Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

# Trabajo de Fin de Grado

Grado en Ingeniería Informática

# Tratamiento Inteligente de datos en la Actividad Deportiva y la Salud

*Intelligent treatment of data in Sports Activity and Health*

Carmen Castro González

D. **José Andrés Moreno Pérez**, con N.I.F. 42.935.437-A, Catedrático de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

**C E R T I F I C A**

Que la presente memoria titulada:

“Tratamiento Inteligente de Datos en la Actividad Deportiva y la Salud”

ha sido realizada bajo su dirección por **Carmen Castro González**,

con N.I.F.  43.836.129-S.

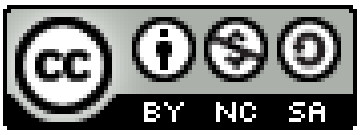
Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a SEIS de JULIO de 2020

# Agradecimientos

A mi tutor, José Andrés Moreno Pérez, por darme la oportunidad de profundizar en este campo de la informática, que se ha convertido en uno de mis favoritos, por su gran implicación y por su ayuda a lo largo de todo el proyecto.

A Miguel Garcia Torres,  profesor de la Universidad Pablo de Olavide (Sevilla), por brindarnos su experiencia y conocimiento, ayudando a conseguir mejores resultados en el trabajo.

# Licencia



# Resumen

La actividad deportiva y el ejercicio físico se relaciona cada vez de forma más pronunciada con la salud y el bienestar social de manera especial al alcanzar edades avanzadas. De esta forma, el creciente interés en todos los sectores sociales y económicos por la actividad deportiva moderada para favorecer la salud genera un gran volumen de datos para cuyo análisis se requieren herramientas inteligentes.

Las técnicas inteligentes del tratamiento de datos masivos permitirán construir herramientas para el apoyo en la toma de decisiones para favorecer la actividad deportiva y hábitos saludables adaptada a las necesidades de amplios sectores de la población para mejorar su salud y bienestar.

En este trabajo se estudiará el aprovechamiento de los datos proporcionados por organismos oficiales que realizan distintas Encuestas de Salud, como las Encuestas de Salud Canaria o la Encuesta Nacional de Salud de España o la Encuesta Europea..

La propuesta consiste en aplicar técnicas de IA para extraer conclusiones que relacionen la práctica de ejercicio o la actividad deportiva con el estado de salud. Por ello, se ha planteado el uso de técnicas de Aprendizaje Automático con herramientas específicas de R, Python o Weka, para procesar, estudiar y analizar los datos que permitan valorar esta relación y extraer conclusiones.



**Palabras clave:** Big data, Data Science, Inteligencia Artificial, Machine Learning, Salud, Deporte, Actividad física

## Abstract

Physical exercise is everyday more and more important when it comes to health and social welfare. The social and economical sectors are showing interest in the field of moderate physical activity to beneficiate health and this generate a huge amount of data that requires intelligent tools to manage them.

Intelligent techniques of massive data treatment will allow to build tools for the develop of different ways to favor physical activity and healthy habits, fitting this into the needs of the population to upgrade their health and welfare.

This project will study the use of data provided by official organizations that carry out different Health Surveys, such as the Canary, Spanish or European Health Surveys.

The proposal consist to apply AI techniques to obtain conclusions that relate the practice of exercise with the state of health. For this reason, it will be used Machine Learning techniques with R, Python or Weka to process, study and create predictive models with the data.



**Keywords:** Big data, Data Science, Artificial Intelligence, Machine Learning, Health, Sport, Physical Activity

# Índice

# Índice de figuras

# CAPÍTULO 1: Introducción

## 1.1 Motivación

La salud es un tema que nos atañe a todos de manera cada vez más significativa. Son múltiples los factores que tienen impacto sobre ella, desde la comida que se elige poner en el plato, hasta los antecedentes médicos personales y familiares, pasando por la práctica de actividad física que realiza cada persona.

El gran impacto que tienen estas variables, influye enormemente en la calidad de vida, de modo que resulta interesante su estudio para así evitar futuras complicaciones en la salud y generar una buena calidad de vida. Realizando diversos análisis basados en métodos de la disciplina Machine Learning se pretende conocer cuáles son los individuos más vulnerables a padecer determinadas enfermedades en función de la presencia o no de actividad física.

## 1.2 Objetivos

Los objetivos que se pretenden conseguir con este Trabajo Fin de Grado se diferencian en dos categorías:

1. Objetivo general:
   Estudiar el uso de herramientas informáticas para analizar la relación entre el ejercicio físico y el estado de salud aplicando técnicas de Aprendizaje Automático a datos obtenidos de la encuestas de salud

2. Objetivos específicos:
   - 2.1. Estudiar las características de las Encuestas de Salud
   - 2.2. Realizar el preprocesado de los datos de las encuestas
   - 2.3. Elegir las técnicas de Aprendizaje Automático apropiadas al objetivo general
   - 2.4. Evaluar las herramientas informáticas disponibles para aplicar estas técnicas
   - 2.5. Aplicar las técnicas con las distintas herramientas y comparar su rendimiento
   - 2.6. Analizar los indicadores obtenidos con las diferentes alternativas
   - 2.7. Visualizar de forma apropiada los resultados obtenidos
   - 2.8. Extraer conclusiones más relevantes que se hayan podido obtener

## 1.3 Esquema del documento

El documento está dividido en 5 capítulos, la distribución y contenido de los mismos es la siguiente:

- Capítulo 1: Introducción

Es el capítulo inicial del documento, donde se explica la motivación y objetivos del proyecto, para dar a entender el autor el por qué y el cómo se va a llevar a cabo.

- Capítulo 2: Antecedentes y Estado del arte

  Se introducen los conceptos más importantes relacionados con el trabajo, estos tienen que ver con la salud y el mundo de los datos. Así, se explica por encima todo lo necesario para entender el procedimiento de tratamiento de datos.

- Capítulo 3: Preparación de datos

  Se describe el conjunto de datos del que se dispone, así como el manejo de los valores ausentes que contiene y cómo manejarlos. Además, en este capítulo se creará el objetivo a predecir y se dará una breve introducción a las herramientas que se van a utilizar a lo largo del proyecto.

- Capítulo 4: Aplicación de clasificadores

  Este es el capítulo más extenso. Se describen los clasificadores escogidos para crear distintos modelos predictivos y se aplican al conjunto de datos en las tres herramientas descritas en el capítulo anterior. Se procede a seleccionar un subconjunto óptimo de atributos para reducir las dimensiones del conjunto de datos sin que afecte en gran medida a la capacidad predictiva de los clasificadores. Luego, se vuelven a aplicar los mismos clasificadores sobre el subconjunto de atributos resultantes de la selección de variables para comprobar la pérdida o no de capacidad predictiva. Por último, se localizan las variables de actividad física y se observa la importancia que tienen en relación al sobrepeso.

- Capítulo 5: Conclusiones

  Se exponen las conclusiones finales, relacionadas con la importancia de las variables de actividad física, el rendimiento de los distintos clasificadores y las diferencias encontradas en las herramientas.

# CAPÍTULO 2:  Antecedentes y Estado del arte

## 2.1 La salud en la sociedad

La salud ha sido, es y será un aspecto fundamental para el ser humano desde que la humanidad tiene conciencia del funcionamiento del cuerpo.

En las últimas décadas se ha ido poniendo de moda en la sociedad el culto al aspecto físico y se ha tomado consciencia de la importancia de la salud en general, causando el aumento

de la práctica deportiva en todos los grupos de edad, así como la demanda de asistencia sanitaria.

Estudios recientes [1] con datos de la Organización Mundial de la Salud [2] muestran a España como el segundo país más longevo del mundo, posiblemente reflejo de tener un servicio de salud público accesible para toda la población, junto con otros factores como el buen clima o la dieta.

A fin de verificar qué variables están influyendo en la salud de los españoles, es necesario estudiar su estilo de vida, antecedentes médicos y factores de riesgo, como pueden ser tabaquismo o enfermedades de alta incidencia en nuestra sociedad (diabetes, hipertensión, obesidad, etc).

Por ello, a través de las Encuesta de Salud de Canarias, disponibles mediante una solicitud al Instituto Canario de Estadística[1], se procede a hacer un análisis de las diversas cuestiones contestadas por los usuarios que realizaron dichas encuestas, en donde se incluyen preguntas que abordan aspectos relacionados con la salud en general: salud física, salud mental, hábitos tóxicos y conductas de promoción para la salud (dieta, actividad física, etc).

Antes de proceder con el desarrollo, es necesario un estudio del campo en el que queremos investigar, por eso se exponen los siguientes artículos con información relevante al tema que concierne.

Desde hace unos años se vienen publicando estudios que tratan de establecer relaciones del IMC con diversos tipos de factores. En el artículo [3] publicado en la revista oficial de la Sociedad Española de Salud Pública y Administración Sanitaria, se hace un estudio para comprobar la relación entre las horas que se ve la televisión, las horas de sueño y las horas de actividad física con el IMC en jóvenes de entre 17 y 35 años de edad. Los resultados muestran que los jóvenes que ven las televisión y que tienen menos actividad física, se asocian significativamente con mayor IMC. Sin embargo, las horas de sueño no parece ser un factor determinante.

Por otro lado, en el artículo [4] de BMC PUBLIC HEALTH, revista de atención médica de acceso abierto que cubre investigaciones sobre el tema de los servicios de salud, se utiliza la misma base de datos que la que se usará en este proyecto para relacionar el IMC con el nivel educativo. Algunas de las conclusiones más interesantes de dicho trabajo son, el descenso de la obesidad en Canarias en la última década, así como una relación inversamente proporcional entre el nivel de estudios y el índice de masa corporal (IMC), probando que el sector poblacional menos formado académicamente es, también, el más propenso a padecer sobrepeso u obesidad.

[1] http://www.gobiernodecanarias.org/istac/

## 2.2. Conceptos del mundo de los datos

Este trabajo se relaciona con varios campos del conocimiento que están conectados entre sí. Para establecer de forma más clara esta relación se procede a definir algunos de los conceptos básicos que intervienen.

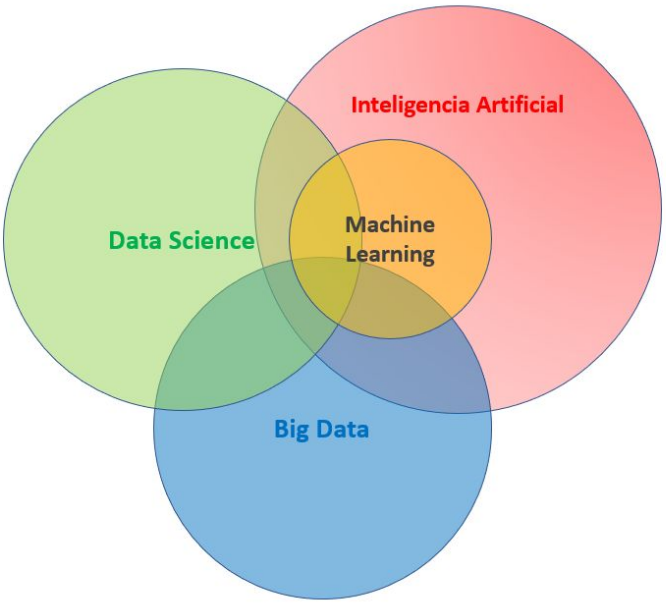



*Figura 1: Relación de conceptos*

### 2.2.1 Inteligencia Artificial

La Inteligencia Artificial [5] se refiere al estudio, al desarrollo y a la aplicación de técnicas informáticas que les permiten a las computadoras adquirir ciertas habilidades propias de la inteligencia humana. Algunas de estas habilidades en las que se han conseguido éxitos relevantes son identificar y reconocer objetos, analizar y resolver problemas, aprender a realizar nuevas tareas y mejorar en su realización, interpretar el lenguaje natural. En la mayoría de los avances en Inteligencia Artificial juegan un papel fundamental el tratamiento inteligente de los datos. Durante su desarrollo han aparecido distintos términos para identificar áreas específicas en el tratamiento de los datos como Ciencia de Datos (*Data Science*), Datos Masivos (*Big Data*), Minería de Datos (*Data Mining*), o Aprendizaje Automático (*Machine Learning*) que están muy relacionados,

### 2.2.2 Ciencia de Datos

La Ciencia de Datos (*Data Science*) [6][7] es la disciplina que abarca todo lo relacionado con la preparación y análisis de datos independientemente si son estructurados o no. Esto, de forma extendida, significa que combina diferentes disciplinas científicas como la estadística, la matemática y la programación con otra serie de habilidades entre las que se incluyen la resolución de problemas y la capacidad de análisis para poder extraer la mayor cantidad de datos posible, y transformar estos datos en información de valor

El término Big Data [8], [9] hace referencia al análisis masivo de datos que no pueden ser tratados por el software convencional. Desde sus comienzos, las características del Big Data se han definido en torno a las denominadas 3 Vs: Volumen, Velocidad y Variedad. El volumen se refiere a la cantidad masiva de datos generados y guardados. La velocidad está presente porque los datos se generan y fluyen a una velocidad sin precedentes y deben manejarse de manera oportuna. La variedad hace referencia a que los datos se presentan en todo tipo de formatos, desde texto sencillo, a imágenes, videos, hojas de cálculos y enteras bases de datos. Recientemente el número de Vs se ha ido aumentando, incorporándose la Velocidad y el Valor. La veracidad implica extraer y tratar los datos de alta calidad y eludir los que poseen mayores defectos. El valor hace referencia a la importancia de sacar a la luz los datos relevantes y sacarles provecho.

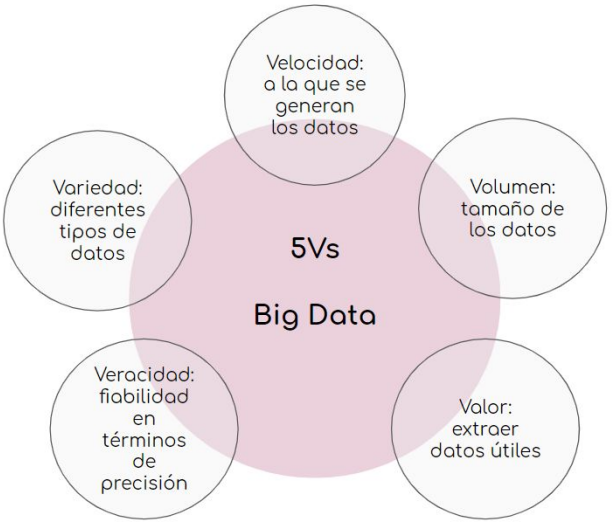



*Figura 2: Las 5 Vs en Big Data*

Los datos se procesan, se les da formato y almacenan, cargándose de manera distribuida en distintas fuentes, de este modo, datos de gran tamaño pueden ser divididos y distribuidos entre varios procesadores. Al haberlos estructurado, la búsqueda de los datos se hace a una mayor velocidad.

La Minería de Datos es el proceso de búsqueda en grandes bases de datos para encontrar información útil que sirva para la toma de decisiones. Se puede entender como la tecnología y software utilizado para encontrar patrones de comportamiento dentro de la base de datos. La base fundamental de esto es que esos patrones ayuden a la toma de decisiones.

Se suele componer de cuatro etapas principales:

- Determinación de los objetivos. Trata de la delimitación de los objetivos que el cliente desea bajo la orientación del especialista en data mining.
- Preprocesamiento de los datos. Se refiere a la selección, la limpieza, el enriquecimiento, la reducción y la transformación de las bases de datos. Esta etapa consume generalmente alrededor del setenta por ciento del tiempo total de un proyecto de data mining.
- Determinación del modelo. Se comienza realizando unos análisis estadísticos de los datos, y después se lleva a cabo una visualización gráfica de los mismos para tener una primera aproximación. Según los objetivos planteados y la tarea que debe llevarse a cabo, pueden utilizarse algoritmos desarrollados en diferentes áreas de la Inteligencia Artificial.
- Análisis de los resultados. Verifica si los resultados obtenidos son coherentes y los coteja con los obtenidos por los análisis estadísticos y de visualización gráfica. El cliente determina si son novedosos y si le aportan un nuevo conocimiento que le permita considerar sus decisiones.

## 2.2.3 Aprendizaje Automático

El término Aprendizaje Automático (*Machine Learning*) [10], [11], [12] se aplica a las técnicas y algoritmos de la Inteligencia Artificial que se usan para extraer información útil de los datos, que aprenden de su utilización y permiten estimar o predecir tendencias futuras. Generalmente el interés se centra en predecir una o varias variables a partir de los datos disponibles; cuando estas variables son numéricas se habla de regresión pero cuando son variables categóricas cuyos valores se interpretan como etiquetas que identifican clases se habla de clasificación. En Machine Learning en general, y en clasificación en particular, se distinguen dos tipos de problemas: los de aprendizaje o clasificación supervisada y los de aprendizaje o clasificación no supervisada.

### 2.2.3.1 Clasificación no supervisada

La clasificación no supervisada tiene lugar cuando no se dispone de datos “etiquetados” para el entrenamiento. Sólo se conocen los datos de entrada, pero no existen datos de salida que correspondan a un determinado input. Por tanto, sólo se puede describir la estructura de los datos, para intentar encontrar algún tipo de organización que simplifique el análisis. El aprendizaje no supervisado está más estrechamente ligado con la inteligencia

artificial, ya que la computadora puede aprender a identificar procesos y patrones complejos sin un humano para proporcionar orientación a lo largo del camino. La clasificación no supervisada implica la identificación de clases y generalmente se habla de clustering o agrupamiento de los datos en clases o "*clusters*" [13]. El objetivo general es conseguir clusters formado por elementos muy parecidos entre sí, pero diferentes de los elementos de otros clusters. Algunos de los algoritmos más representativos de la clasificación no supervisada son la clasificación jerárquica, el algoritmo de la k-media, la p-mediana o los centroides y las reglas de asociación.

#### 2.2.3.2 Clasificación supervisada

En la clasificación supervisada, se trabaja con datos "etiquetados", intentando encontrar un mecanismo que, dados los valores de las variables de entrada, les asigne la etiqueta o clase adecuada. Los algoritmos se entrenan con un histórico de datos y así aprenden a asignar la etiqueta de salida adecuada a nuevos valores, es decir, predice el valor de salida.

El aprendizaje supervisado comprende problemas de clasificación supervisada y problemas de regresión donde la diferencia entre ellas reside en el tipo de la variable respuesta. En clasificación es de tipo categórico, mientras que, en los casos de regresión, es de tipo numérico.

Los algoritmos más habituales que aplican para el aprendizaje supervisado son, Árboles de Decisión (*Decision Trees* o Random Forest), Algoritmos de los vecinos más cercanos (k-NN; k-*Nearest Neighbours*), Clasificadores Bayesianos, como el Naïve Bayes, Redes Neuronales, Regresión Logística, Máquinas de Vectores Soporte (*Support Vector Machines*; SVM), e incluso los Conjuntos de clasificadores (Métodos "Ensemble").

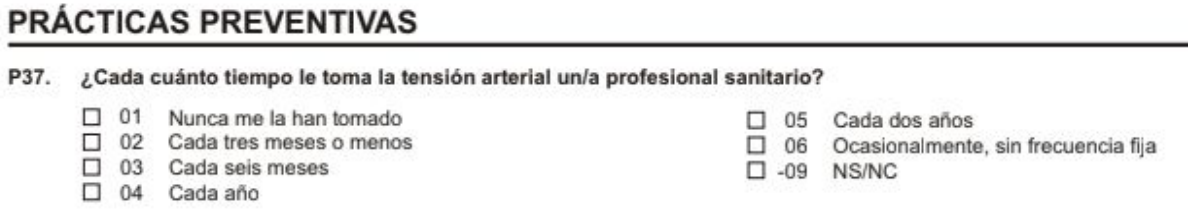
# CAPÍTULO 3: Preparación de datos

En este capítulo la descripción de los datos con los que se trabaja, el preprocesamiento de los mismos y se exponen las herramientas utilizadas durante el proyecto. En la sección 3.1 se describe el conjunto datos, la sección 3.2 se trata la elección de la variable de respuesta para realizar el análisis propuesto, en la sección 3.3 se trata la problemática ocasionada por los valores ausentes y en la sección 3.4 las herramientas utilizadas para el tratamiento de los datos.

## 3.1 Conjunto de datos

El Instituto Canario de Estadística[2] ha realizado las Encuesta de Salud de Canarias en los años 2004, 2009 y 2015 y el Servicio de Evaluación del Servicio Canario de Salud diseñará la de 2020[3]. En este trabajo usaremos los conjuntos de datos (*datasets*) resultantes de las Encuestas del año 2015[4]. El Gobierno de España realiza también de forma periódica las Encuesta Nacional de Salud[5], la última de las cuales se desarrolló en 2017 (las anteriores datan de 2006 y 2011/12) realizada por el Ministerio de Sanidad en colaboración con el Instituto Nacional de Estadística. La Unión Europea realiza la European Health Interview Survey (EHIS)[6], coordinada por Eurostat. Los datos sobre España de la última Encuesta Europea de Salud corresponden a 2014[7].

En estas encuestas se hacen preguntas relacionadas con información personal, situación económica, prácticas preventivas de salud, evaluación del estado de salud, salud mental, accidentes sufridos, morbilidad, limitaciones de actividad, consumo de medicamentos, hábitos higiénicos, alimentación, actividad física y descanso, consumo de tabaco y alcohol y apoyo y afecto personal. Para responder a la mayoría de las preguntas, hay que seleccionar una de las opciones que se representan por números y estos números son los que pasarán a formar parte del conjunto de datos.



**PRÁCTICAS PREVENTIVAS**

**P37. ¿Cada cuánto tiempo le toma la tensión arterial un/a profesional sanitario?**

- ☐ 01 Nunca me la han tomado
- ☐ 02 Cada tres meses o menos
- ☐ 03 Cada seis meses
- ☐ 04 Cada año
- ☐ 05 Cada dos años
- ☐ 06 Ocasionalmente, sin frecuencia fija
- ☐ -09 NS/NC

*Figura 3: Ejemplo de pregunta de las encuestas*

En el conjunto de datos (*dataset*) resultante de la encuesta de Salud de Canarias de 2015, cada pregunta se convierte en una variable, teniendo un total de 531 variables.

[2] http://www.gobiernodecanarias.org/istac/
[3] http://funcanis.es/encuesta-de-salud-de-canarias-2020/
[4] http://www.gobiernodecanarias.org/istac/temas_estadisticos/sociedad/salud/estadodesalud/C00035A.html
[5] https://www.mscbs.gob.es/estadEstudios/estadisticas/encuestaNacional/encuesta2017.htm
[6] https://ec.europa.eu/eurostat/web/microdata/european-health-interview-survey
[7] https://www.mscbs.gob.es/estadEstudios/estadisticas/EncuestaEuropea/Enc_Eur_Salud_en_Esp_2014.htm

# Encuesta de Salud de Canarias 2015

## *Diseño de registros*

La hoja *Variables* contiene la relación de variables existentes en el fichero público, con la estructura siguiente:

| | |
|---|---|
| **Nombre** | Nombre de la variable |
| **Etiqueta** | Descripción de la variable |
| **Tipo** | Tipo de variable: numérica o alfanumérica (texto) |
| **Tabla de etiquetas de valor** | Todas las variables cualitativas han sido codificadas. La tabla indicada contiene las etiquetas de los valores |

## *Fichero de datos*

| | |
|---|---|
| **Nombre del fichero** | ESC15_POBLACION_PUB.txt |
| **Número de resgistros** | 5703 |
| **Número de variables** | 531 |
| **Nombres de variables** | Los nombres de las variables están en la primera línea |
| **Delimitador de columnas** | Punto y coma |
| **Separador decimal** | Punto |
| **Valores perdidos** | Los valores perdidos de las variables son:<br>-1 'No procede'<br>-9 'No sabe / no contesta' |

*Figura 4: Diseño de registros de las encuestas*

Las dimensiones del conjunto de datos son 5703 columnas (tamaño de la muestra; número de encuestas realizadas) y 531 filas (número de variables). Además, hay que tener en cuenta que los valores -1 y -9 se deben tratar como valores ausentes.

Las variables que dan información acerca del ejercicio físico y la actividad deportiva de cada individuo participante en la encuesta son las que aparecen en la siguiente figura:

**HÁBITOS DE VIDA: DESCANSO Y ACTIVIDAD FÍSICA**

| Nº | Nombre | Etiqueta | Tipo | Tabla de etiquetas de valor |
|---|---|---|---|---|
| 438 | SVF001A | Actividad física en el trabajo | Numérica | TB_FISICA_TRAB |
| 439 | SVF011A | Actividad física en el tiempo libre +70 años | Numérica | TB_ACTIVIDAD_FISICA_NINO |
| 440 | SVF012A | Días actividad física intensa | Numérica | |
| 441 | SVF013A | Horas actividad física intensa | Numérica | |
| 442 | SVF013B | Minutos actividad física intensa | Numérica | |
| 443 | SVF014A | Días actividad física moderada | Numérica | |
| 444 | SVF015A | Horas actividad física moderada | Numérica | |
| 445 | SVF015B | Minutos actividad física moderada | Numérica | |
| 446 | SVF016A | Días actividad física caminar | Numérica | |
| 447 | SVF017A | Horas actividad física caminar | Numérica | |
| 448 | SVF017B | Minutos actividad física caminar | Numérica | |
| 449 | SVF018A | Horas actividad física sentado | Numérica | |
| 450 | SVF018B | Minutos actividad física sentado | Numérica | |
| 451 | SVF007A | Horas que duerme | Numérica | |
| 452 | SVF010A | Frecuencia actividad física (menores) | Numérica | TB_ACTIVIDAD_FISICA_NINO |
| 453 | SVF019A | Horas diarias TV (De lunes a viernes) | Numérica | |
| 454 | SVF019B | Horas diarias TV (En fin de semana) | Numérica | |
| 455 | SVF020A | Horas diarias Internet, videoconsola (De lunes a viernes) | Numérica | |
| 456 | SVF020B | Horas diarias Internet, videoconsola (En fin de semana) | Numérica | |
| 457 | SVF021A | Horas diarias Lectura (De lunes a viernes) | Numérica | |
| 458 | SVF021B | Horas diarias Lectura (En fin de semana) | Numérica | |

*Figura 5: Variables relacionadas con la actividad deportiva*

```
[ ] datos_todo <- read_csv("salud_canarias.csv")
    head(datos_todo)
```

```
Parsed with column specification:
cols(
  .default = col_double(),
  comarca = col_character(),
  isla = col_character(),
  isla_gru = col_character(),
  sd001a = col_character()
)

See spec(...) for full column specifications.
```

A tibble: 6 × 532

| id | idhog | comarca | isla | isla_gru | sd001a | clase_so | edad | sd002c | sd003a | ··· | phyper | ppeer | pprosoc | pebdtot | eva | apoyo | kp |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| <dbl> | <dbl> | <chr> | <chr> | <chr> | <chr> | <dbl> | <dbl> | <dbl> | <dbl> | ··· | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> | < |
| 1 | 1 | ES705A2 | ES705 | ES705 | F | 6 | 20 | 1995 | 1 | ··· | -1 | -1 | -1 | -1 | 1.0000000 | 22 | -1 |
| 2 | 2 | ES705A2 | ES705 | ES705 | F | 5 | 56 | 1959 | 2 | ··· | -1 | -1 | -1 | -1 | 1.0000000 | 50 | -1 |
| 3 | 3 | ES705A2 | ES705 | ES705 | M | 2 | 23 | 1992 | 1 | ··· | -1 | -1 | -1 | -1 | 1.0000000 | 53 | -1 |
| 4 | 4 | ES705A2 | ES705 | ES705 | F | 7 | 42 | 1973 | 2 | ··· | -1 | -1 | -1 | -1 | 1.0000000 | 44 | -1 |
| 5 | 4 | ES705A2 | ES705 | ES705 | M | 7 | 4 | 2011 | -1 | ··· | 2 | 2 | 10 | 6 | -1.0000000 | -1 | -1 |
| 6 | 5 | ES705A2 | ES705 | ES705 | M | 5 | 64 | 1951 | 2 | ··· | -1 | -1 | -1 | -1 | 0.2055692 | 37 | -1 |

*Figura 6: Dataset Encuesta Canaria de Salud de 2015*

La encuesta incluye cuestionarios distintos para los encuestados menores de 15 y los adultos (16 o más años). De las 5703 personas encuestadas en la encuesta de 2015, 1125 son menores y el resto, 4578 son personas de 16 o más años. Algunas de las preguntas realizadas en la encuesta eran diferentes si se referían a un menor a un adulto. La distribución de las 531 variables de la encuesta es la siguiente:

- Variables comunes a todas las encuestas: 159
- Variables sólo en las encuestas a mayores: 252
- Variables sólo en las encuestas a menores: 121

Se suelen utilizar métodos para reducir el número de variables sin perder mucha calidad en el análisis realizado, estos se verán más adelante.

Otro aspecto a tener en cuenta, es que todas los datos almacenados son de tipo numérico excepto 4. Como se mencionó anteriormente, estos números generalmente no tienen un significado numérico pero en muchos casos tienen un sentido ordinal, como se puede apreciar en la figura 3, lo que podría dar pié a un tipo de análisis específico. El hecho de que figuren los datos como tipo numérico es conveniente porque muchos clasificadores sólo soportan variables de entrada de este tipo, y además, en el caso de que sean variables ordinales el orden entre ellas es el asociado a su significado. El problema es tener variables nominales codificadas en los datos como números, ya que en estas el orden puede no tener significado.

Es importante la visualización y entendimiento de los datos. Los datos disponibles de las encuestas vienen con el formulario de las preguntas y la descripción del tipo y significado de cada variable, lo que favorece la comprensión e interpretación de los datos.

## 3.2 Valores ausentes

Como ya se mencionó, los datos provienen de 2 tipos de encuestas, uno para mayores (16 o más años) y otro para menores (15 o menos años) con muchas preguntas distintas. Esto ocasiona que muchas de las variables dirigidas sólo a uno de los grupos tenga demasiados "valores ausentes", lo que en el dataset figura como -1 = 'No procede' o -9 = 'No sabe/No contesta'.

Para tratar de evitar la distorsión que provocan las variables con un elevado número de valores ausentes, se descartan las variables con muchos valores ausentes. Primero, se divide el conjunto de datos en dos conjuntos de datos (*datasets*), uno con las encuestas a menores y otro con las encuestas a mayores. También se pasan los valores que figuran como -1 y -9 a NA (*Not Available*) que representa valores ausentes, y se eliminan las variables con un alto porcentaje de valores NA. Por último, se vuelven a imputar los valores NA restantes con -1. La imputación de valores ausentes puede ser por la media, la mediana, la moda, entre otros, pero en este caso se volverá a imputar con el valor -1.

```
# Separo dataset en menores y mayores de 15 porque se les han hecho encuenstas distintas
datos_menores <- datos[which(datos$edad <= 15),]
datos <- datos[which(datos$edad > 15),]
dim(datos)            # dataset de mayores de 15
dim(datos_menores)    # dataset de menores de 15

# MISSING DATA DESPUES DE PARTIR EL DATASET ENTRE MENORES Y MAYORES
# Si la mayoria de valores son -1 los elimino

# con el dataset de mayores
# cambio los -1 a NA
map_dbl(datos, .f = function(x){sum(x == -1)})
map_dbl(datos, .f = function(x){sum(x == -9)})
datos[datos == -1] <- NA
datos[datos == -9] <- NA
mean(is.na(datos))

# elimino variables con mas del 70% de valores como NA
x <- datos[ lapply( datos, function(x) sum(is.na(x)) / length(x) ) < 0.7 ]
dim(x)


is.na(x)
mean(is.na(x))
sum(is.na(x))
apply(is.na(x), 2, mean)

datos <- x
datos[is.na(datos)] <- -1
sum(is.na(datos))
dim(datos)
```

*Figura 7: División del dataset y control de valores NA en R*

Se eliminan las variables que les faltan el 70% de la información. El resultado es una reducción del número de variables en el conjunto de datos que pasa de tener una dimensión de 5703 filas y 529 columnas, al conjunto de datos de mayores con 4578 filas y 235 columnas, y el de menores 1125 filas y 175 columnas.

Paralelamente, para saber cuántas de las variables relacionadas con la actividad física les afectaría este proceso, se crea un nuevo conjunto de datos formado por los atributos relacionados con el ejercicio físico que figuran enumeradas en la figura 5. Con este conjunto de datos (*dataset df_ejercicio*) se hace de forma análoga a lo comentado anteriormente, se pasan NA (valores ausentes) los datos con -1 y -9, y se eliminan las variables que superen el 70% de valores ausentes.

```
# Creo dataset que almacena solo las variables de ejercicio
df_ejercicio <- datos[c(434:454)]
dim(df_ejercicio) # 21 variables

# Cambio los -1 y -9 a NA
df_ejercicio[df_ejercicio == -1] <- NA
df_ejercicio[df_ejercicio == -9] <- NA

# La media de NA del dataset
mean(is.na(df_ejercicio)) # más de 60%

# La media de NA por variable
apply(is.na(df_ejercicio), 2, mean)

# elimino variables con mas del 70% de valores como NA
x <- df_ejercicio[ lapply( df_ejercicio, function(x) sum(is.na(x)) / length(x) ) < 0.7 ]
dim(x) # quedan 9 variables (se eliminan 12)

mean(is.na(x)) # 19% de valores ausentes
apply(is.na(x), 2, mean)
```

*Figura 8: Eliminación de variables con NA en el dataset de actividad física*

El conjunto de datos relacionados con el ejercicio físico (*df_ejercicio*) se reduce de 21 variables y un 58% de valores ausentes a tan sólo 9 variables y un porcentaje de valores ausentes del 17%, al eliminar aquellas variables con más del 70% de valores NA. En la figura 9, se aprecia que antes de la eliminación de variables, incluso hay algunas que tienen un 100% de valores ausentes.

```
> apply(is.na(df_ejercicio), 2, mean)
   SVF001A    SVF011A    SVF012A    SVF013A    SVF013B    SVF014A    SVF015A    SVF015B    SVF016A
 SVF017A    SVF017B    SVF018A    SVF018B    SVF007A
0.0000000 0.8228484 0.1771516 0.7857143 0.7857143 0.1771516 0.7289209 0.7289209 0.1771516
 0.3665356 0.3665356 0.1771516 0.1771516 0.1771516
   SVF010A    SVF019A    SVF019B    SVF020A    SVF020B    SVF021A    SVF021B
1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
```

*Figura 9: Valores ausentes por variable del dataset df_ejercicio*

Hay que tener en cuenta estas 9 variables, que definen la actividad deportiva, para detectar cuales se van a escoger posteriormente en la selección de variables.

Variables de actividad física escogidas

| Nº | Nombre | Etiqueta | Tipo | Tabla de etiquetas de valor |
|---|---|---|---|---|
| 438 | SVF001A | Actividad física en el trabajo | Numérica | TB_FISICA_TRAB |
| 440 | SVF012A | Días actividad física intensa | Numérica | |
| 443 | SVF014A | Días actividad física moderada | Numérica | |
| 446 | SVF016A | Días actividad física caminar | Numérica | |
| 447 | SVF017A | Horas actividad física caminar | Numérica | |
| 448 | SVF017B | Minutos actividad física caminar | Numérica | |
| 449 | SVF018A | Horas actividad física sentado | Numérica | |
| 450 | SVF018B | Minutos actividad física sentado | Numérica | |
| 451 | SVF007A | Horas que duerme | Numérica | |

*Figura 10: Variables de actividad física finalmente escogidas*

## 3.3 Variable de respuesta

Una vez descritos los datos, es necesario establecer cuál es la variable sobre la que se van a hacer las predicciones. Con las variables disponibles, se va a establecer una variable respuesta que será el objetivo a predecir por los modelos predictivos. Esta variable deberá reflejar el aspecto de la salud en la que más directamente influye el ejercicio físico y la actividad deportiva moderada.

La Organización Mundial de la Salud ha adoptado el índice de masa corporal como una manera usada en los estudios de salud para valorar los problemas relacionados con el peso. El Índice de Masa Corporal (IMC) que se calcula dividiendo el peso de las personas en kg por el cuadrado de su altura en metros

$$IMC = peso(kg)/talla(m)^2$$

Haciendo uso de dos variables que figuran en el dataset, el peso y la altura, para relacionarlo con el estado de salud de cada individuo a partir del IMC.

La Organización Mundial de la Salud establece para los adultos que hay sobrepeso cuando el IMC es 25 o superior y obesidad cuando es 30 o superior[8]. Además, en los estudios relacionados con la salud se establece el infrapeso en los adultos que tienen un IMC inferior a 20, quedando como personas con normapeso los que tienen un IMC entre 20 y 25. Una primera aproximación fue la de crear una variable respuesta de tipo categórica, teniendo así un problema de clasificación, en donde el nombre de la variable sería “Estado” y tendría 4 clases: “Infrapeso”, “Normapeso o Peso Normal”, “Sobrepeso” y “Obesidad”. Sin embargo, las características especiales del problema sesgaba la capacidad predictiva de los algoritmos, alcanzando en el mejor de los casos poco más del 50%.

[8] https://www.who.int/es/news-room/fact-sheets/detail/obesity-and-overweight

No se trataría de un problema de clasificación normal, sino un problema de clasificación ordinal, ya que las etiquetas tienen un orden. Observando la matriz de confusión se detecta que una gran parte de los errores suelen encontrarse entorno a la diagonal principal.

Por tanto, se decidió cambiar la variable respuesta a una de tipo binomial, para identificar únicamente los casos que más interesan, en este caso, el sobrepeso. Se creó la variable respuesta llamada "Sobrepeso", utilizando nuevamente la fórmula del IMC y teniendo 2 valores "Si", en caso de sobrepeso y obesidad, y "No", en caso contrario. El código en R que crea esta nueva columna y la añade al dataset es:

```
datos <- datos %>%
  mutate(Sobrepeso = case_when((kg/((altura/100)^2)) <= 25  ~ "No",
                               (kg/((altura/100)^2)) > 25  ~ "Si"))
datos$Sobrepeso <- as.factor(datos$Sobrepeso)
```

*Figura 11: Creación de variable respuesta en R*

Además se eliminan las variables *kg* y *altura* después de usarlas porque estas definen la variable respuesta, al igual que la variable llamada *imc* de tipo numérica que figura en el conjunto de datos. Por último se eliminan también *id* e *idhog*, que identifican la persona encuestada y el hogar, porque se sabe que no dan información de valor.

```
datos <- select(datos, -id, -idhog, -kg, -altura, -imc)
```

*Figura 12: Eliminación de variables en R*

## 3.4 Herramientas utilizadas

Existe una multitud de herramientas para implementar algoritmos de clasificación, la mayoría son herramientas de tipo estadístico cuyo uso requiere un desembolso económico. Entre las herramientas libres se han elegido tres de las más accesibles: R, Python y Weka. A continuación se describen muy brevemente estas tres herramientas y las características que las diferencian.

### 3.4.1 R

R[9] es uno de los lenguajes y entornos más efectivos para analizar y manipular los datos con fines estadísticos. En cuanto a Aprendizaje Automático, R tiene implementados una gran cantidad de algoritmos, como consecuencia de las diferentes líneas de investigación de

[9] https://www.r-project.org/

grupos que dieron pie a su creación, debido precisamente al hecho de que R nació en el ámbito académico [14]. Algunas de las librerías más útiles en el campo de Análisis de Datos, visualización y Aprendizaje Automático son *dplyr*, *ggplot2*, *corrplot*, *tidyverse*, *mlr*, *randomForest* o *caret*.

El lenguaje R es más potente que Python a la hora de visualizar datos, y la red de paquetes de modelos estadísticos para R es más extensa que en Python. Como contra de R, está su curva de aprendizaje, que suele ser más lenta y complicada si se compara con las otras dos opciones, y la lentitud a la hora de crear modelos predictivos y algoritmos que requieren iterar sobre el dataset.

### 3.4.2 Python

Python[10] es un lenguaje multipropósito, se puede utilizar en ciencias de datos, desarrollo y administración de sistemas, construcción de aplicaciones web y scripts, entre otros. Contiene paquetes que ayudan en las tareas de análisis y tratamiento de datos, y Aprendizaje Automático, entre otros, con los paquetes *Pandas*, *Scikit-learn*, *SciPy* o *NumPy*.

A favor de Python, cabe mencionar su curva de aprendizaje suave, versatilidad, velocidad, mejor integración y legibilidad, lo que provoca un aumento de productividad en proyectos. En cuanto a las desventajas, cabe destacar que incluye menos paquetes de modelos estadísticos.

### 3.4.3 Weka

Weka[11] es una plataforma de software libre para Aprendizaje Automático (*Machine Learning*) y la Minería de Datos (*Data Mining*) escrito en Java y desarrollado en la Universidad de Waikato. Contiene una colección de herramientas de visualización y algoritmos para análisis de datos y modelado predictivo, unidos a una interfaz gráfica de usuario para acceder fácilmente a sus funcionalidades.

Las principales ventaja de Weka es que es muy fácil de usar y no hace falta programar, así que es apta para principiantes o no expertos en programación. La creación de modelos predictivos ha sido más rápida en Weka que en las anteriores opciones y con la opción de filtros que ofrece se puede realizar preprocesado de datos como la selección de variables o balanceo de clases (entre otros). La mayor desventaja de Weka es que es la menos flexible de las tres opciones.

[10] https://www.python.org/
[11] https://www.cs.waikato.ac.nz/ml/weka/

# CAPÍTULO 4: Aplicación de clasificadores

En este capítulo se van a aplicar distintos clasificadores para tratar de estimar, a través del IMC, la categoría en la que se encuentra la persona encuestada. Se aplicarán algunos métodos de selección variables. Con ello podemos evaluar si ocurre una disminución en la calidad de la predicción al utilizar los métodos de selección de variables. Se emplearán los mismos clasificadores antes y después de aplicar los métodos de selección para evaluar la pérdida en capacidad predictiva. Se prestará especial atención al papel de las variables asociadas al ejercicio físico y la actividad deportiva.

## 4.1 Selección de clasificadores

Los clasificadores han sido escogidos debido a que son los más populares en la documentación estudiada.

En este proyecto se trabajarán con los siguientes:

- **Clasificador de los vecinos más próximos** (*k-Nearest-Neighbor; k-NN*):
  El clasificador de los *k* vecinos más próximos es un algoritmo basado en instancias, que busca las "k" observaciones más cercanas a la que se está tratando de predecir y se clasifica en función de la clase de la mayoría de datos que le rodean. En este clasificador influye la función de distancia que se adopte o los pesos asignados a las distintas características.

- **Regresión Logística**:
  Es un método estadístico para predecir clases binarias basado en probabilidades. Mide la relación entre la variable dependiente categórica y una o más variables independientes, estimando las probabilidades utilizando una función logística. La función logística es de la forma $f(x) = 1/(1 + e^{x})$ también denominada sigmoide.

- **Naïve Bayes**:
  Es uno de los algoritmos más simples y poderosos para la clasificación basado en el Teorema de Bayes, con una suposición de independencia condicional entre los variables predictoras, dado el resultado. Entre los clasificadores bayesianos, el clasificador bayesiano ingenuo o Naïve Bayes es fácil de construir y particularmente útil para conjuntos de datos muy grandes. Se trata de establecer la clasificación más probable a posteriori, una vez conocidas las variables predictoras. Las proporciones en la muestra establecen las probabilidades a priori y las probabilidades condicionales directas para cada valor de cada variable predictora según la clase.

- **Árboles de Decisión**:
  Los árboles de decisión (*Decision Trees*) o árboles de clasificación son

representaciones gráficas en forma de árbol donde en cada nodo de ramificación se toma una decisión basadas en ciertas condiciones de las variables predictoras. Es uno de los algoritmos de aprendizaje supervisado más utilizados en aprendizaje supervisado y pueden realizar tareas de clasificación o regresión. La comprensión de su funcionamiento suele ser simple y a la vez muy potente. Algunos de los clasificadores basados en árboles de decisión más corrientes son el ID3, el C4.5, el J48 o el CART.

- **Random Forest**:
  Es un algoritmo capaz de realizar tanto tareas de regresión como de clasificación. Se ejecutan varios algoritmos de árbol de decisiones en lugar de uno solo. Para clasificar un nuevo objeto basado en atributos, cada árbol de decisión da una clasificación y finalmente la decisión con mayor "votos", es decir, la clasificación más repetida, es la predicción del algoritmo.

Se aplican estos clasificadores utilizado R, Python y Weka para el conjunto de datos correspondiente a los mayores de 15 años. Para evaluar el rendimiento de los clasificadores se usará validación cruzada. Puesto que se dispone de sólo un conjunto de datos se divide en dos partes, una se toma como conjunto de datos de entrenamiento (*Training dataset*) y el resto como conjunto de datos de prueba (*Test dataset*). Una proporción usual es 80-20 de forma que el conjunto de entrenamiento tiene el 80% de los datos, seleccionados al azar, y el 20% restante se toma como datos de prueba. Con el conjunto de entrenamiento se realizará la validación cruzada con *k* iteraciones (*k-Fold Cross Validation*) con cada uno de los clasificadores. Para aplicar la *k*-validación cruzada se divide aleatoriamente el conjunto de datos en *k* partes similares y cada una de ellas se toma una vez como conjunto de test mientras que el resto es el conjunto de entrenamiento. En cada iteración se emplean k-1 grupos para entrenar el algoritmo y el grupo restante como conjunto de test, este proceso se repite k veces utilizando un grupo distinto como test en cada iteración.

### 4.1.1 En R

Se utiliza la librería *caret* para crear los clasificadores. El paquete *caret* (acrónimo de *Classification And REgression Training*) tiene un conjunto de funciones que facilitan el proceso de crear modelos predictivos.

Primero se divide el conjunto de datos en los conjuntos de entrenamiento y de test en una proporción 80-20 de forma aleatoria.

```r
set.seed(1234)
split <- sample.split(datos$Estado, SplitRatio = 0.80)
training_set <- subset(datos, split == TRUE)
test_set <- subset(datos, split == FALSE)

# Hay que comprobar en ambos dataset los valores sean parecidos
prop.table(table(training_set$Estado)) %>% round(digits = 2)
prop.table(table(test_set$Estado)) %>% round(digits = 2)
```

*Figura 13: R: División del dataset*

Por tanto, el conjunto de datos correspondientes a las 5703 encuestas se descompone en un conjunto de datos de entrenamiento de 3663 instancias y un conjunto de datos de test a 915 instancias, ambos con 234 variables.

Se procede a entrenar los distintos clasificadores en el conjunto de entrenamiento, usando validación cruzada con k = 10 iteraciones (k-Fold Cross Validation) a través de la función *train*() de *caret*. Esta función permite crear modelos con distintos clasificadores, parámetros y métodos de muestreo. Los parámetros que deben especificarse son: la variable respuesta, el dataset de entrenamiento, el algoritmo a utilizar y el método de remuestreo (en este caso se escoge siempre “cv”, con fold = 10).

Como resultado, para cada clasificador se expondrán los indicadores por defecto de *caret*: la precisión (*accuracy*) y el índice *kappa de Cohen*. La precisión es simplemente la proporción de clasificaciones correctas de todas las instancias, mientras que *kappa* trata de descontar la proporción de clasificaciones correctas que se producirían por azar. El valor óptimo de kappa es 1 pero podría incluso llegar a ser negativo. Si *a* es la precisión y *p* es la probabilidad de que la clasificación al azar sea correcta entonces *kappa* = (*a*-*p*)/(1-*p*). El estadístico *kappa* tiene una difícil interpretación y fue introducido por Cohen [15].

- K - Nearest Neighbors (KNN)

```r
# K-Nearest Neighbors (KNN)

grid = expand.grid(k = c(7, 50, 100))
set.seed(123)
model_knn <- caret::train(Sobrepeso ~ .,
                          data = training_set,
                          method = "knn",
                          tuneGrid = grid,
                          trControl = trainControl(method = "cv",
                                                   number = 10,
                                                   search = "grid"))
```

*Figura 14: R: Clasificador KNN antes de selección de variable*

```
> model_knn
k-Nearest Neighbors

3663 samples
 233 predictor
   2 classes: 'No', 'Si'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3296, 3296, 3298, 3297, 3296, 3297, ...
Resampling results across tuning parameters:

  k    Accuracy   Kappa
    7  0.5755105  0.1290531
   50  0.6052570  0.1761084
  100  0.5943243  0.1450598

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 50.
```

*Figura 15: R: Resultado de KNN*

Se especifica en la función train() que se debe ejecutar el clasificador con k = 7, 50 y 100. El k que mejor resultados obtiene es k = 50.

- Regresión Logística

```
# Logistic Regression

set.seed(123)
model_lg<- caret::train(Sobrepeso ~ .,
                        data = training_set,
                        method = "glm",
                        family = "binomial",
                        trControl = trainControl(method = "cv",
                                                 number = 10))
```

*Figura 16: R: Clasificador Regresión Logística antes de selección de variable*

```
> model_lg
Generalized Linear Model

3663 samples
 233 predictor
   2 classes: 'No', 'Si'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3297, 3297, 3296, 3296, 3296, 3297, ...
Resampling results:

  Accuracy   Kappa
  0.6467275  0.2798586
```

*Figura 17: R: Resultado de Regresión Logística*

- Naïve Bayes

```
# Naive Bayes

set.seed(123)
model_nb<- caret::train(Sobrepeso ~ .,
                        data = training_set,
                        method = "naive_bayes",
                        trControl = trainControl(method = "cv",
                                                 number = 10))
```

*Figura 18: R: Clasificador Naive Bayes antes de selección de variable*

```
> model_nb
Naive Bayes

3663 samples
 233 predictor
   2 classes: 'No', 'Si'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3297, 3296, 3296, 3296, 3297, 3297, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.5785033  0.1876399
   TRUE      0.6260064  0.2104953

Tuning parameter 'laplace' was held constant at a value of 0
Tuning parameter 'adjust' was held
 constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were laplace = 0, usekernel = TRUE and adjust = 1.
```

*Figura 19: R: Resultado de Naive Bayes*

- Decision Tree (CART)

```
# Decision tree

set.seed(123)
model_dtree<- caret::train(Sobrepeso ~ .,
                           data = training_set,
                           method = "rpart",
                           trControl = trainControl(method = "cv",
                                                    number = 10))
```

*Figura 20: R: Clasificador CART antes de selección de variable*

```
> model_dtree
CART

3663 samples
 233 predictor
   2 classes: 'No', 'Si'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3296, 3296, 3298, 3297, 3296, 3297, ...
Resampling results across tuning parameters:

  cp          Accuracy   Kappa
  0.02953846  0.6109895  0.2066868
  0.04984615  0.5978605  0.1671434
  0.06430769  0.5834221  0.1059061

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.02953846.
```

*Figura 21: R: Resultado de CART*

El indicador *cp* se refiere al parámetro de complejidad, y se usa como una medida de robustez para podar el árbol, consiste en evitar dividir una partición si la división no mejora significativamente la calidad general del modelo. Cp impone una penalización al árbol por tener dos divisiones.

- Random Forest

```
# Random Forest

set.seed(123)
model_rf <- caret::train(Sobrepeso ~ .,
                         data = training_set,
                         method = "rf",
                         trControl = trainControl(method = "cv",
                                                  number = 10))
```

*Figura 22: R: Clasificador Random Forest antes de selección de variables*

```
> model_rf
Random Forest

3663 samples
 233 predictor
   2 classes: 'No', 'Si'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3296, 3296, 3298, 3297, 3296, 3297, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
    2   0.6584740  0.2929625
  125   0.6647492  0.3137138
  249   0.6683033  0.3209125

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 249.
```

*Figura 23: R: Resultados del rf*

El parámetro *mtry* se refiere a el número de variables aleatorias disponibles para dividir en cada nodo del árbol.

Se puede apreciar que con Random Forest y Regresión Logística, se obtienen unos resultados predictivos mejores que con CART, KNN y Naive Bayes. De momento estas son las mejores opciones para realizar las predicciones.

### 4.1.2 En Python

Para la creación de modelos en Python, se usará la librería *scikit-learn*. Esta es una librería especializada para aprendizaje automático, incluye varios algoritmos de clasificación en forma de funciones. La librería *scikit-learn* y está diseñada para trabajar junto con otras librerías numéricas y científicas, NumPy y SciPy, que han sido cargadas también en el proyecto.

En Python se pueden realizar fácilmente las operaciones descritas anteriormente de preparación de de datos. En particular, crear la variable de respuesta, dividir el conjunto de datos entre los menores y mayores de 15 años, eliminación de variables con 70% de valores ausentes que se reemplazan por -1 (en Python los valores ausentes se identifican con el valor NaN ; *Not a Number*). En la siguiente figura se divide el dataset en

entrenamiento y test, para luego aplicar la validación cruzada de 10 capas al dataset de entrenamiento.

```
[158] # Elimino variables de tipo factor para que funcionen los clasificadores
      dataset=df_mayores.drop(columns=[ "comarca", "isla", "isla_gru", "sd001a"])

      X = dataset.drop(['Sobrepeso'], axis = 1)
      y = dataset.Sobrepeso

      # split data train 80 % and test 20 %
      X_train, X_validation, Y_train, Y_validation = train_test_split(X, y, test_size=0.20, random_state=1)
```

*Figura 24: Python: División del dataset*

```
# Se van a crear modelos para estimar la precisión
# se usara validación cruzadda de 10 capas ( 9 de train y 1 de test de 1)

# Spot Check Algorithms
models = []
models.append(('KNN', KNeighborsClassifier(n_neighbors=100)))
models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
models.append(('NB', GaussianNB()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('RF', RandomForestClassifier(n_estimators=100)))
# evaluate each model in turn
results = []
names = []
for name, model in models:
  kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
  cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy')
  results.append(cv_results)
  names.append(name)
  print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
```

```
KNN: 0.599675 (0.013196)
LR: 0.640089 (0.025731)
NB: 0.579198 (0.030404)
CART: 0.580016 (0.024046)
RF: 0.655124 (0.029716)
```

*Figura 25: Python: Creación de modelos*

Con este breve código de Python se observa en la salida que la primera columna numérica corresponde a la precisión del modelo, y la segunda, al índice *kappa de Cohen.*

Se observa que el mejor clasificador es el Random Forest, seguido por la Regresión Logística. Se pueden pasar parámetros a los clasificadores para ser modificados, en caso de no especificar nada, se utilizan los valores por defecto. Por ejemplo, en el Random Forest se especifica que el número de árboles (*n_estimators*) sea 100, y en el K Vecinos más Cercanos, se establece el número de “k” a 100.

### 4.1.3 En Weka

Crear modelos en Weka es muy sencillo, sólo hay que ir a la pestaña *Classify* y escoger el clasificador deseado con los parámetros específicos. Se escogen los mismos clasificadores usados en R y Python, y en la opción *Test Options*, se elige validación cruzada (*Cross-validation*) con 10 iteraciones (*Folds* = 10).

Al escoger un clasificador y ejecutarlo, a la derecha se reflejan distintas métricas como el porcentaje de instancias clasificadas correctamente, la precisión por clase y la matriz de confusión, entre otros.

Para la creación del árbol de decisión se ha usado J48 en vez de CART. El árbol de decisión J48 es una implementación Java de código abierto del algoritmo C4.5 en Weka. El árbol de decisión CART usado en R es muy similar a C4.5, pero difiere en que admite variables de respuesta numéricas para la regresión y no calcula conjuntos de reglas.

Se han escogido cinco clasificadores, y este es el porcentaje de Instancias clasificadas correctamente:

- K Nearest Neighbors (k = 100): 63.6522 %
- Regresión Logística: 65.1376 %
- Naive Bayes: 61.0092 %
- Árbol de Decisión (J48): 57.6234 %
- Random Forest: 65.6182 %

Los métodos con un mayor porcentaje de predicción (*accuracy*) son el Random Forest y Regresión Logística.

En la figura 26 aparece la matriz de confusión del Random Forest,se puede ver que el ratio de verdaderos positivos y negativos. Se observa que se clasifican con mayor acierto las instancias con calor “Si”. Esto puede deberse al desbalanceo de clases.

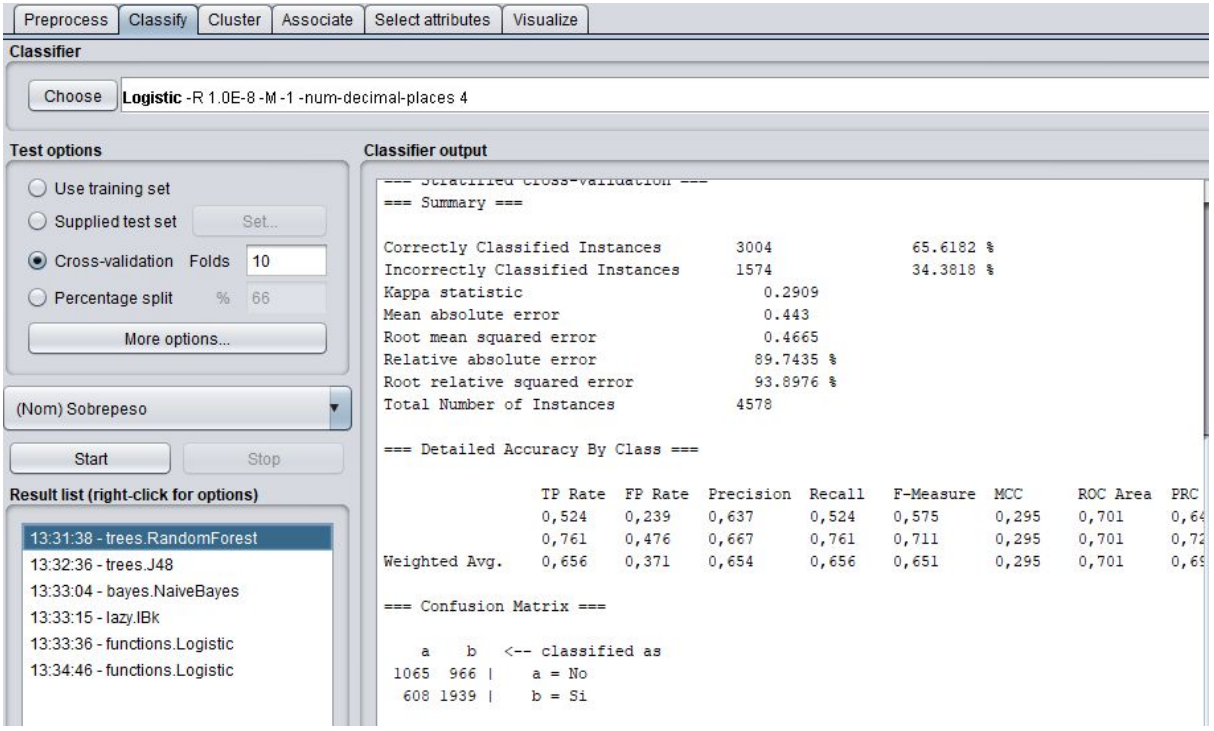



*Figura 26: Weka: Random Forest Classifier*

El desbalance de clases ocurre cuando hay menos instancias de una clase que de otra. Esto afecta a los algoritmos en su proceso de generalización de la información y perjudica a las clases minoritarias.

Para comprobar si este es el problema, se va a realizar un balanceo de clases, con un filtro que proporciona Weka, llamado “ClassBalancer”, cuya funcionalidad es descrita por Weka como volver a ponderar las instancias en los datos para que cada clase tenga el mismo peso total. Se mantendrá la suma total de pesos en todas las instancias. Hecho esto, se ejecuta de nuevo el clasificador Random Forest.

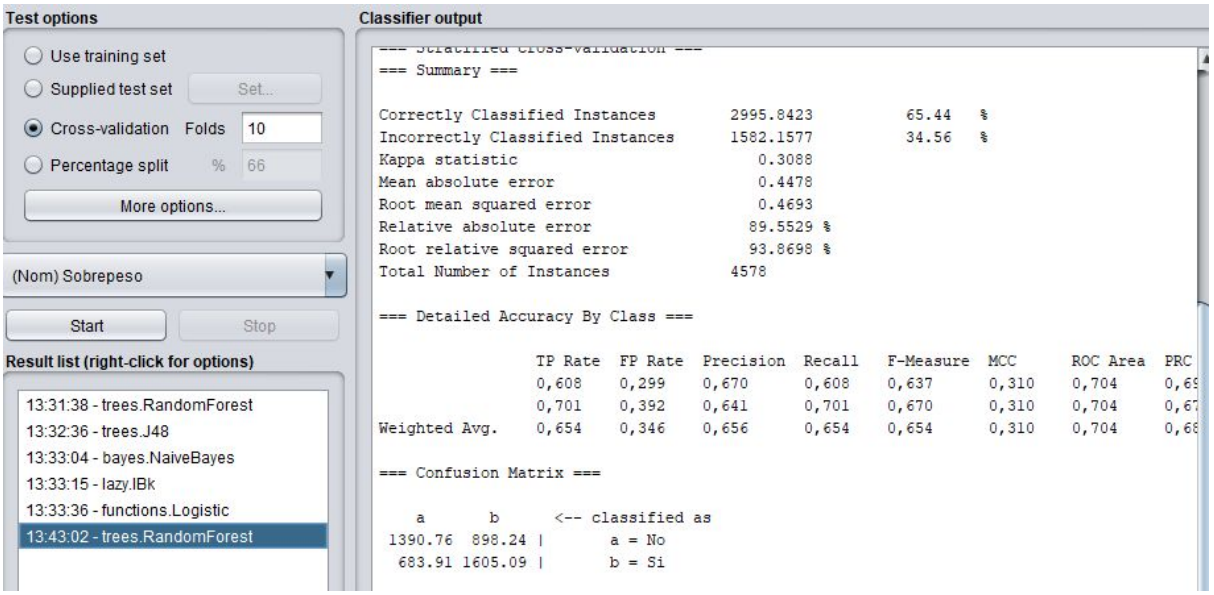



*Figura 27 Weka: Random Forest con clases balanceadas*

Ahora el TP Rate se lleva menos diferencia entre ambas clases, aunque ha disminuido ligeramente el porcentaje de instancias clasificadas correctamente, pero esto no es problema porque es una disminución poco significativa.

En este caso se considera mejor idea dejar las clases desbalanceadas porque el TP rate de los casos positivos de sobrepeso ha disminuido, y esto no interesa, ya que es más importante clasificar los casos de sobrepeso correctamente.

## 4.2 Selección de variables

Una vez obtenidos los resultados de distintos modelos, se entra a utilizar los métodos de selección de variables. Para analizar las diferentes características se ha optado por escoger distintos métodos.

En todas las herramientas se usará el conjunto de datos de las 4578 instancias y 235 variables visto en los apartados anteriores, resultado de eliminar valores ausentes de las encuestas a personas de más de 15 años (Figura 7).

La selección de atributos consiste en la selección del mejor subconjunto de atributos que puede representar el conjunto de datos original. Este proceso persigue reducir el número de características en el tamaño del conjunto de datos, seleccionando las características más

útiles e importantes para el problema a resolver, para mejorar la eficiencia de los procedimientos sin un pérdida significativa en su rendimiento.

Un algoritmo de selección de características se puede ver como una medida de evaluación que puntúa diferentes subconjuntos de características con la combinación de una técnica de búsqueda para sugerir nuevos subconjuntos de características. La selección de la métrica de evaluación afecta significativamente al algoritmo. Los métodos de selección de variables se dividen en tres categorías principales: envolventes (*wrappers*), filtrado (*filter*) e integrados (*embedded*) .

Los métodos integrados forman parte de la herramienta de clasificación por lo que en este trabajo se consideran métodos envolventes y de filtrado. Los métodos de filtrado sólo evalúan las variables de forma individual mientras que los envolvente evalúan conjuntos de variables.

- **Métodos Envolventes**
  A diferencia de los métodos de filtrado, los envolventes evalúan subconjuntos de variables lo que permite detectar posibles interacciones entre variables. Las principales desventajas de estos métodos son:

  - Cuando el número de observaciones es insuficiente, el riesgo creciente de sobreajuste (*overfitting*).
  - Cuando el número de variables es alto, el tiempo de procesamiento se extiende

  Se usa un modelo predictivo o clasificador concreto para evaluar los subconjuntos de variables.

  Los métodos envolventes usados han sido:

  - *Boruta* en R
  - *Recursive Feature Elimination* (RFE) en Python

- **Métodos de filtrado**
  Los métodos de filtrado eligen las variables independientemente del modelo predictivo o clasificador. Se basan sólo en características generales, como la correlación con la variable de respuesta. Los métodos de filtrado eliminan las variables que están menos relacionadas con la variable dependiente. Los métodos de filtro tienden a elegir variables innecesarias, ya que no toman en cuenta las relaciones entre las variables independientes.

  Los procedimientos de filtrado utilizados han sido:
  - Correlación en R
  - *Fast Correlation Based Filter* (FCBF) en Weka
  - *Scatter Search* en Weka

### 4.2.1 En R

#### 4.2.1.1 Metodo de filtrado: Correlación

Es necesario dividir el conjunto de datos entre en variables numéricas y variables categóricas (factores, caracteres,..). Esto resultará en un dataset con 5 variables categóricas (las 4 originales y la variable respuesta creada) y otro dataset con el resto de datos numéricos (229 variables). Esto se hace porque la correlación sólo se puede establecer entre los datos numéricos.

Se usará la correlación para detectar las variables redundantes a través de la función de caret *findCorrelation*(). Una vez detectadas se eliminan del dataset.

```
# datos de tipo numerico
datos_numeric <- datos[ ,!sapply(datos, is.factor)]
glimpse(datos_numeric)

# datos de tipo factor
datos_factor <- datos[ ,sapply(datos, is.factor)]
datos_factor

# 1. Se eliminan variables en el dataset numerico

set.seed(7)
# calculate correlation matrix
correlationMatrix <- cor(datos_numeric)
# summarize the correlation matrix
print(correlationMatrix)
# find attributes that are highly corrected (ideally >0.75)
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.75, exact = TRUE)
# print indexes of highly correlated attributes
print(highlyCorrelated)

# elimino columnas muy correlacionadas
datos_numeric <- datos_numeric[ -c(highlyCorrelated) ]

# Después del filtrado, en el dataset de datos numericos, se añade variable
# respuesta y quedan 160 variables
datos_filter <- cbind(datos_numeric, Estado = datos$Estado)
dim(datos_filter)
```

*Figura 28: R: Eliminación de variables redundantes*

Esto reduce el número de variables en el dataset numérico a 178. Al final se crea el conjunto de datos *datos_filter* y se añade el atributo respuesta para evaluar la importancia de variables en el siguiente método.

#### 4.2.1.2 Método envolvente: Boruta

El método Boruta pertenece a la categoría de métodos envolventes y está construido a partir del algoritmo de clasificación Random Forest. Intenta capturar todas las variables

importantes e interesantes que pueda tener en su conjunto de datos con respecto a una variable de resultado.

El método clasifica las variables en importantes (*important*), no importantes (*unimportant*) y tentativas (*tentative*). Las variables no importantes son las que se eliminarán del conjunto de datos, y para las tentativas se usa la función *TentativeRoughFix*() que decidirá si estas son finalmente importantes o no.

Se ejecuta este método sobre el conjunto de datos resultante del paso anterior más la variable respuesta, llamado *datos_filter* (Figura 28).

```
# 3. METODO DE WRAPPER: BORUTA

trainData <- datos_filter
head(trainData)
set.seed(111)
# Perform Boruta search
boruta_output <- Boruta(Sobrepeso ~ ., data=trainData, doTrace=2, maxRuns=150)
print(boruta_output)

# Plot variable importance
plot(boruta_output, cex.axis=.3, las=2, xlab="", main="Variable Importance")

# ver que variables se han escogido
print(boruta_output)

# Do a tentative rough fix
roughFixMod <- TentativeRoughFix(boruta_output)
print(roughFixMod)
```

*Figura 29: Wrapper Method: Boruta*

En la siguiente gráfica se observan marcados en rojo son las variables no importantes (*unimportant*), en amarillo las variables tentativas (*tentative)*, y en verdes las importantes (*important*). Como ya se mencionó, los de amarillo se escogen con el método TentativeRoughFix()

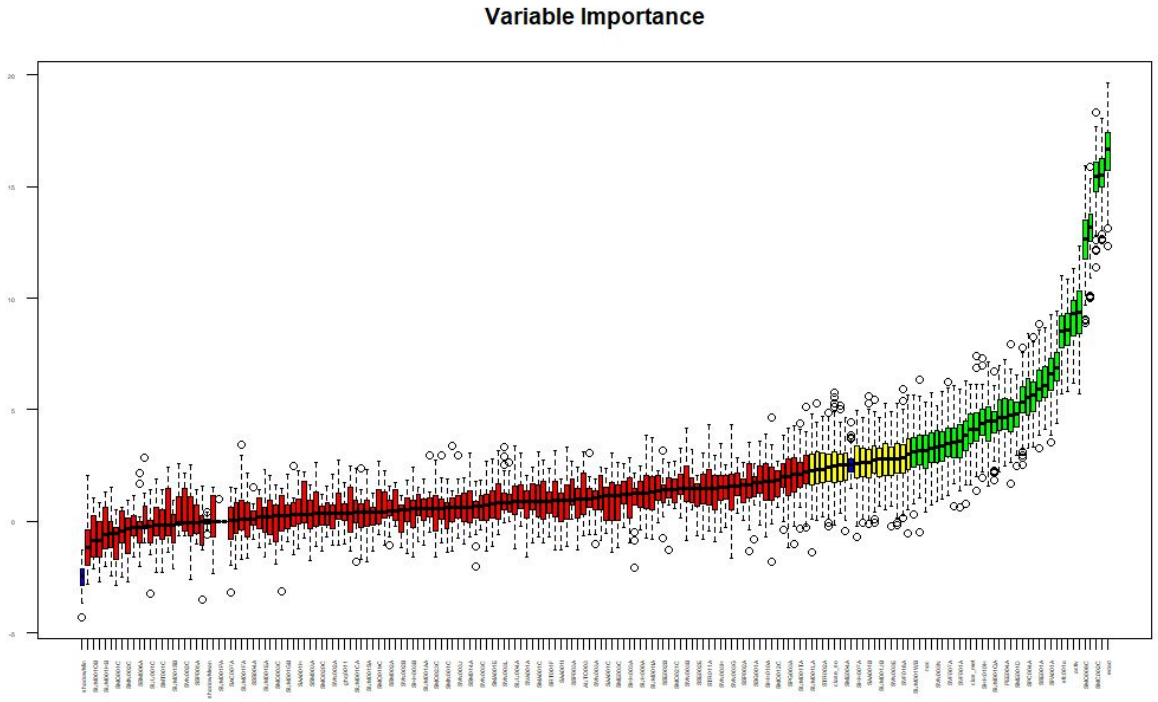



*Figura 30 R: Boruta: Importancia de variables*

```
> print(roughFixMod)
Boruta performed 149 iterations in 1.026681 hours.
Tentatives roughfixed over the last 149 iterations.
 40 attributes confirmed important: activ, clas_met, edad, eva, nac and 35
more;
 138 attributes confirmed unimportant: apoyo, AUTO001, AUTO003, clase_so,
ghq0011 and 133 more;
> # Plot variable importance
> plot(roughFixMod, cex.axis=.3, las=2, xlab="", main="Variable Importance")
> # Get significant variables not including tentatives
> getSelectedAttributes(roughFixMod, withTentative = F)
 [1] "edad"     "sd003a"   "nac"      "sfc001a"  "STR002A"  "PEE004A"  "PEE002A"
 [8] "SPT004A"  "SPC004A"  "SPA001A"  "SPM003A"  "SSE002D"  "SSE004A"  "SSE001A"
[15] "SMC002C"  "SMO008C"  "SMO009C"  "SME001D"  "SME001A"  "SUM001WB" "SUM001KA"
[22] "SUM001QA" "SUM001RB" "SSS001A"  "SSS003A"  "SSP001A"  "SHH001A"  "SHH007A"
[29] "SHH010H"  "SVN003E"  "SVN003N"  "SVF001A"  "SVF012A"  "SVF018A"  "SVF018B"
[36] "SVF007A"  "SVT015A"  "eva"      "clas_met" "activ"
```

*Figura 31: R: Boruta: Atributos seleccionados*

En la figura 31 se observa como el resultado de este algoritmo ha considerado importantes 40 atributos.

**Atributos seleccionados en R**

| Nº | Nombre | Etiqueta | Tipo | Tabla de etiquetas de valor |
|---|---|---|---|---|
| 8 | EDAD | Edad (80 = '80 y más años') | Numérica | |
| 10 | SD003A | Estado civil | Numérica | TB_EST_CIVIL |
| 12 | SFC001A | Nivel de estudios | Numérica | TB_ESTUDIOS |
| 14 | STR002A | Conviven menores | Numérica | TB_SINO_NSNC |
| 26 | PEE004A | Tipo de contrato | Numérica | TB_TIPO_CONTRATO |
| 28 | PEE002A | Horas semanales de trabajo | Numérica | |
| 37 | SPT004A | Frecuencia medición de la tensión arterial | Numérica | TB_TENSION_FREC_A |
| 38 | SPC004A | Frecuencia medición nivel de colesterol | Numérica | TB_COLTR_FREC_A |
| 39 | SPA001A | Frecuencia medición nivel de azúcar | Numérica | TB_AZUCAR_FREC |
| 42 | SPM003A | Motivo mamografía | Numérica | TB_MA_MOTIVO |
| 53 | SSE002D | EQ-5D: Dolor o malestar | Numérica | TB_DOLOR |
| 55 | SSE004A | EQ-5D: EVA | Numérica | |
| 56 | SSE001A | Autovaloración estado salud último año | Numérica | TB_BUENO |
| 153 | SMC002C | Padecido alguna vez: Tensión alta | Numérica | TB_SINO |
| 167 | SMO008C | Padecido alguna vez: Alteraciones menstruación | Numérica | TB_SINO |
| 168 | SMO009C | Padecido alguna vez: Mala circulación, varices en las piernas | Numérica | TB_SINO |
| 185 | SME001D | Padecido últimos 12 meses: Dolores reumáticos; dolores en las articulaciones por artritis o art | Numérica | TB_SINO |
| 226 | SME001A | Diagnosticado: Dolores reumáticos; dolores en las articulaciones por artritis o artrosis | Numérica | TB_SINO |
| 309 | SUM001WB | Recetado: medicinas para el dolor | Numérica | TB_SINO |
| 330 | SUM001KA | Consumido: medicamentos para la tensión arterial | Numérica | TB_SINO |
| 342 | SUM001QA | Consumido: medicamentos para bajar el colesterol | Numérica | TB_SINO |
| 347 | SUM001RB | Recetado: medicamentos para diabéticos | Numérica | TB_SINO |
| 352 | SSS001A | Frecuencia consulta del centro de salud | Numérica | TB_FREC_SERV_SANITARIOS |
| 353 | SSS003A | Calidad de la atención del centro de salud | Numérica | TB_BUENO |
| 370 | SSP001A | Prueba: Análisis de sangre | Numérica | TB_SINO_NSNC |
| 384 | SHH001A | Frecuencia con la que se lava los dientes | Numérica | TB_DENTAL_FREC |
| 386 | SHH007A | Tiempo que hace que acudió al dentista por última vez | Numérica | TB_TIEMPO_DENTISTA |
| 402 | SHH010H | Tiene o conserva todos sus dientes/muelas naturales | Numérica | TB_SINO_NSNC |
| 426 | SVN003E | Pasta, arroz y papas | Numérica | TB_ALIMENTO_FREC |
| 435 | SVN003N | Snacks salados | Numérica | TB_ALIMENTO_FREC |
| 438 | SVF001A | Actividad física en el trabajo | Numérica | TB_FISICA_TRAB |
| 440 | SVF012A | Días actividad física intensa | Numérica | |
| 449 | SVF018A | Horas actividad física sentado | Numérica | |
| 450 | SVF018B | Minutos actividad física sentado | Numérica | |
| 451 | SVF007A | Horas que duerme | Numérica | |
| 476 | SVT015A | Frecuencia de exposición al humo del tabaco dentro de casa | Numérica | TB_EXPUESTO_HUMO_TABACO |
| 527 | EVA | Puntuación Escala Visual Analógica | Numérica | |
| 531 | CLAS_MET | Clasificación de nivel de actividad | Numérica | TB_IPAQ |
| 532 | ACTIV | Relación con la actividad | Numérica | TB_ACTIV |

*Figura 32: R: Descripción de atributos seleccionados*

En cuanto a las variables relacionadas con la actividad física, este procedimiento escoge las siguientes 6 variables:

- SVF001A: Actividad física en el trabajo
- SVF012A: Días actividad física intensa
- SVF018A: Horas actividad física sentado
- SVF018B: Minutos actividad física sentado
- SVF007A: Horas que duerme
- CLAS_MET: Clasificación de nivel de actividad

Son 6 en total, las 5 primeras son de tipo numérico, y la otra es de tipo numérico ordinal, siendo esta su interpretación:

**TB_IPAQ** (Para la variable CLAS_MET)
0 = 'Nivel bajo o inactivo'
1 = 'Nivel moderado'
2 = 'Nivel alto';

*Figura 33: Etiqueta de valor de la variable CLAS_MET*

Finalmente, se añaden al conjunto de 40 variables, las variables de tipo factor que se habían puesto en un dataset aparte.

### 4.2.2 En Python

En python se ha optado por el método RFE (*Recursive Feature Elimination*) que es un método de selección de variables envolvente, que se adapta a un modelo y elimina las características más débiles hasta que se alcanza el número especificado de variables.

El modelo que se usa es el Random Forest, y el número de variables elegido ha sido de 20.

```
# Recursive Feature elimination (RFE)
from sklearn.feature_selection import RFE
# Create the RFE object and rank each pixel
clf_rf_3 = RandomForestClassifier()
rfe = RFE(estimator=clf_rf_3, n_features_to_select=20, step=1)
rfe = rfe.fit(X_train, Y_train)
```

```
print('Chosen best features by rfe:',X_train.columns[rfe.support_])
```

```
Chosen best features by rfe: Index(['clase_so', 'edad', 'sd002c', 'sfc001a', 'sc015a', 'SIH001A', 'SPA001A',
       'SPI002A', 'SSE004A', 'SHH007A', 'SHH003A', 'SVN003K', 'SVF018A',
       'SVT010A', 'SVA001A', 'peso', 'hs_tarea', 'ghq0011', 'eva', 'apoyo'],
      dtype='object')
```

*Figura 34: Python: RFE*

**Atributos seleccionados en Python**

| Nº | Nombre | Etiqueta | Tipo | Tabla de etiquetas de valor |
|---|---|---|---|---|
| 7 | CLASE_SO | Clase social | Numérica | TB_CLASE_SOCIAL |
| 8 | EDAD | Edad (80 = '80 y más años') | Numérica | |
| 9 | SD002C | Año de nacimiento (1935 = '1935 o anterior') | Numérica | |
| 12 | SFC001A | Nivel de estudios | Numérica | TB_ESTUDIOS |
| 13 | SC015A | Tipo de hogar | Numérica | TB_TIPO_HOGAR |
| 34 | SIH001A | Ingresos mensuales netos del hogar | Numérica | TB_INGRESOS_HOGAR |
| 39 | SPA001A | Frecuencia medición nivel de azúcar | Numérica | TB_AZUCAR_FREC |
| 43 | SPI002A | Frecuencia citología | Numérica | TB_MA_CI_FREC |
| 55 | SSE004A | EQ-5D: EVA | Numérica | |
| 386 | SHH007A | Tiempo que hace que acudió al dentista por última vez | Numérica | TB_TIEMPO_DENTISTA |
| 387 | SHH003A | Motivo de la última consulta al dentista (respuesta 1) | Numérica | TB_DENTAL_MOTIVO |
| 431 | SVN003K | Dulces | Numérica | TB_ALIMENTO_FREC |
| 449 | SVF018A | Horas actividad física sentado | Numérica | |
| 460 | SVT010A | Edad en que comenzó a fumar de forma habitual | Numérica | |
| 477 | SVA001A | Frecuencia de consumo de alcohol | Numérica | TB_ALCOHOL_FREC |
| 513 | PESO | Ponderación | Numérica | |
| 516 | HS_TAREA | Horas semanales promedio dedicadas a las tareas del hogar | Numérica | |
| 520 | GHQ0011 | Puntuación GHQ-12 | Numérica | |
| 527 | EVA | Puntuación Escala Visual Analógica | Numérica | |
| 528 | APOYO | Puntuación Cuestionario Duke-UNC | Numérica | |

*Figura 35: Python: Descripción de atributos seleccionados*

Esta vez se escoge tan sólo una variable relacionado con la actividad física:

- SVF018A: Horas actividad física sentado

Si se eleva el número de variables a seleccionar en RFE a 40, aumenta a 4, añadiendo las siguientes:

- SVF016A: Días actividad física caminar
- SVF017A: Horas actividad física caminar
- SVF007A: Horas que duerme

### 4.2.3 En Weka

El conjunto de datos que se carga en Weka es el de 235 variables y 4578 instancias de mayores de 15 años, que incluye la variable de respuesta.

En Weka, para seleccionar variables hay que ir a la pestaña “Select Attributes” y aparecen varios métodos. Se utiliza el método de búsqueda de filtrado llamado FCBF : *Fast Correlation Based Filter*, y como evaluador de atributos *SymmetricalUncertAttributeSetEval*, que evalúa el valor de un conjunto de atributos midiendo la incertidumbre simétrica con respecto a otro conjunto de atributos [16].

El algoritmo FCBF consta de dos etapas: la primera es un análisis de relevancia, destinado a ordenar las variables de entrada en función de una puntuación de relevancia, que se calcula como la incertidumbre simétrica con respecto a la variable de respuesta. Esta etapa también se usa para descartar variables irrelevantes, que son aquellas cuyo puntaje de clasificación está por debajo de un umbral predefinido. La segunda etapa es un análisis de redundancia, destinado a seleccionar variables predominantes del conjunto relevante obtenido en la primera etapa.

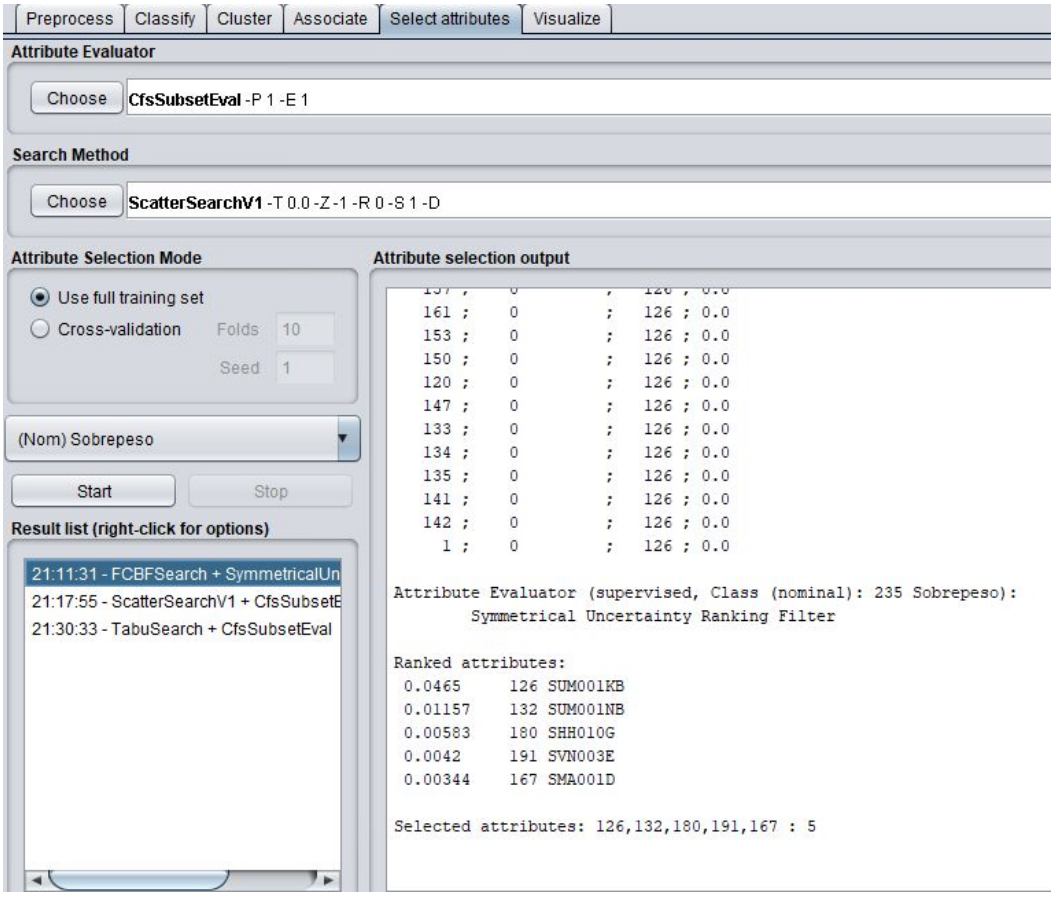



*Figura 36: Weka: Feature Selection con FCBFSearch*

El algoritmo ha clasificado 5 atributos como más importantes:

- SUM001KB: Recetado: medicamentos para la tensión arterial
- SUM001NB: Recetado: píldoras para no quedar embarazada
- SHH010G: Faltan dientes o muelas sin sustituir
- SVN003E: Consumo Pasta, arroz y papas
- SMA001D: Consulta a otros profesionales de medicina alternativa

No se llega a considerar ninguna de las variables relacionadas con la actividad física como importante, así que se usará otro método de selección de variables llamado Scatter Search [17], y como evaluador de atributos *CfsSubSetEval* [18], que evalúa el valor de un subconjunto de atributos al considerar la capacidad predictiva individual de cada característica junto con el grado de redundancia entre ellas.

Scatter Search comienza con una población de soluciones de las cuales se selecciona un subconjunto de soluciones para el set de referencia RefSet que evoluciona mediante mecanismos de intensificación y diversificación. Las soluciones de este conjunto se combinan para generar nuevas soluciones para actualizar el conjunto de referencia. La combinación de la solución en Scatter Search es guiada y no aleatoria, a diferencia de los algoritmos genéticos. Otra diferencia importante es que el conjunto de referencia que evoluciona en el Scatter Search es más pequeño que la población habitual de los algoritmos evolutivos.

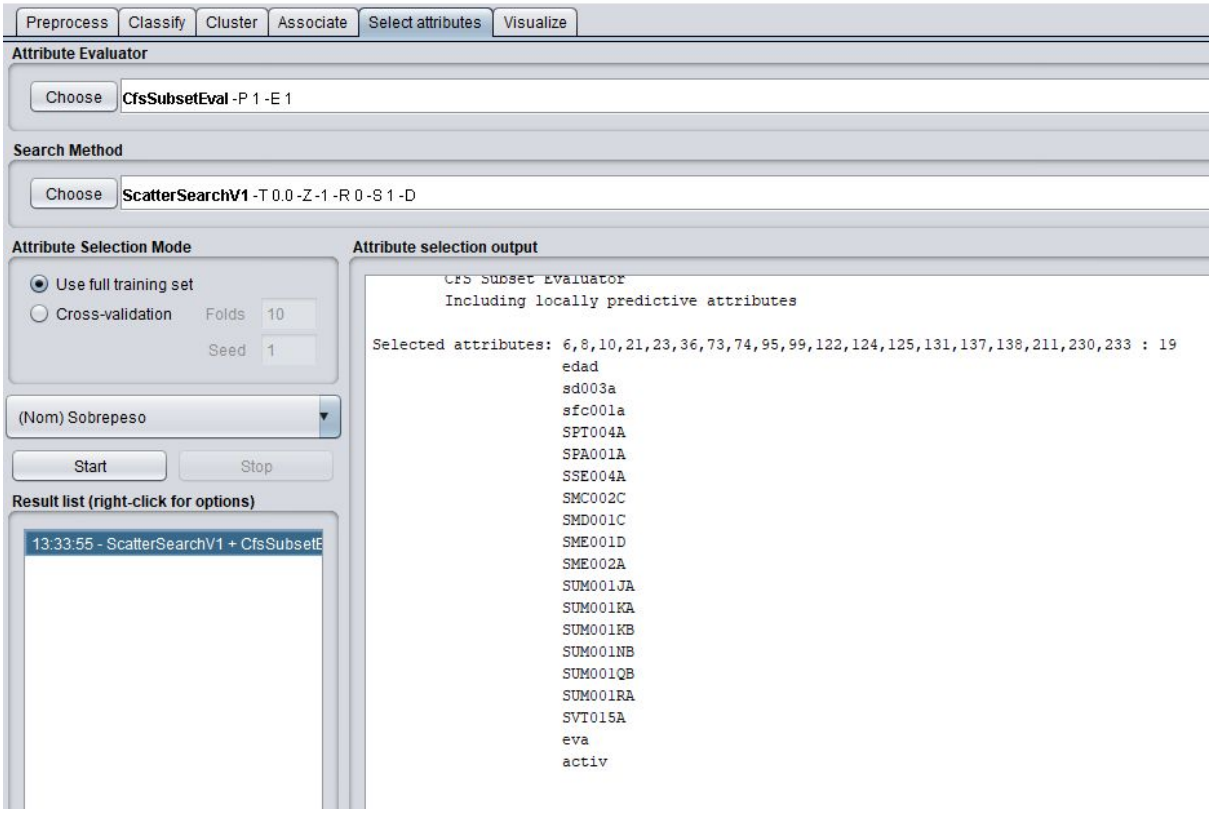



*Figura 37: Weka: Feature Selection con Scatter Search*

Se escogen 19 atributos, cuya descripción es la siguiente:

**Atributos seleccionados en Weka**

| Nº | Nombre | Etiqueta | Tipo | Tabla de etiquetas de valor |
|---|---|---|---|---|
| 8 | EDAD | Edad (80 = '80 y más años') | Numérica | |
| 10 | SD003A | Estado civil | Numérica | TB_EST_CIVIL |
| 12 | SFC001A | Nivel de estudios | Numérica | TB_ESTUDIOS |
| 37 | SPT004A | Frecuencia medición de la tensión arterial | Numérica | TB_TENSION_FREC_A |
| 39 | SPA001A | Frecuencia medición nivel de azúcar | Numérica | TB_AZUCAR_FREC |
| 55 | SSE004A | EQ-5D: EVA | Numérica | |
| 153 | SMC002C | Padecido alguna vez: Tensión alta | Numérica | TB_SINO |
| 154 | SMD001C | Padecido alguna vez: Diabetes o azúcar en sangre | Numérica | TB_SINO |
| 185 | SME001D | Padecido últimos 12 meses: Dolores reumáticos; dolores en las articulaciones por artritis o art | Numérica | TB_SINO |
| 227 | SME002A | Diagnosticado: Dolor de espalda por lumbago, ciática o hernia discal | Numérica | TB_SINO |
| 328 | SUM001JA | Consumido: medicamentos para el corazón | Numérica | TB_SINO |
| 330 | SUM001KA | Consumido: medicamentos para la tensión arterial | Numérica | TB_SINO |
| 331 | SUM001KB | Recetado: medicamentos para la tensión arterial | Numérica | TB_SINO |
| 337 | SUM001NB | Recetado: píldoras para no quedar embarazada | Numérica | TB_SINO |
| 343 | SUM001QB | Recetado: medicamentos para bajar el colesterol | Numérica | TB_SINO |
| 346 | SUM001RA | Consumido: medicamentos para diabéticos | Numérica | TB_SINO |
| 476 | SVT015A | Frecuencia de exposición al humo del tabaco dentro de casa | Numérica | TB_EXPUESTO_HUMO_TABACO |
| 527 | EVA | Puntuación Escala Visual Analógica | Numérica | |
| 532 | ACTIV | Relación con la actividad | Numérica | TB_ACTIV |

*Figura 38: Weka: Descripción de atributos seleccionados*

En esta ocasión, tampoco se considera importante a ningún atributo que da información sobre la actividad física.

## 4.3 Comparación de clasificador después de selección de variables

En este apartado se volverán a aplicar los clasificadores vistos anteriormente en el conjunto de datos en el que se aplicó selección de variables en cada una de las herramientas para comprobar los cambios producidos en los distintos indicadores del rendimiento de cada uno de ellos.

### 4.3.1 En R

En primer lugar, hay que volver a dividir el conjunto de datos resultante de la selección de variables, en entrenamiento y test. Luego se procede a aplicar los clasificadores sobre el conjunto de datos de entrenamiento. Las figuras 39 a 43 muestran los resultados obtenidos con los distintos clasificadores.

- K Nearest Neighbors

```
k-Nearest Neighbors

3663 samples
  44 predictor
   2 classes: 'No', 'Si'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3296, 3296, 3298, 3297, 3296, 3297, ...
Resampling results across tuning parameters:

  k    Accuracy   Kappa
    7  0.6248986  0.2321981
   50  0.6314440  0.2312483
  100  0.6287237  0.2216625

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 50.
```

*Figura 39: R: Clasificador KNN después de FS*

- Regresión Logística

```
> model_lg_FS
Generalized Linear Model

3663 samples
  44 predictor
   2 classes: 'No', 'Si'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3296, 3296, 3298, 3297, 3296, 3297, ...
Resampling results:

  Accuracy   Kappa
  0.6319808  0.2537353
```

*Figura 40: R: Clasificador Regresión Logística después de FS*

- Naïve Bayes

```
Naive Bayes

3663 samples
  44 predictor
   2 classes: 'No', 'Si'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3296, 3297, 3297, 3296, 3296, 3297, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.4640974  0.02784063
   TRUE      0.5563751  0.00000000

Tuning parameter 'laplace' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were laplace = 0, usekernel = TRUE and adjust = 1.
```

*Figura 41: R: Clasificador Naive Bayes después de FS*

- Árbol de decisión

```
CART

3663 samples
  44 predictor
   2 classes: 'No', 'Si'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3296, 3296, 3298, 3297, 3296, 3297, ...
Resampling results across tuning parameters:

  cp          Accuracy   Kappa
  0.02892308  0.6175327  0.2238548
  0.04984615  0.6049688  0.1823264
  0.06430769  0.5864119  0.1182849

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.02892308.
```

*Figura 42: R: Clasificador CART después de FS*

- Random Forest

```
Random Forest

3663 samples
  44 predictor
   2 classes: 'No', 'Si'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3296, 3296, 3298, 3297, 3296, 3297, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
    2   0.5569215  0.001367723
  205   0.6664019  0.315432616
  408   0.6650268  0.313828287

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 205.
```

*Figura 43: R: Clasificador Random Forest después de FS*

Se puede ver que se ha mantenido la precisión de los clasificadores en todos menos en Naive Bayes, que disminuye un 7%, y en KNN aumenta un 3%.

### 4.3.2 En Python

Se vuelven a aplicar los clasificadores, esta vez en el dataset en el que se aplicó el RFE con 40 variables:

```
# Spot Check Algorithms
models = []
models.append(('KNN', KNeighborsClassifier(n_neighbors=100)))
models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
models.append(('NB', GaussianNB()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('RF', RandomForestClassifier(n_estimators=100)))
# evaluate each model in turn
results = []
names = []
for name, model in models:
  kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
  cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy')
  results.append(cv_results)
  names.append(name)
  print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
```

```
KNN: 0.591755 (0.021256)
LR: 0.645290 (0.021517)
NB: 0.615525 (0.019426)
CART: 0.567445 (0.032470)
RF: 0.663032 (0.025548)
```

*Figura 44: Python: Precisión de modelos después de la selección de variables*

Tras la selección de variables, prácticamente se mantiene la misma precisión de los algoritmos, al igual que ocurría en R, con lo cual la selección de variables ha sido realizada con éxito. El único que ha aumentado la precisión es el Naive Bayes, un 3 %.

### 4.3.3 En Weka

Para el caso del dataset en el que se usó FCBF, se han escogido los mismos clasificadores que antes de la selección de variables, esta vez ejecutado sobre el dataset de 6 variables (las 5 elegidas y la respuesta).

Este es el porcentaje de Instancias clasificadas correctamente:

- K Nearest Neighbors (k=150): 60.6597 %
- Naive Bayes: 60.1573 %
- Regresión Logística: 61.3368 %
- J48 (Árbol de decisión): 61.0529 %
- Random Forest: 61.8174 %

El porcentaje ha disminuido en algunos modelos, pero aumentado en otros, aunque no son porcentajes determinantes. Se observa que se ha reducido el dataset de 235 variables a 6 variables y sólo pierde, en el peor de los casos, algo más de un 3% de instancias clasificadas correctamente.

Para el dataset resultante de Scatter Search, aplicar los clasificadores sobre este tiene la siguiente salida:

- K Nearest Neighbors (k=150): 65.5526 %
- Regresión Logística: 64.9629 %
- Naive Bayes: 61.2931 %
- J48 (Árbol de decisión): 61.2931 %
- Random Forest: 65.3124 %

## 4.4 Predicciones

Para realizar las predicciones sobre los datos de test, se escogerá el modelo con mayor precisión.

### 4.4.1 En R

En R, los mejores resultados se han obtenido con el Random Forest, así que se procede a realizar la predicción con este clasificador.

```
# Predicción del Random Forest

model_knn_FS <- readRDS( "C://Users//Carmen//Desktop//TFG//Modelos//Modelos//model_knn_FS")
prediction_knn <- predict(model_knn_FS, test_set)
str(prediction_knn)
head(prediction_knn)

confusionMatrix(data = prediction_knn, test_set$Sobrepeso)
```

*Figura 45: R: Código para predecir con Random Forest*

```
Confusion Matrix and Statistics

          Reference
Prediction  No  Si
        No 227 116
        Si 179 393

               Accuracy : 0.6776
                 95% CI : (0.6462, 0.7078)
    No Information Rate : 0.5563
    P-Value [Acc > NIR] : 4.354e-14

                  Kappa : 0.3365

 Mcnemar's Test P-Value : 0.0003065

            Sensitivity : 0.5591
            Specificity : 0.7721
         Pos Pred Value : 0.6618
         Neg Pred Value : 0.6871
             Prevalence : 0.4437
         Detection Rate : 0.2481
   Detection Prevalence : 0.3749
      Balanced Accuracy : 0.6656

       'Positive' Class : No
```

*Figura 46: R: Resultados predictivos*

Usando el Random Forest como predictor, se obtiene una precisión del 67%.

Entre los indicadores que se muestran, se encuentran:

- Matriz de Confusión (*Confusion Matrix*): es una tabla que describe el rendimiento de un modelo supervisado de clasificación en los datos de prueba, donde se desconocen los verdaderos valores. La diagonal principal describe las instancias clasificadas correctamente, los otros valores son instancias clasificadas incorrectamente.

- Sensibilidad (*Sensitivity*): La sensibilidad o *recall*, es la proporción de personas que se identifican correctamente por no tener sobrepeso, es decir verdadero positivo, sobre el número total de personas que no tienen sobrepeso.

- Especificidad (*Specificity*): La especificidad, es la proporción de personas que se identifican correctamente por tener sobrepeso, verdadero negativo, sobre el número total de personas que realmente tienen sobrepeso.

### 4.4.2 En Python

Si se comparan los distintos clasificadores que se usaron en pasos anteriores, se puede observar que el Random Forest es superior al resto en términos de precisión.

```
# Compare Algorithms
plt.boxplot(results, labels=names)
plt.title('Algorithm Comparison')
plt.show()
```

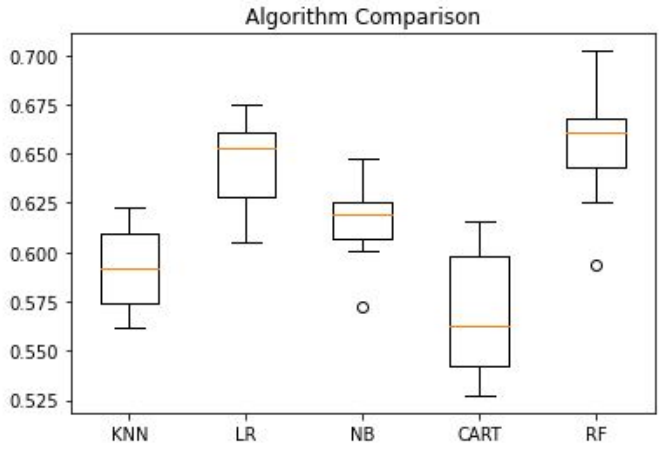



*Figura 47: Python: Comparación de algoritmos*

Si se usa el Random Forest como predictor, se obtiene lo siguiente:

▾ Hacer las predicciones

```
[55] # Make predictions on validation dataset
     # se usa el algoritmo RF porque es el que mejor resultados
     model = RandomForestClassifier(n_estimators=100)
     model.fit(X_train, Y_train)
     predictions = model.predict(X_validation)
```

▾ Evaluar las predicciones

```
[56] # Evaluate predictions
     print(accuracy_score(Y_validation, predictions))
     print(confusion_matrix(Y_validation, predictions))
     print(classification_report(Y_validation, predictions))
```

```
0.7041484716157205
[[240 154]
 [117 405]]
              precision    recall  f1-score   support

          No       0.67      0.61      0.64       394
          Si       0.72      0.78      0.75       522

    accuracy                           0.70       916
   macro avg       0.70      0.69      0.69       916
weighted avg       0.70      0.70      0.70       916
```

*Figura 48: Python: Predicción con Random Forest*

Las métricas que aparecen en la salida que aún no se han descrito son:

- F1-score: se puede interpretar como un promedio ponderado de la precisión y la sensibilidad. La fórmula es:

```
F1 = 2 * (precisión * sensibilidad) / (precisión + sensibilidad)
```

- Soporte (*Support*): número de instancias de cada clase

La clasificación tiene éxito en las dos clases, ya que logra clasificar correctamente a ambas en más del 67% de las ocasiones. Además, la precisión general de este clasificador es del 70%.

### 4.4.3 En Weka

Los resultados predictivos de Weka son los del apartado 4.6.3, ya que en Weka, en la validación cruza de 10 capas, al dividir el conjunto de datos en 10, guarda uno de estos como conjunto de test, que luego aplica automáticamente, y por eso se dispone de la matriz de confusión, TP Rate, FP Rate, recall, etc (Figura 27) .

## 4.5 Importancia de los predictores actividad física

Para comprobar la importancia de las variables que contienen información sobre la actividad física para predecir la variable respuesta, se ha realizado un ranking en el dataset que contiene estas 9 variables que aparecen en la figura 10, mediante la función varImp() de caret y el algoritmo Random Forest.

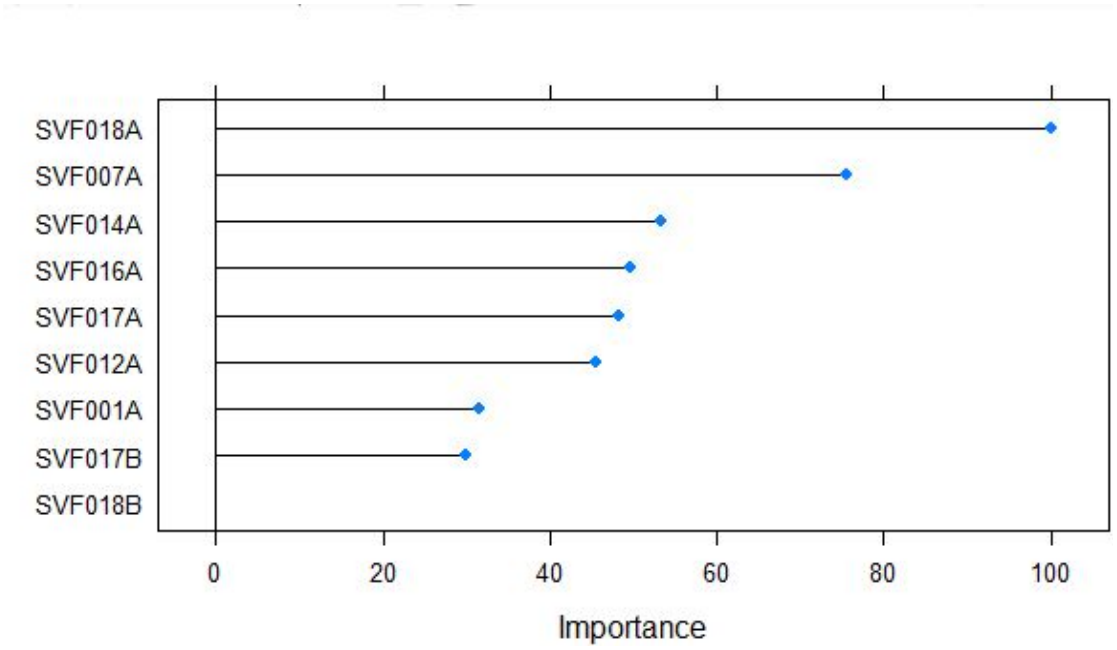



*Figura 49: Importancia de variables*

Se observa que la más significativa es SVF018A, que aporta el número de horas que se pasa sentado, seguido de SVF007A, que expone el número de horas de sueño.

Además, si se aplican los algoritmos para calcular la capacidad predictiva de este dataset sobre la variable respuesta, se obtienen los siguientes resultados:

```
LR: 0.562269 (0.019813)
KNN: 0.523473 (0.027832)
CART: 0.525669 (0.026325)
NB: 0.565826 (0.020040)
RF: 0.534413 (0.020973)
```

*Figura 50: Resultados de aplicar clasificador al dataset de actividad física*

Estos resultados distan en un aproximadamente un 10% con respecto a las predicciones realizadas con las demás variables, por lo que se puede decir que, si bien las variables relacionadas con la actividad física dan información de valor, no son las más significativas para predecir el objetivo.

Por último, si se verifica la importancia de las variables dentro del dataset finalmente escogido en R, el que ha pasado por el preprocesado y selección de variables, en el top 20 de características más importantes, se encuentran cuatro de actividad física.

```
rf variable importance

  only 20 most important variables

         Overall
edad      100.00
SSE004A    55.07
sfc001a    43.99
SVF018A    43.85
SHH007A    39.91
SMC002C    34.19
SVF007A    33.68
SPA001A    32.54
sd003a     31.23
SVT015A    30.42
SSS003A    29.23
SPT004A    28.19
SVN003E    26.96
SVN003N    26.49
SHH001A    26.19
SSE001A    25.08
SUM001KA   24.46
SVF012A    23.52
PEE002A    22.97
SVF001A    21.93
```

*Figura 51: Importancia de variables en el dataset final de R*

Estas son las comentadas anteriormente, SVF018A, SVF007A, SVF012A, SVF001A, cuya descripción se encuentra en la figura 5.

# CAPÍTULO 5: Conclusiones

## 5.1 Importancia de la actividad física

A través del estudio de todas las variables que forman parte del conjunto de datos de la Encuesta de Salud Canaria de 2015, relacionándolas con el IMC, puede verificarse la importancia que tienen distintas variables en relación al sobrepeso. Para ello se han usado herramientas informáticas comunes para establecer clasificadores que intentan predecir el sobrepeso usando las respuestas a las preguntas de la encuesta. Se ha conseguido la creación de un modelo predictivo en Python, que es capaz de predecir los casos de sobrepeso con una precisión del 70% (Figura 48).

Se ha verificado que, las características que describen la actividad física aportan información importante para la predicción de los modelos, pero no son tan determinantes como otras, como por ejemplo, la edad, patologías relacionadas con la tensión arterial y nivel de azúcar/diabetes, el nivel de estudios y la Escala Visual Analógica, que permite medir el dolor de cada individuo.

Entre los atributos que describen la actividad física y deportiva los más influyentes han resultado ser: las horas al día que se pasan sentado y el número de horas de sueño, lo cual contrapone el artículo [3] comentado en el Capítulo 2 de este informe, publicado por la Gaceta Sanitaria, donde se expone que el sueño no parece guardar relación con el IMC. En la figura 52 se observa que las personas con sobrepeso tienden a dormir menos horas, en contraposición a las personas que no tienen sobrepeso.

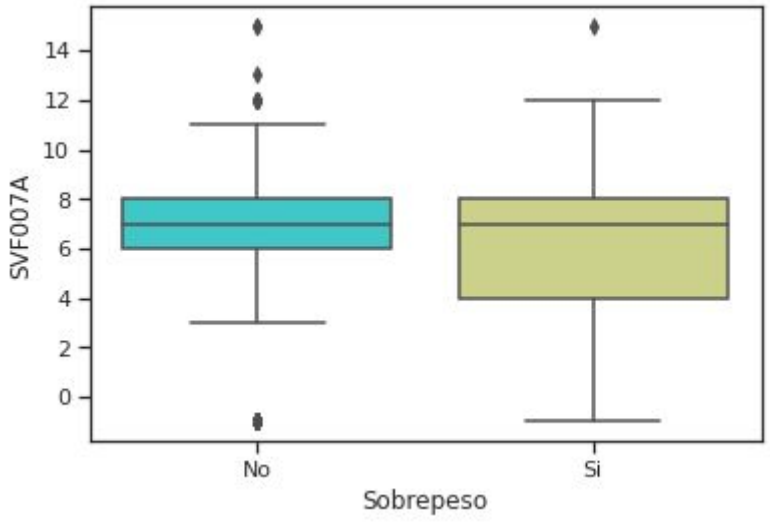


*Figura 52: Relación del sobrepeso y las horas de sueño*

## 5.2 Herramientas

Durante el proyecto se ha trabajado con tres herramientas comunes en informática: R, Python y Weka. Se han observado algunas diferencias significativas entre ellas.

- En R, con el paquete *caret* se usa la misma función para crear los modelos predictivos, *train*(), sin embargo, en Python, con *scikit-learn,* cada clasificador tiene su función, a la que únicamente hay que llamar con los parámetros deseados. Al contrario que ambas, en Weka simplemente hay que ir a la pestaña de *Classify* y escoger el clasificador que se desee, pudiendo cambiar los valores de los distintos parámetros.

- Weka es la herramienta que se ha encontrado más fácil de utilizar, intuitiva y rápida de las tres opciones. Es bastante potente y se aprende mucho debido a lo anteriormente mencionado, su facilidad de uso. Sin embargo, es menos flexible que las otras opciones.

- R ha sido con mucha diferencia la más lenta y menos intuitiva, la creación de modelo tardaba mucho más que en Python y Weka. A su favor, cuenta con mucho apoyo de la comunidad y una infinidad de librerías para crear modelos, a parte de *caret*.

- Python ha sido la herramienta en la que más información y ayuda había de la comunidad, lo que favoreció el entendimiento de los procedimientos, además, esta no era del todo desconocida, ya que es un lenguaje de programación con el que se había trabajado ya.

Si se tuviese que elegir una de ellas para un estudio práctico en profundidad, Python es la opción que se encuentra más apropiada, por su rapidez, facilidad de uso y por que hay multitud de documentación y plataformas como *Kaggle,* donde es el lenguaje que más se usa para el análisis y tratamiento de datos.

## 5.3 Selección de variables

Durante la selección de variables se ha observado que hay distintos métodos y algoritmos que se pueden aplicar para conseguir un subconjunto óptimo de variables, pudiendo reducir el conjunto de datos hasta 10 veces, como fue el caso de Weka, con el FCBF (*Fast Correlation Based Filter*), sin llegar a perder gran capacidad predictiva empleando el mismo

clasificador. Esto es un indicador de que la selección de variables ha funcionado correctamente.

Además, se han comparado los subconjuntos escogidos en las distintas herramientas, y en todos han aparecido variables relacionadas con la edad, nivel de estudios, Escala Visual Analógica y actividades preventivas, como la medición del nivel de azúcar o tensión. Aunque no se ha escogido exactamente el mismo subconjunto de variables, se seleccionan bastantes en común en las tres herramientas. Las diferencias se deben a que en cada herramienta se usaron distintos métodos de selección de variables.

## 5.4 Clasificadores

Se ha comprobado que se obtiene prácticamente la misma precisión en la capacidad predictiva de los algoritmos en todas las herramientas utilizadas: R, Python y Weka.

Utilizando Weka, se ha evaluado la eficiencia de todos los clasificadores utilizados, mediante la curva ROC (acrónimo de *Receiver Operating Characteristic*), la cual establece con qué tasa de éxito un modelo puede distinguir entre dos clases. El AUC (*Area Under the Curve*) es el área bajo la curva ROC y su valor establece cómo de bueno es el modelo.

Al visualizar estas curvas para los clasificadores, se ha confirmado que el Random Forest y la Regresión Logística son los clasificadores con mejor puntaje AUC, rozando el 0.7, lo que indica que hay casi un 70% de probabilidad de que el modelo pueda distinguir correctamente entre clase positiva y clase negativa (figuras 53 y 54). Cuando se aplican estos clasificadores sobre el conjunto de datos, se verifica que son los que mejor rinden de todos los considerados, ya que clasifican instancias con mayor ratio de éxito.

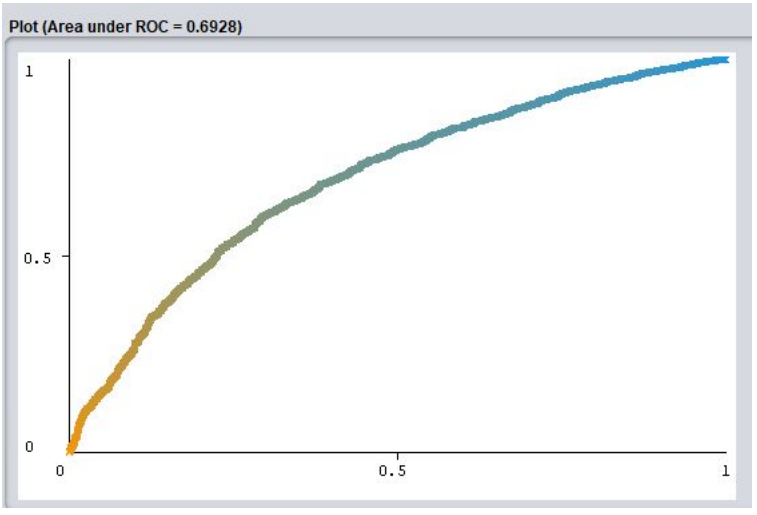


*Figura 53: Curva ROC del clasificador Random Forest*

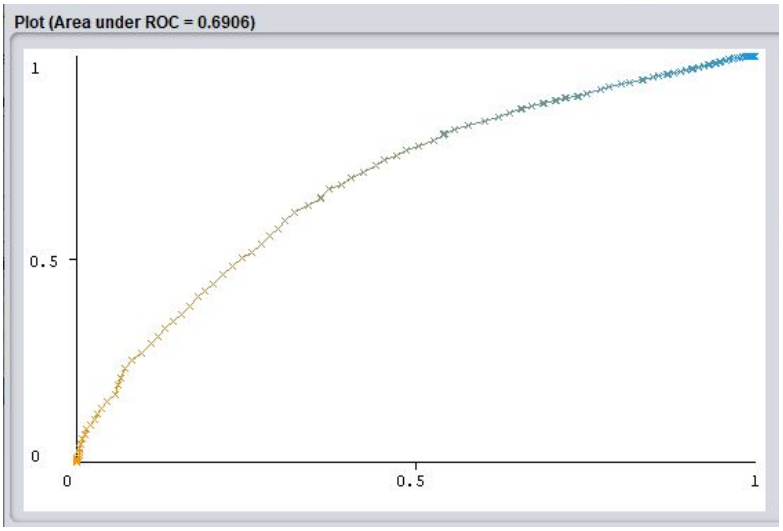



*Figura 54: Curva ROC del clasificador Regresión Logística*

En la tabla incluida en la figura 55 se puede comprobar la precisión (*accuracy*) de los clasificadores usados en todas las herramientas. La etiqueta “sin SV” significa antes de la selección de variables, y “con SV”, después de la selección. En la primera columna figuran las etiquetas de los algoritmos usados siendo KNN el clasificador de los k Vecinos más cercanos, RL el de Regresión Logística, NB el Naive Bayes, AD el Árbol de Decisión, y RF el Random Forest.

| | R | | Python | | Weka | |
|---|---|---|---|---|---|---|
| | sin SV | con SV | sin SV | con SV | sin SV | con SV |
| KNN | 60.52570% | 63.14440% | 59.9675% | 59.1755% | 63.6522% | 65.5526% |
| RL | 64.67275% | 63.19898% | 64.0089% | 64.5260% | 65.1376% | 64.9629% |
| NB | 62.60064% | 55.63751% | 57.9198% | 61.5525% | 61.0092% | 61.2931% |
| AD | 61.09895% | 61.75327% | 0.580016% | 56.7445% | 57.6234% | 61.2931% |
| RF | 66.83033% | 66.64019% | 65.5124% | 66.3032% | 65.6182% | 65.3124% |

*Figura 55: Tabla con el accuracy de los clasificadores en las 3 herramientas*

# Apéndice

## Enlace al código en R

https://drive.google.com/file/d/14JTX1q7y5j5i5jBVxIxssGroVIlkSPci/view

## Enlace al código en Python

https://colab.research.google.com/drive/1ROjgaaH2rBwIpcj_QLABE1c0N6limQ3s#scrollTo=pwO20A7b-fki

# Referencias


[1] Martín Cervantes, P. A., Rueda López, N., & Cruz Rambaud, S. (2019). A Causal Analysis of Life Expectancy at Birth. Evidence from Spain. *International Journal of Environmental Research and Public Health*, *16*(13), 2367. doi:10.3390/ijerph16132367

[2] Organización Mundial de la Salud. (2018). *Life expectancy and Healthy life expectancy. Data by country.* Recuperado de Global Health Observatory data repository: https://apps.who.int/gho/data/node.main.688

[3] Martínez-Moyá, M., Navarrete-Muñoz, E. M., García de la Hera, M., Giménez-Monzo, D., González-Palacios, S., Valera-Gran, D., Sempere-Orts, & M., Vioque, J. (2014). Asociación entre horas de televisión, actividad física, horas de sueño y exceso de peso en población adulta joven. *Gaceta Sanitaria*, 28(3), 203-208.

[4] Hernández-Yumar, A., Abásolo Alessón, I., & López-Valcárcel, B. G. (2019). Economic crisis and obesity in the Canary Islands: an exploratory study through the relationship between body mass index and educational level. *BMC Public Health* 19, 1755. https://doi.org/10.1186/s12889-019-8098-x

[5] Nilsson, N. J. (2014). *Principles of artificial intelligence*. Morgan Kaufmann.

[6] Agarwal, R., & Dhar, V. (2014). Big Data, Data Science, and Analytics: The Opportunity and Challenge for IS Research. *Information Systems Research*, *25*(3), 443-448.

[7] Consoli, S., Recupero, D. R., & Petkovic, M. (2019). *Data Science for Healthcare*. Springer International Publishing.

[8] Sivarajah, U., Kamal, M. M., Irani, Z., & Weerakkody, V. (2017). Critical analysis of Big Data challenges and analytical methods. *Journal of Business Research*, *70*, 263-286.

[9] Shah, N. D., Steyerberg, E. W., & Kent, D. M. (2018). Big data and predictive analytics: Recalibrating expectations. *JAMA-Journal of the American Medical Association*, *320*(1), 27-28.

[10] Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). *Foundations of machine learning*. MIT press.

[11] Mitchell, T. M. (1997). *Machine Learning.* McGraw-Hill Science

[12] Alpaydin, E. (2020). *Introduction to machine learning*. MIT press.

[13] Kameshwaran, K., & Malarvizhi, K. (2014). Survey on clustering techniques in data mining. *International Journal of Computer Science and Information Technologies*, *5*(2), 2272-2276.

[14] Gaurav, K. A., & Patel, L. (2020). Machine Learning With R. In Applications of Artificial Intelligence in Electrical Engineering (pp. 291-331). IGI Global.

[15] Cohen, Jacob (1960). "A coefficient of agreement for nominal scales". *Educational and Psychological Measurement*. **20** (1): 37–46. doi:10.1177/001316446002000104

[16] Yu, L., & Liu, H. (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)* (pp. 856-863).

[17] García López, F., García Torres, M., Melián Batista, B., Moreno Pérez, J.A., & Moreno-Vega, J.M. 2006. Solving feature subset selection problem by a parallel scatter search. European Journal of Operational Research, 169(2), 477--489.

[18] Yu, L., & Liu, H.: Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution. In: Proceedings of the Twentieth International Conference on Machine Learning, 856-863, 2003.