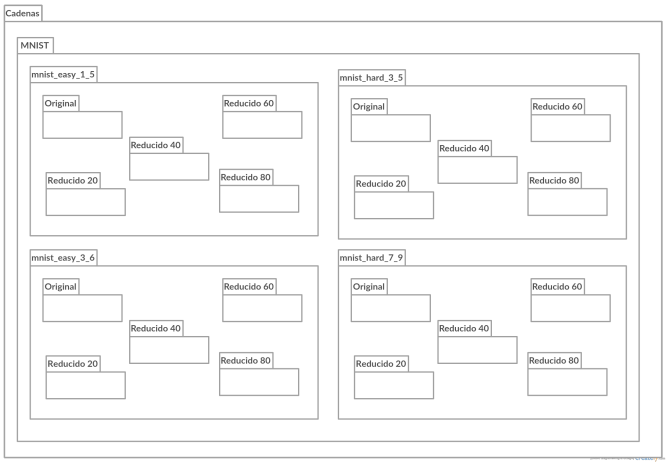


Figura 3.8.: Directorio de trabajo se nos queda para cada base de datos que se va a lanzar en el sistema.

3.3. Diseño del sistema

Una vez tenemos los datos del sistema preparados, vamos a proceder a indicar cual sera su implementación. Tras estudiar todo el conjunto de algoritmos explicado en los apartado anteriores podemos detallar el conjunto de métodos que serán necesarios para nuestro sistema.

- Cálculo de distancias.
  - Algoritmo de la distancia de edición o distancia de Levenshtein.
- Algoritmo de clasificación.
  - Algoritmo kNN.
- Algoritmo de generación de cadenas.
  - Algoritmo de cálculo de cadenas intermedias.
- Algoritmos de sobremuestreo de la clase minoritaria.
  - Algoritmo SMOTE.
  - Algoritmo BorderLine-SMOTE1 (B1).
  - Algoritmo BorderLine-SMOTE2 (B2).
- Algoritmo de validación cruzada.
  - Algoritmo *k-fold cross-validation*.
- Medida para la obtención de resultados.
  - Algoritmo de la medida de exactitud F1.

Primeramente realizaremos la implementación de un algoritmo que nos permita obtener el cálculo de distancias entre dos cadenas que se quieran comparar. Este algoritmo será necesario ya que a la hora de implementar el método de clasificación kNN deberemos tener una forma de poder comparar una muestra a clasificar con el resto del conjunto de los datos suministrados al sistema.

Una vez tenemos las técnicas que nos permiten trabajar con los datos a nivel de cla-

sificación, nos centraremos en la implementación de los métodos para la generación de muestras sintéticas en espacio de datos estructurado del tipo cadenas. Para ello deberemos elaborar primeramente una técnica que nos permita a través de dos muestras del sistema generar nuevos prototipos, por lo que basándonos en el algoritmo explicado en la sección 2.2.3 el cual nos permitía a partir de dos cadenas generar una intermedia, nosotros realizaremos la implementación de un método que permita generar cadenas a lo largo del segmento que une ambas muestras.

A continuación se pasara con la implementación de las técnicas de sobremuestreo de la clase minoritaria. En este caso nos encontramos con los tres algoritmos de generación de SMOTE, por lo que deberemos preparar el sistema para que de manera sencilla se pueda decidir entre sobremuestrear el borde de la clase únicamente o la clase entera y por otro lado que se pueda decidir si sobremuestrear teniendo en cuenta la clase minoritaria, caso del SMOTE original y BorderLine-SMOTE1, o la clase con mayor número de muestras, caso que ocurre con BorderLine-SMOTE2.

Ya para finalizar, se implementara la técnica de *k-fold cross-validation* y una método de medida de las pruebas, en este caso la medida de exactitud F1 (Sec. 2.1.4) los cuales permitirán realizar las pruebas sobre el sistema y poder con posterioridad indicar si el sistema ofrece ventajas o por el contrario no aporta beneficios.

La implementación del sistema se decidió por que fuera en lenguaje Java, ya que para este lenguaje teníamos una mayor número de herramientas de desarrollo que facilitaban el proceso.

3.4. Implementación

En este apartado de la memoria nos centraremos en detallar la implementación llevada a cabo de todos los algoritmos del sistema mencionados en el apartado anterior. Para cada algoritmo detallaremos conceptos de implementación, variaciones con respecto a los originales y problemas con los cuales nos hemos encontrado a la hora de realizar su implementación.

3.4.1. Distancia de edición

Como ya se indicaba en la sección 2.1.2 este algoritmo hace uso de la programación dinámica para ir registrando el coste de cada operación que realiza. Para llevar a cabo la implementación de la distancia de edición se siguió el proceso indicado en el algoritmo 2. En este código la parte más importante seria la de decir que operación registrar en la

matriz por ello nos quedamos con el valor mínimo de las tres posibles operaciones y esa es la que registramos.

Algoritmo 2 Algoritmo del cálculo de la distancia de edición.

Entrada:
Cadena de entrada $s1[tam_1]$
Cadena de entrada $s2[tam_2]$
Salida:
Distancia entre ambas cadenas d

1: Se define una matriz de $M[tam_1 + 1][tam_2 + 1]$ empleada para el almacenamiento del coste de las transformaciones.

2: **para** $i = 0$ hasta $tam_1 + 1$ **hacer**
3: Hacer $M[i][0] = i$ para inicializar la primera fila.
4: **fin para**

5: **para** $j = 0$ hasta $tam_2 + 1$ **hacer**
6: Hacer $M[0][j] = j$ para inicializar la primera columna.
7: **fin para**

8: **para** $i = 1$ hasta $tam_1 + 1$ **hacer**
9: **para** $j = 1$ hasta $tam_2 + 1$ **hacer**
10: $eliminacion = M[i-1][j] + 1$
11: $insercion = M[i][j-1] + 1$
12: $sustitucion = M[i-1][j-1]$
13: **si** $s1[i-1] == s2[j-1]$ **entonces**
14: La sustitución tendría un coste $coste = 0$
15: **si no**
16: La sustitución tendría un coste $coste = 1$
17: **fin si**
18: $M[i][j] = minimo(eliminacion, insercion, sustitucion + coste)$
19: **fin para**
20: **fin para**
21: Devuelve $d = M[tam_1][tam_2]$

Una de las variantes que tocó aplicar a este proceso fue el realizar el ascenso desde la última posición de la matriz, que se crea para registrar las transformaciones, con el fin de obtener el conjunto de transformaciones $W(X, Y)$ mencionada en apartados anteriores. Para ello tendremos que partir de la posición tam_1 y la posición tam_2 de la matriz que registro las transformaciones, procederemos a evaluar sus casillas vecinas las cuales hacen referencia a el coste de una inserción, borrado o sustitución. Para la sustitución se deberá

comprobar además si tuvo coste el realizarla, para ello se comprobaran los caracteres de las cadenas situados en las posiciones $tam_1 - 1$ y $tam_2 - 1$. Una vez tenemos los valores del coste de las operaciones anteriores para llegar a la casilla actual en la que nos encontramos evaluaremos que si el coste de insertar es menor que el de borrar y el de insertar + 1 es ¡= que el de sustituir, la operación anterior fue una inserción, si el borrado es menor que la sustitución fue un borrado y si no es ninguna de ambas quiere decir que la operación previa fue la sustitución (Ver alg. 3).

Algoritmo 3 Algoritmo que obtiene el conjunto de transformaciones.

Entrada:

Cadena de entrada $s1[tam_1]$
Cadena de entrada $s2[tam_2]$
Matriz rellenada del algoritmo 2: $M[tam_1 + 1][tam_2 + 1]$

1: **mientras** $(tam_1 > 0)\&\&(tam_2 > 0)$ **hacer**
2: $insertar = M[tam_1][tam_2 - 1]$
3: $borrar = M[tam_1 - 1][tam_2]$
4: $sustituir = [tam_1 - 1][tam_2 - 1]$

5: **si** $s1[tam_1 - 1] == s2[tam_2 - 1]$ **entonces**
6: La sustitución valió $coste = 0$
7: **si no**
8: La sustitución valió $coste = 1$
9: **fin si**

10: $sustitucion+ = coste$

11: **si** $(insertar <= borrar)\&\&(insertar + 1 <= sustituir)$ **entonces**
12: Registro $movimientoAnterior.INSERCCION$
13: $tam_2 = tam_2 - 1$
14: **si no**, **si** $borrar < sustituir$ **entonces**
15: Registro $movimientoAnterior.BORRAR$
16: $tam_1 = tam_1 - 1$
17: **si no**
18: Registro $movimientoAnterior.SUSTITUIR$
19: $tam_1 = tam_1 - 1$
20: $tam_2 = tam_2 - 1$
21: **fin si**
22: **fin mientras**

Con este conjunto sera con el que luego realizaremos el cálculo de la generación de cadenas intermedias entre dos muestras del conjunto de la base de datos, tal y como se indicaba en la sección 2.2.3.

3.4.2. kNN

Una vez construido los algoritmos del cálculo de la distancia de edición, procedemos con la implementación de kNN.

```
1  Etiqueta knn(int k, List<Muestra> T, Muestra S) //T = Conjunto de
       entrenamiento | S = Prototipo a clasificar
2  {
3      List<Distancia> O = new ArrayList(); //O = conjunto de distancias a S.
4      Histograma H = new Histograma(); //H = cantidad de vecinos por etiqueta
5      Etiqueta et; //et = resultado de la clasificación de S.
6      for(Muestra p : T)
7      {
8          Distancia d = p.getDistancia(S); //Distancia entre dos prototipos.
9          O.add(d); // Guarda el resultado de la distancia en una lista.
10     }
11     Collections.sort(O); //Ordena las distancias de menor a mayor.
12
13     for(int i = 0; i<k ; ++i)
14     {
15         H.add(O.get(i).etiqueta); //Contabiliza el número de etiquetas
               repetidas en los 'k' vecinos más cercanos.
16     }
17
18     et = H.masRepetido();
19     return et;
20 }
```

Sea k el número de vecinos más cercanos a tener en cuenta en kNN, T el conjunto de entrenamiento y S el prototipo que necesitamos etiquetar, el algoritmo deberá devolver el tipo de dato al que pertenece S.

En la implementación, hacemos uso de la clase *Distancia* que representa la distancia entre dos prototipos y que a su vez contiene la etiqueta asociada a la muestra perteneciente al conjunto de entrenamiento. También disponemos de la clase *Histograma*, que implementa un mapa hash que contabiliza el número de veces que se añade al mismo un elemento. La clase *Etiqueta* representará un tipo de dato disponible en la base de datos, y lo utilizaremos para obtener el resultado de la clasificación.

kNN debe recorrer T, obteniendo la distancia entre los prototipos que contiene y la muestra que debemos clasificar S. Esta distancia, así como la etiqueta del ejemplo perteneciente a T, se incluyen en un objeto *Distancia* que se almacenará en una lista. Al final, tendremos una lista de objetos de tipo *Distancia* que ordenaremos de menor a mayor según el valor de la distancia. A continuación añadimos en el histograma los primeros k elementos de la lista ordenada, y devolvemos la etiqueta que más se ha repetido en este subconjunto de datos como resultado de la clasificación de S.

3.4.3. Algoritmo de generación de cadenas intermedias

Para poder emplear los algoritmos de sobremuestreado era necesaria la implementación de un método de generación de prototipos de datos estructurados del tipo cadena. La generación de muestras sintéticas que tuvimos que desarrollar fue basandonos en el algoritmo comentado en la sección 2.2.3. En dicho algoritmo siempre se generaba la cadena media a dos cadenas, cosa que necesitábamos cambiar para poder generarla a lo largo del segmento que unen las dos muestras. A continuación se detalla el algoritmo que se implemento.

```
1  public Sample populate(Sample sample, Sample neighbor, String[]
       transformations, double distanceSN)
2  {
3     Random r = new Random();
4     double cont1 = 0, cont2 = 0;
5
6     Sample sinthetic = new Sample();
7     sinthetic.setClassSample(sample.getClassSample());
8
9     for(String s : transformations)
10    {
11       double random = r.nextDouble();
12       if(random < threshold)
13       {
14          if(s.charAt(1) != 'e')
15             sinthetic.insertValue(s.charAt(1));
16          if(s.charAt(1) != s.charAt(2))
17             ++cont1; //Distancia del prototipo al vecino
18       }
19       else
20       {
21          if(s.charAt(2) != 'e')
22             sinthetic.insertValue(s.charAt(2));
23          if(s.charAt(1) != s.charAt(2))
24             ++cont2; //Distancia de prototipo al ejemplo evaluado
25       }
26    }
27    return sinthetic;
28 }
```

Partiendo de dos muestras *sample* y *neighbor*, junto con las transformaciones para pasar de una cadena a otra y la distancia de edición entre ambas, tendremos que se definirán dos contadores que nos llevaran el conteo de distancias de la muestra sintética que estamos generando a los dos prototipos enviados al método. Por otro lado tendremos un valor denominado *threshold* y un número aleatorio r el cual se encuentra comprendido entre 0 y 1, a partir de los cuales iremos formando la cadena que se aproximara más a la muestra *sample* o a la *neighbor*. En este algoritmo el valor *threshold* se emplea porque hay que definir probabilidades de aproximarse más a una muestra o otra dependiendo

de los algoritmos de SMOTE, en el caso de SMOTE original y BorderLine-SMOTE1 tendremos que dicho valor es de $0,5$ con lo que conseguiremos que la cadena que se genere este a lo largo de todo el segmento que une las dos muestras y por otro lado tendremos que para BorderLine-SMOTE2 el valor sera $0,75$ para que la muestra que se genere se encuentre en lo que vendría a ser la mitad más próxima del segmento a la muestra *sample*.

El proceso es el siguiente, teniendo representadas las transformaciones como $W(X, Y)$, por cada transformación s que se deba realizar entre ambas cadenas, generaremos el número aleatorio comprendido en el rango $[0, 1]$ y tendremos que si dicho valor es menor que el valor de *threshold* comprobaremos que si el valor de X es distinto de la e, que es el carácter que añadíamos para representar la inserción y el borrado, este se añade a la cadena sintética y anotaríamos que la muestra sintética se aleja un punto mas del *neighbor* ya que el carácter recogido pertenece a la cadena del *sample*. Por el contrario tendremos que si es mayor que el *threshold* observaremos el valor de la Y viendo si este es distinto de e, y si es así se añadirá a la cadena sintética y tendremos que incrementar en uno la distancia de la muestra hacia el *sample* ya que el carácter introducido pertenece al *neighbor*. Este proceso se realizara mientras queden transformaciones y finalmente se devolverá el valor de la nueva cadena generada artificialmente. Los valores de los contadores se emplean con posterioridad para confirmar que la cadena generada entre ambas muestras se encuentra situada en el segmento del medio entre ambas. La comprobación que se realiza es que si la suma de ambos contadores es igual a la distancia de edición entre las dos cadenas, la cadena generada es correcta, si no es así se vuelve a ejecutar el algoritmo.

3.4.4. Algoritmos de sobremuestreo SMOTE, B1 y B2

Tras la implementación de algoritmos de generación de muestras, en este caso cadenas nos centramos en el desarrollo de los algoritmos de SMOTE. En este caso nos encontramos con un problema que surgía por trabajar con datos estructurados del tipo cadenas, este problema consiste en que a partir de dos muestras de cadenas no se pueden generar infinitas cadenas intermedias, ya que el número de combinaciones posibles entre las diferentes transformaciones para pasar de una cadena a otra se encuentra limitado. En el SMOTE original no se encuentran con dicho problema debido a que ellos trabajan con datos del tipo vectores de características, y las posibilidades de generar muestras intermedias entre dos vectores si es infinita.

Con este problema se tuvo que lidiar en el algoritmo 4 el cual controla que no se llame más veces al método *populate* (Sec. 3.4.3) cuando no existan más combinaciones posibles. Tanto para el SMOTE original como para sus dos variantes B1 y B2 se empleó el método detallado en el algoritmo 4, con la diferencia en las veces en que se ejecutaba y los parámetros de entrada que este recibía.

* Algoritmo SMOTE Original: En este los parámetros que se le envían al sistema son como conjunto de datos a sobremuestrear el conjunto completo de la clase minoritaria, como conjunto sobre el que hacer el kNN también se envía la clase minoritaria completa y la ejecución del algoritmo se realiza una única vez.

* Algoritmo BorderLine - SMOTE1: Aquí se manda al método únicamente el borde de la clase minoritaria como conjunto a sobremuestrear, la clase minoritaria al completo para realizar el algoritmo de kNN y también con una única ejecución.

* Algoritmo BorderLine - SMOTE2: En este se llevan a cabo dos llamadas a este método, donde en la primera se envian el borde de la clase minoritaria como conjunto a sobremuestrear y la clase minoritaria al completo para realizar el algoritmo de kNN. Con esto sobremuestreamos el 50 % y la otra mitad se haría en la segunda ejecución mandando el borde de la clase minoritaria como conjunto a sobremuestrear y la clase mayoritaria al completo para realizar el algoritmo de kNN. En este algoritmo de los 3 es donde interviene principalmente la variable *threshold* del método *populate*, siendo de un valor de $0,5$ en la primera fase y de un valor de $0,75$ en la segunda fase para conseguir que la muestra sintética se aproxime más al *sample* propio de la clase minoritaria y no al vecino más cercano de la clase mayoritaria.

Algoritmo 4 Algoritmo para el sobremuestreo de cadenas.

Entrada:

Conjunto de datos a sobremuestrear $M = \{m_1, \ldots, m_n\}$
Conjunto de datos sobre los que realizar el kNN $P = \{p_1, \ldots, p_n\}$
Número de prototipos a los que llegar para el con seguir equilibrado N
Número de vecinos que se tendrán en cuenta para el cálculo del kNN k

Salida:

Conjunto de datos generados artificialmente $A = \{a_1, \ldots, a_n\}$

1: Se calcula el número de muestras sintéticas que se deberán generar por muestra en el conjunto M.
2: $NP = N/tam(M)$

3: **para todo** m_i en M **hacer**
4: Obtener los k vecinos más cercanos a m_i pertenecientes al conjunto P.
5: **para todo** vecino p_j **hacer**
6: Llamar al algoritmo que genera muestras sintéticas NP veces.
7: $a = populate(m_i, p_j, transformaciones, distanciaEd)$
8: **si** $(distAlSample + distAlNeighbor == distEdicion)$ **entonces**
9: Añado el sample a al conjunto A
10: **si no**
11: **mientras** Existan combinaciones ó Cadena generada no sea correcta **hacer**
12: Llamar de nuevo al método *populate* hasta conseguir una cadena válida
13: **fin mientras**
14: **fin si**
15: **fin para**
16: **fin para**
17: Devuelve el conjunto A relleno de muestras sintéticas.

3.4.5. Validación cruzada y Medida de exactitud F1

Para el algoritmo de validación cruzada *k-fold cross-validation* hemos empleado un k de 10, o lo que es lo mismo, tendremos 10 particiones de la base de datos para ejecutar las pruebas. Como ya se menciona en apartados anteriores (Sec. 3.2) tendremos las 10 particiones y con ellas realizaremos 10 ejecuciones del sistema donde para cada ejecución tendremos que una partición formara el *test*, por lo que seria el 10 % de pruebas, y el resto formaría el *training*, lo que vendría a ser el 90 % de entrenamiento del sistema.

Para llevar a cabo todo el registro de valores que luego nos permitieran comparar los resultados de las pruebas se optó por implementar la medida de exactitud F1 (Sec. 2.1.4). En esta medida se tienen en cuenta los valores de precisión y exhaustividad, por

ello fue necesario realizar la implementación de una clase Java.

```
1  public class F1
2  {
3    private int truePositive; //Valores que el clasificador dice que son de
         la clase A cuando realmente lo son.
4    private int trueNegative; //Valores que el clasificador dice que son de
         la clase B cuando realmente lo son.
5    private int falsePositive; //Valores que el clasificador dice que son de
         la clase A cuando realmente no lo son.
6    private int falseNegative; //Valores que el clasificador dice que son de
         la clase B cuando realmente no lo son.
7    private double recall; //Valor de la exhaustividad
8    private double accuracy; //Valor de la precisión
9    private double f1; //Valor de la medida F1
10
11   public void calcula_recall()
12   {
13     accuracy = ((truePositive)/(double)(truePositive+falsePositive));
14   }
15
16   public void calcula_accuracy()
17   {
18     recall = ((truePositive)/(double)(truePositive+falseNegative));
19   }
20
21   public void calculateF1()
22   {
23     f1 = 2*((accuracy*recall)/(accuracy+recall));
24   }
25 }
```

A medida que se van ejecutando las iteraciones del *10-fold cross-validation* se van registrando los datos en un objeto de la clase F1 para posteriormente llevar la información de los resultados a un fichero.

4. Experimentación

En este apartado mencionaremos los datos que se obtuvieron tras realizar el conjunto de pruebas detallado en los apartados anteriores. A continuación se explicarán los resultados de las tablas y detallaremos que aportan realmente.

4.1. Resultados sobre bases de datos poco solapadas

Tras realizar todo el conjunto de las pruebas para bases de datos en las que la información se encontraba poco solapada, obtuvimos como resultados los datos reflejados en la tabla 4.1. En esta tabla nos encontramos con los resultados en promedio de todas las base de datos citadas en los apartados anteriores (Ver tablas completas en el anexo A). Tras analizar los resultados podemos observar como el nuevo sistema desarrollado consigue de media igualar al Base son el algoritmo de SMOTE cosa que no aportaría ninguna ventaja, para B1 conseguimos mejora en algún resultado pero tampoco existe una diferencia significativa entre ambos y por último para el algoritmo B2 ocurre lo mismo que con el B1.

Con esto podemos extraer de conclusiones que para bases de datos donde los conjuntos de datos estén poco solapados, no siempre aportaremos ventajas en la clasificación. Este análisis se corrobora al observar los resultados obtenidos por los test de Wilcoxon (ver tabla 4.2). Ahí se demuestra que el algoritmo no aporta ninguna ventaja empleando el algoritmo de SMOTE original. En el caso de sobremuestrear el borde con B1 y B2 se consiguen algunas ventajas para el desequilibrio del 40 % y el 60 %, esto puede ser debido a que generar más muestras sobre el borde ayude todavía más en la clasificación.

Datos poco solapados promediados

Reducción	k	Original	Base	SMOTE	B1	B2
20	1	**99,7**	**99,5**	**99,5**	99,5	**99,6**
	3	**99,7**	**99,5**	**99,5**	99,5	99,5
	5	99,6	**99,5**	**99,5**	**99,6**	**99,6**
	7	99,5	99,4	99,4	99,4	99,4
40	1	**99,7**	**99,5**	**99,5**	**99,5**	**99,5**
	3	**99,7**	99,2	99,3	99,4	99,3
	5	99,6	99,2	99,3	99,3	99,3
	7	99,5	99,2	99,2	99,2	99,3
60	1	**99,7**	**99,4**	**99,4**	**99,4**	**99,3**
	3	**99,7**	99,1	99,1	99,2	99,2
	5	99,6	98,9	99,2	99,0	99,0
	7	99,5	98,9	99,1	99,0	99,0
80	1	**99,7**	**98,4**	**98,4**	**98,5**	**98,5**
	3	**99,7**	97,8	98,3	97,8	97,8
	5	99,6	97,9	98,2	98,0	98,0
	7	99,5	97,6	98,2	97,7	97,7

Tabla 4.1.: La columna "Original" muestra los resultados sin aplicar desequilibrios, "Base" que muestra los resultados tras aplicar el desequilibrio indicado sobre una clase, "SMOTE" muestra resultados tras pasar por el algoritmo SMOTE original, "B1" resultados del algoritmo BorderLine-SMOTE1 y "B2" los resultados del algoritmo BorderLine-SMOTE2. Los resultados reflejados están medidos con la medida de exactitud F1 y expresados en %. Los valores resaltados muestran los mejores resultados obtenidos para la F1 por cada situación de desequilibrio inicial.

Test de Wilcoxon (poco solapado)

Reducción	SMOTE	B1	B2
20	=	=	=
40	=	✓	✓
60	=	✓	=
80	=	=	=

Tabla 4.2.: Resultados de las pruebas de suma de rangos de Wilcoxon para las bases de datos poco solapadas. El símbolo ✓ indica que se puede asegurar con una confianza del 95 % que el algoritmo empleado para sobremustrear los datos indicado en la columna consigue mejoras de clasificación sobre los resultados "Base" obtenidos de la tabla 4.1. El símbolo = indica que no es posible afirmar que el algoritmo empleado para el sobremuestreo de los datos sea ni mejor ni peor con una confianza del 95 %.

4.2. Resultados sobre bases de datos muy solapadas

Una vez realizado el estudio sobre bases de datos poco solapadas nos pondremos a detallar los resultados obtenidos para aquellas bases de datos que si que son muy solapadas.

Tras realizar todo el conjunto de las pruebas para bases de datos en las que la información se encontraba muy solapada, obtuvimos como resultados los datos reflejados en la tabla 4.4. A si mismo, en esta tabla también nos encontramos con los resultados en promedio de todas las base de datos citadas en los apartados anteriores (ver tablas completas en el anexo A). Tras analizar los resultados en este caso si podemos observar como el nuevo sistema desarrollado consigue de media superar al Base en los tres algoritmos de sobremuestreado de datos y en muchas ocasiones por más de 2 puntos de diferencia.

Con esto podemos concluir con que el sistema se comporta de manera favorable ante bases de datos donde los conjuntos de datos se encuentran muy solapados. Con el fin de corroborar esta conclusión también se aplicó el test de Wilcoxon (ver tabla 4.3) sobre los resultados.

Test de Wilcoxon (muy solapado)

Reducción	SMOTE	B1	B2
20	=	✓	=
40	✓	✓	✓
60	✓	=	=
80	✓	=	=

Tabla 4.3.: Resultados de las pruebas de suma de rangos de Wilcoxon para las bases de datos muy solapadas. El símbolo ✓ indica que se puede asegurar con una confianza del 95 % que el algoritmo empleado para sobremustrear los datos indicado en la columna consigue mejoras de clasificación sobre los resultados "Base" obtenidos de la tabla 4.4. El símbolo = indica que no es posible afirmar que el algoritmo empleado para el sobremuestreo de los datos sea ni mejor ni peor con una confianza del 95 %.

Datos muy solapados promediados

Reducción	k	Original	Base	SMOTE	B1	B2
20	1	94,1	92,5	92,6	92,8	92,6
	3	**94,2**	92,7	93,2	93,6	93,1
	5	94,1	**92,8**	**93,3**	94,1	93,5
	7	94,0	92,6	93,2	**94,2**	**94,2**
40	1	94,1	90,4	91,2	91,1	90,7
	3	**94,2**	**90,8**	91,9	91,8	91,7
	5	94,1	90,1	91,8	92,1	**92,4**
	7	94,0	90,2	**92,0**	**92,5**	91,9
60	1	94,1	88,5	89,9	89,6	**89,7**
	3	**94,2**	**88,6**	90,7	90,2	89,3
	5	94,1	87,5	90,9	90,2	89,2
	7	94,0	85,7	**91,2**	**90,7**	89,4
80	1	94,1	83,1	84,8	83,3	83,0
	3	**94,2**	83,3	87,6	85,9	82,4
	5	94,1	**87,5**	**90,9**	**90,2**	**89,2**
	7	94,0	81,7	87,9	86,8	83,7

Tabla 4.4.: La columna “Original” muestra los resultados sin aplicar desequilibrios, “Base” que muestra los resultados tras aplicar el desequilibrio indicado sobre una clase, “SMOTE” muestra resultados tras pasar por el algoritmo SMOTE original, “B1” resultados del algoritmo BorderLine-SMOTE1 y “B2” los resultados del algoritmo BorderLine-SMOTE2. Los resultados reflejados están medidos con la medida de exactitud F1 y expresados en %. Los valores resaltados muestran los mejores resultados obtenidos para la F1 por cada situación de desequilibrio inicial.

5. Conclusiones y perspectivas de futuro

5.1. Conclusiones

En cuanto a conclusiones con respecto al proyecto desarrollado decir que tras realizar un estudio de una serie de técnicas que aportaban mejoras en la clasificación subsanando el desequilibrio sobre datos del tipo espacio de vectores, con este nuevo sistema se ha conseguido llevar todo este conjunto de métodos y técnicas al espacio de datos estructurado y demostrando además que también se aportan ventajas en la clasificación tras generación de cadenas sintéticas para el sobremuestreado de la clase minoritaria.

Además, se ha demostrado que con conjuntos de datos poco solapados no merece la pena emplear esta técnica ya que al estar lo suficientemente diferenciados no necesitamos de más muestras para poder clasificar correctamente. Por ello, también se puede deducir que partiendo de bases de datos equilibradas y tras realizarles un desequilibro sintético sobre ellas no somos capaces de observar una gran diferencia en los resultados de clasificación debido a que los datos no se encuentran muy solapados, con esto podemos extaer que el problema de clasificación de la base de datos no reside en el desequilibrio y que por lo tanto el aportar nuevas muestras sintéticas a partir de nuestro sistema no conllevara grandes ventajas. Por otro lado para datos que se encuentran muy solapados entre ellos hemos podido apreciar como si que esta nueva técnica aporta ventajas en la clasificación. Esto sucede justamente por el caso contrario a lo mencionado con las bases de datos de muestras poco solapadas, por lo que con ello extraemos que este método permite obtener mejoras de clasificación cuando existe un problema de desequilibrio en los datos.

Una vez dicho esto cabe detallar que se han conseguido mantener todas las ventajas de representación de la información que nos aportan los datos del tipo estructurado, a la vez que empleamos algoritmos de *Prototype generation (PG)* para realizar mejoras en la clasificación. A día de hoy existe una gran decadencia en métodos de PG que nos permitan trabajar con este tipo de datos y con esto hemos conseguido demostrar que extrapolar estas técnicas, que solo trabajan con datos del tipo vectores de características, a estos datos también pueden aportar ventajas a pesar de su alto coste de manejo.

5.2. Perspectivas de futuro

Una vez finalizado el proyecto hemos podido observar como técnicas que son poco empleadas para los datos estructurados nos han aportado grandes beneficios a la hora de trabajar con ellos.

Esto nos hace pensar en proyectos futuros a desarrollar, centrándonos en el problema del desequilibrio de datos que aborda este trabajo final de grado se podría estudiar la posibilidad de realizar el sobremuestreo de datos sobre bases de datos con más de dos clases e intentar balancear con respecto a la de mayor número de muestras para estudiar su comportamiento y con esto evitaríamos tener que limitar nuestro sistema a un único par de clases.

Por otro lado podríamos no limitar el estudio a *Prototype generation (PG)*, es decir, se podrían desarrollar nuevos métodos en los que se emplee la PG para conseguir realizar una *Prototype selection (PS)* sobre la clase mayoritaria del conjunto de los datos. Para este proceso se seleccionarían una serie de pivotes a partir de los cuales generaríamos uno nuevo sintético que los represente. Con eso conseguiríamos realizar una reducción del número de muestras en la clase con mayor número de prototipos hasta conseguir que ambas clases se equilibren. Este proceso se emplearía sobre datos del tipo estructurado cadenas para así conseguir mantener la idea principal del proyecto.

Bibliografía

[Abreu and Rico-Juan, 2013] Abreu, J. and Rico-Juan, J. R. (2013). An improved fast edit approach for two-string approximated mean computation applied to ocr. *Pattern Recognition Letters*, 34(5):496–504.

[Calvo-Zaragoza et al., 2016] Calvo-Zaragoza, J., Valero-Mas, J. J., and Rico-Juan, J. R. (2016). Prototype generation on structural data using dissimilarity space representation. *Neural Computing and Applications*, pages 1–10.

[Chawla et al., 2002] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, pages 321–357.

[Cover and Hart, 1967] Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27.

[Decaestecker, 1997] Decaestecker, C. (1997). Finding prototypes for nearest neighbour classification by means of gradient descent and deterministic annealing. *Pattern Recognition*, 30(2):281–288.

[Demšar, 2006] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30.

[Fernández and Isasi, 2004] Fernández, F. and Isasi, P. (2004). Evolutionary design of nearest prototype classifiers. *Journal of Heuristics*, 10(4):431–454.

[Freeman, 1961] Freeman, H. (1961). On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers*, (2):260–268.

[Han et al., 2005] Han, H., Wang, W.-Y., and Mao, B.-H. (2005). Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International Conference on Intelligent Computing*, pages 878–887. Springer.

[Hart, 1968] Hart, P. (1968). The condensed nearest neighbor rule (corresp.). *Information Theory, IEEE Transactions on*, 14(3):515–516.

[Refaeilzadeh et al., 2009] Refaeilzadeh, P., Tang, L., and Liu, H. (2009). Cross-validation. In *Encyclopedia of database systems*, pages 532–538. Springer.

[Rico-Juan and Iñesta, 2012] Rico-Juan, J. R. and Iñesta, J. M. (2012). New rank methods for reducing the size of the training set using the nearest neighbor rule. *Pattern Recognition Letters*, 33(5):654–660.

[Sánchez, 2004] Sánchez, J. S. (2004). High training set size reduction by space partitioning and prototype abstraction. *Pattern Recognition*, 37(7):1561–1564.

[Wilson, 1972] Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *Systems, Man and Cybernetics, IEEE Transactions on*, (3):408–421.

A. Anexo I

A.1. NIST

NIST Poco solapado (clases I - Q)

Reducción	k	Original	Base	SMOTE	B1	B2
20	1	**100,0**	**100,0**	**100,0**	**100,0**	**100,0**
	3	**100,0**	99,7	99,7	99,7	99,7
	5	99,8	**100,0**	**100,0**	**100,0**	**100,0**
	7	99,8	99,7	99,7	99,7	99,7
40	1	**100,0**	**100,0**	**100,0**	**100,0**	**100,0**
	3	**100,0**	99,6	99,6	99,6	99,6
	5	99,8	99,6	99,6	99,6	99,6
	7	99,8	99,6	99,2	99,6	99,6
60	1	**100,0**	**100,0**	**100,0**	**100,0**	**100,0**
	3	**100,0**	**100,0**	**100,0**	**100,0**	**100,0**
	5	99,8	99,2	**100,0**	99,2	99,2
	7	99,8	99,2	99,2	99,2	99,2
80	1	**100,0**	98,0	98,0	98,0	98,0
	3	**100,0**	96,0	**100,0**	96,0	96,0
	5	99,8	96,0	98,6	96,0	96,0
	7	99,8	96,0	98,6	96,0	96,0

Tabla A.1.: La columna "Original" muestra los resultados sin aplicar desequilibrios, "Base" que muestra los resultados tras aplicar el desequilibrio indicado sobre una clase, "SMOTE" muestra resultados tras pasar por el algoritmo SMOTE original, "B1" resultados del algoritmo BorderLine-SMOTE1 y "B2" los resultados del algoritmo BorderLine-SMOTE2. Los resultados reflejados están medidos con la medida de exactitud F1 y expresados en %. Los valores resaltados muestran los mejores resultados obtenidos para la F1 por cada situación de desequilibrio inicial.

NIST Poco solapado (clases I - U)

Reducción	k	Original	Base	SMOTE	B1	B2
20	1	**99,0**	**98,8**	**98,8**	**98,8**	**98,8**
	3	**99,0**	**98,8**	**98,8**	**98,8**	**98,8**
	5	**99,0**	**98,8**	**98,8**	**98,8**	**98,8**
	7	**99,0**	**98,8**	**98,8**	**98,8**	**98,8**
40	1	**99,0**	**99,2**	**98,8**	**99,2**	**99,2**
	3	**99,0**	98,4	98,4	98,4	98,4
	5	**99,0**	98,4	98,4	98,4	98,4
	7	**99,0**	98,4	98,4	98,4	98,4
60	1	**99,0**	**98,7**	**98,7**	**98,7**	**98,7**
	3	**99,0**	98,0	98,0	98,0	98,0
	5	**99,0**	98,0	98,0	98,0	98,0
	7	**99,0**	98,0	98,0	98,0	98,0
80	1	**99,0**	**95,1**	**95,1**	**95,1**	**95,1**
	3	**99,0**	93,7	93,7	93,7	93,7
	5	**99,0**	**95,1**	93,7	**95,1**	**95,1**
	7	**99,0**	93,7	93,7	93,7	93,7

Tabla A.2.: La columna “Original” muestra los resultados sin aplicar desequilibrios, “Base” que muestra los resultados tras aplicar el desequilibrio indicado sobre una clase, “SMOTE” muestra resultados tras pasar por el algoritmo SMOTE original, “B1” resultados del algoritmo BorderLine-SMOTE1 y “B2” los resultados del algoritmo BorderLine-SMOTE2. Los resultados reflejados están medidos con la medida de exactitud F1 y expresados en %. Los valores resaltados muestran los mejores resultados obtenidos para la F1 por cada situación de desequilibrio inicial.

NIST Muy solapado (clases C - G)

Reducción	k	Original	Base	SMOTE	B1	B2
20	1	**91,4**	**87,9**	**88,2**	**88,7**	**87,9**
	3	87,8	84,4	86,6	88,3	86,3
	5	85,6	82,9	83,7	88,1	86,5
	7	84,9	81,5	83,5	87,6	86,8
40	1	**91,4**	**83,4**	**85,5**	85,6	84,9
	3	87,8	80,5	84,3	**86,6**	**87,2**
	5	85,6	76,5	81,6	83,0	87,1
	7	84,9	74,5	81,7	83,4	85,0
60	1	**91,4**	**82,9**	85,7	86,2	**88,8**
	3	87,8	78,6	**85,8**	86,0	85,9
	5	85,6	73,6	85,3	**86,8**	86,1
	7	84,9	68,4	84,8	85,8	87,1
80	1	**91,4**	78,3	81,3	80,4	**82,4**
	3	87,8	81,0	82,0	80,0	78,3
	5	85,6	78,0	**84,7**	83,3	75,2
	7	84,9	**81,1**	**84,7**	**84,7**	77,3

Tabla A.3.: La columna "Original" muestra los resultados sin aplicar desequilibrios, "Base" que muestra los resultados tras aplicar el desequilibrio indicado sobre una clase, "SMOTE" muestra resultados tras pasar por el algoritmo SMOTE original, "B1" resultados del algoritmo BorderLine-SMOTE1 y "B2" los resultados del algoritmo BorderLine-SMOTE2. Los resultados reflejados están medidos con la medida de exactitud F1 y expresados en %. Los valores resaltados muestran los mejores resultados obtenidos para la F1 por cada situación de desequilibrio inicial.

NIST Muy solapado (clases V - Y)

Reducción	k	Original	Base	SMOTE	B1	B2
20	1	92,3	92,3	92,3	92,4	92,8
	3	92,6	91,6	92,3	93,4	92,9
	5	**94,0**	92,3	93,8	95,0	94,1
	7	**94,0**	**93,8**	**94,6**	**95,6**	**97,1**
40	1	92,3	88,5	89,8	89,3	88,8
	3	92,6	89,8	90,3	90,1	90,9
	5	**94,0**	89,1	90,8	92,3	**92,8**
	7	**94,0**	**91,2**	**92,0**	**93,5**	92,0
60	1	92,3	83,2	86,2	87,7	87,7
	3	92,6	**86,0**	86,3	87,5	85,7
	5	**94,0**	**86,0**	88,4	88,4	86,6
	7	**94,0**	80,4	**89,2**	**89,8**	**87,8**
80	1	92,3	77,0	77,6	74,2	74,7
	3	92,6	**86,0**	**86,3**	**87,5**	**85,7**
	5	**94,0**	78,6	82,6	84,0	75,0
	7	**94,0**	71,3	82,0	82,6	81,7

Tabla A.4.: La columna "Original" muestra los resultados sin aplicar desequilibrios, "Base" que muestra los resultados tras aplicar el desequilibrio indicado sobre una clase, "SMOTE" muestra resultados tras pasar por el algoritmo SMOTE original, "B1" resultados del algoritmo BorderLine-SMOTE1 y "B2" los resultados del algoritmo BorderLine-SMOTE2. Los resultados reflejados están medidos con la medida de exactitud F1 y expresados en %. Los valores resaltados muestran los mejores resultados obtenidos para la F1 por cada situación de desequilibrio inicial.

A.2. MNIST

MNIST Poco solapado (clases 1 - 5)

Reducción	k	Original	Base	SMOTE	B1	B2
20	1	**99,9**	**99,8**	**99,8**	**99,8**	**99,8**
	3	99,8	99,7	**99,8**	**99,8**	**99,8**
	5	99,7	99,7	99,7	99,7	99,7
	7	99,7	99,7	99,7	99,7	99,7
40	1	**99,9**	**99,7**	**99,7**	**99,9**	**99,8**
	3	99,8	99,4	**99,7**	99,5	99,4
	5	99,7	99,4	99,6	99,4	99,4
	7	99,7	99,4	99,5	99,4	99,4
60	1	**99,9**	**99,9**	**99,9**	**99,9**	**99,9**
	3	99,8	99,6	99,6	99,6	99,6
	5	99,7	99,8	**99,9**	99,3	99,3
	7	99,7	99,2	99,7	99,3	99,3
80	1	**99,9**	**99,7**	99,7	**99,7**	**99,7**
	3	99,8	**99,7**	**100,0**	**99,7**	99,5
	5	99,7	**99,7**	99,7	**99,7**	**99,7**
	7	99,7	99,5	99,7	99,5	99,5

Tabla A.5.: La columna "Original" muestra los resultados sin aplicar desequilibrios, "Base" que muestra los resultados tras aplicar el desequilibrio indicado sobre una clase, "SMOTE" muestra resultados tras pasar por el algoritmo SMOTE original, "B1" resultados del algoritmo BorderLine-SMOTE1 y "B2" los resultados del algoritmo BorderLine-SMOTE2. Los resultados reflejados están medidos con la medida de exactitud F1 y expresados en %. Los valores resaltados muestran los mejores resultados obtenidos para la F1 por cada situación de desequilibrio inicial.

MNIST Poco solapado (clases 3 - 6)

Reducción	k	Original	Base	SMOTE	B1	B2
20	1	99,6	**99,6**	**99,6**	**99,6**	**99,6**
	3	99,6	**99,6**	**99,6**	**99,6**	**99,6**
	5	**99,7**	**99,6**	**99,6**	**99,6**	**99,6**
	7	99,6	**99,6**	**99,6**	**99,6**	**99,6**
40	1	99,6	**99,6**	**99,6**	**99,6**	**99,6**
	3	99,6	**99,6**	**99,6**	**99,6**	**99,6**
	5	**99,7**	**99,6**	99,4	**99,6**	**99,6**
	7	99,6	99,5	99,5	99,5	99,5
60	1	99,6	**99,4**	**99,4**	**99,4**	**99,4**
	3	99,6	**99,4**	99,2	**99,4**	**99,4**
	5	**99,7**	99,2	99,2	99,2	99,2
	7	99,6	**99,4**	99,1	**99,4**	**99,4**
80	1	99,6	**99,2**	98,5	**99,2**	**99,2**
	3	99,6	**99,2**	**98,7**	**99,2**	**99,2**
	5	**99,7**	99,0	98,2	99,0	99,0
	7	99,6	99,0	98,5	99,0	99,0

Tabla A.6.: La columna “Original” muestra los resultados sin aplicar desequilibrios, “Base” que muestra los resultados tras aplicar el desequilibrio indicado sobre una clase, “SMOTE” muestra resultados tras pasar por el algoritmo SMOTE original, “B1” resultados del algoritmo BorderLine-SMOTE1 y “B2” los resultados del algoritmo BorderLine-SMOTE2. Los resultados reflejados están medidos con la medida de exactitud F1 y expresados en %. Los valores resaltados muestran los mejores resultados obtenidos para la F1 por cada situación de desequilibrio inicial.

MNIST Muy solapado (clases 3 - 5)

Reducción	k	Original	Base	SMOTE	B1	B2
20	1	96,5	95,6	95,7	95,6	95,6
	3	97,4	**96,9**	**97,1**	96,6	96,6
	5	97,3	96,7	**97,1**	96,4	96,6
	7	**97,5**	96,5	96,8	**96,8**	**97,0**
40	1	96,5	94,7	94,9	94,3	94,1
	3	97,4	**96,0**	96,2	95,2	94,6
	5	97,3	95,9	96,2	**96,4**	**95,1**
	7	**97,5**	95,9	**96,4**	96,3	94,9
60	1	96,5	94,2	94,5	93,4	92,8
	3	97,4	**95,3**	**95,9**	**94,3**	**93,1**
	5	97,3	94,4	94,8	93,9	**93,1**
	7	**97,5**	94,9	95,0	94,0	92,9
80	1	96,5	90,1	92,6	90,5	88,0
	3	97,4	**91,4**	94,7	**91,7**	**90,9**
	5	97,3	89,53	93,5	90,6	89,8
	7	**97,5**	87,6	**96,0**	91,0	88,5

Tabla A.7.: La columna "Original" muestra los resultados sin aplicar desequilibrios, "Base" que muestra los resultados tras aplicar el desequilibrio indicado sobre una clase, "SMOTE" muestra resultados tras pasar por el algoritmo SMOTE original, "B1" resultados del algoritmo BorderLine-SMOTE1 y "B2" los resultados del algoritmo BorderLine-SMOTE2. Los resultados reflejados están medidos con la medida de exactitud F1 y expresados en %. Los valores resaltados muestran los mejores resultados obtenidos para la F1 por cada situación de desequilibrio inicial.

MNIST Muy solapado (clases 7 - 9)

Reducción	k	Original	Base	SMOTE	B1	B2
20	1	97,6	97,4	97,4	**97,6**	**97,4**
	3	**97,8**	**97,6**	**97,6**	97,4	97,2
	5	97,5	97,2	97,2	97,3	97,3
	7	97,6	97,2	97,4	97,0	96,9
40	1	97,6	96,2	96,2	95,9	95,7
	3	**97,8**	**97,1**	**97,1**	96,2	**95,8**
	5	97,5	96,6	96,7	**96,3**	94,9
	7	97,6	96,6	96,1	95,9	95,1
60	1	97,6	95,6	**95,7**	**95,1**	94,0
	3	**97,8**	**96,5**	**95,7**	94,9	**94,7**
	5	97,5	95,6	95,1	94,9	93,2
	7	97,6	95,9	95,5	94,3	93,1
80	1	97,6	93,1	92,5	92,5	90,2
	3	**97,8**	**95,0**	**93,9**	91,0	**90,5**
	5	97,5	94,2	92,9	**94,9**	**90,5**
	7	97,6	94,9	93,2	91,5	90,3

Tabla A.8.: La columna "Original" muestra los resultados sin aplicar desequilibrios, "Base" que muestra los resultados tras aplicar el desequilibrio indicado sobre una clase, "SMOTE" muestra resultados tras pasar por el algoritmo SMOTE original, "B1" resultados del algoritmo BorderLine-SMOTE1 y "B2" los resultados del algoritmo BorderLine-SMOTE2. Los resultados reflejados están medidos con la medida de exactitud F1 y expresados en %. Los valores resaltados muestran los mejores resultados obtenidos para la F1 por cada situación de desequilibrio inicial.

A.3. USPS

USPS Poco solapado (clases 1 - 2)

Reducción	k	Original	Base	SMOTE	B1	B2
20	1	**99,7**	**99,4**	99,4	99,6	**99,7**
	3	**99,7**	**99,4**	**99,6**	**99,7**	**99,7**
	5	99,6	**99,4**	99,5	**99,7**	**99,7**
	7	99,5	**99,4**	99,4	99,6	99,6
40	1	**99,7**	**99,1**	99,3	99,3	99,5
	3	**99,7**	**99,1**	99,3	**99,7**	**99,7**
	5	99,6	99,0	**99,4**	99,3	99,4
	7	99,5	98,8	**99,4**	99,2	99,3
60	1	**99,7**	**98,9**	99,0	99,2	99,0
	3	**99,7**	98,6	98,9	**99,4**	**99,4**
	5	99,6	98,4	98,7	99,3	99,3
	7	99,5	98,6	**99,2**	**99,4**	99,3
80	1	**99,7**	**98,9**	**99,4**	**99,4**	**99,4**
	3	**99,7**	98,6	99,2	**99,4**	**99,4**
	5	99,6	98,3	**99,4**	98,9	98,9
	7	99,5	98,3	**99,4**	98,6	98,6

Tabla A.9.: La columna “Original” muestra los resultados sin aplicar desequilibrios, “Base” que muestra los resultados tras aplicar el desequilibrio indicado sobre una clase, “SMOTE” muestra resultados tras pasar por el algoritmo SMOTE original, “B1” resultados del algoritmo BorderLine-SMOTE1 y “B2” los resultados del algoritmo BorderLine-SMOTE2. Los resultados reflejados están medidos con la medida de exactitud F1 y expresados en %. Los valores resaltados muestran los mejores resultados obtenidos para la F1 por cada situación de desequilibrio inicial.

USPS Poco solapado (clases 3 - 4)

Reducción	k	Original	Base	SMOTE	B1	B2
20	1	99,7	**99,6**	**99,6**	**99,5**	**99,5**
	3	**99,8**	99,5	99,5	99,3	99,3
	5	**99,8**	99,5	99,5	**99,5**	**99,5**
	7	99,6	99,3	99,4	99,3	99,3
40	1	99,7	99,3	99,3	99,2	99,1
	3	**99,8**	99,4	99,4	**99,5**	99,3
	5	**99,8**	**99,5**	**99,5**	**99,5**	99,4
	7	99,6	**99,5**	99,4	**99,5**	**99,5**
60	1	99,7	**99,4**	99,2	**99,2**	**99,1**
	3	**99,8**	99,1	99,1	99,1	98,9
	5	**99,8**	98,9	**99,5**	98,9	98,9
	7	99,6	98,7	99,1	98,7	98,7
80	1	99,7	**99,4**	**99,4**	**99,7**	**99,4**
	3	**99,8**	**99,4**	98,5	98,8	98,8
	5	**99,8**	**99,4**	**99,4**	99,4	**99,4**
	7	99,6	**99,4**	99,1	99,4	**99,4**

Tabla A.10.: La columna "Original" muestra los resultados sin aplicar desequilibrios, "Base" que muestra los resultados tras aplicar el desequilibrio indicado sobre una clase, "SMOTE" muestra resultados tras pasar por el algoritmo SMOTE original, "B1" resultados del algoritmo BorderLine-SMOTE1 y "B2" los resultados del algoritmo BorderLine-SMOTE2. Los resultados reflejados están medidos con la medida de exactitud F1 y expresados en %. Los valores resaltados muestran los mejores resultados obtenidos para la F1 por cada situación de desequilibrio inicial.

USPS Muy solapado (clases 3 - 5)

Reducción	k	Original	Base	SMOTE	B1	B2
20	1	91,3	88,4	88,1	88,5	88,4
	3	93,3	89,8	90,2	90,3	90,1
	5	**93,7**	**91,5**	**92,2**	**92,1**	91,1
	7	**93,7**	90,1	91,1	**92,1**	**91,5**
40	1	91,3	87,3	88,2	89,0	88,0
	3	93,3	87,3	89,2	89,2	87,8
	5	**93,7**	87,8	90,6	90,6	**90,6**
	7	**93,7**	**88,1**	**91,1**	**91,7**	89,9
60	1	91,3	**83,9**	86,0	84,4	84,1
	3	93,3	81,9	86,8	86,2	85,3
	5	**93,7**	81,3	88,7	86,8	**86,3**
	7	**93,7**	80,8	**89,1**	**88,5**	85,7
80	1	91,3	**74,8**	77,0	77,1	78,0
	3	93,3	72,2	**84,3**	83,6	78,6
	5	**93,7**	66,1	83,8	83,6	**80,5**
	7	**93,7**	66,7	83,0	**84,1**	**80,5**

Tabla A.11.: La columna "Original" muestra los resultados sin aplicar desequilibrios, "Base" que muestra los resultados tras aplicar el desequilibrio indicado sobre una clase, "SMOTE" muestra resultados tras pasar por el algoritmo SMOTE original, "B1" resultados del algoritmo BorderLine-SMOTE1 y "B2" los resultados del algoritmo BorderLine-SMOTE2. Los resultados reflejados están medidos con la medida de exactitud F1 y expresados en %. Los valores resaltados muestran los mejores resultados obtenidos para la F1 por cada situación de desequilibrio inicial.

USPS Muy solapado (clases 7 - 9)

Reducción	k	Original	Base	SMOTE	B1	B2
20	1	95,3	93,3	93,6	94,0	93,7
	3	96,4	95,7	95,5	95,7	**95,7**
	5	**96,7**	95,8	95,8	95,9	95,6
	7	**96,7**	**96,4**	**96,0**	**96,0**	**95,7**
40	1	95,3	92,2	92,7	92,5	92,3
	3	96,4	94,1	94,3	93,4	93,8
	5	**96,7**	94,6	**94,6**	**94,0**	93,6
	7	**96,7**	**94,8**	94,5	**94,0**	**94,4**
60	1	95,3	91,3	91,7	90,6	90,5
	3	96,4	92,9	**93,5**	**92,2**	**91,1**
	5	**96,7**	**94,1**	92,9	90,4	90,1
	7	**96,7**	93,8	**93,5**	92,0	89,8
80	1	95,3	85,0	87,7	85,3	**84,7**
	3	96,4	88,4	88,4	**87,2**	83,3
	5	**96,7**	**90,0**	**88,6**	86,0	82,5
	7	**96,7**	88,6	88,4	87,1	83,6

Tabla A.12.: La columna "Original" muestra los resultados sin aplicar desequilibrios, "Base" que muestra los resultados tras aplicar el desequilibrio indicado sobre una clase, "SMOTE" muestra resultados tras pasar por el algoritmo SMOTE original, "B1" resultados del algoritmo BorderLine-SMOTE1 y "B2" los resultados del algoritmo BorderLine-SMOTE2. Los resultados reflejados están medidos con la medida de exactitud F1 y expresados en %. Los valores resaltados muestran los mejores resultados obtenidos para la F1 por cada situación de desequilibrio inicial.