

**FACULTAD DE INFORMÁTICA**

**Grado en Ingeniería Informática**

Mención de Computación

Aprendizaje computacional para la detección de genes influyentes en patologías. Aplicación al cáncer de mama.

Proyecto Típico de Ingeniería

**Autor:** Francisco Cheda Pérez

**Director:** Amparo Alonso Betanzos

**Director:** Vicente Moret Bonillo

A Coruña, Febrero de 2017

## AGRADECIMIENTOS

*Gracias a mis directores, Amparo y Vicente, por aceptar este proyecto en cuanto se lo propuse, así como por todo el apoyo mostrado, estando siempre disponibles y mostrando paciencia conmigo.*

*Gracias también a Carla, por haberme inspirado a seguir cuando estaba agotado y por estar a mi lado cuando la necesitaba.*

*Finalmente, gracias a mi madre, que dedicó su vida a trabajar para que yo pudiera llegar a donde estoy, no hay palabras para describir todo lo que le debo*.

## RESUMEN

En los últimos años, el Big Data y la ciencia de datos se están usando para extraer valor de múltiples ámbitos. Entre ellos se encuentra la genética, donde la reducción en el coste de escanear genomas hace posible su utilización en campos como la prevención de enfermedades. En este proyecto se aborda el uso de la ciencia de datos y tecnologías como Apache Spark, para llevar a cabo análisis en el ámbito de la genética, con el fin de detectar los genes influyentes en patologías y aplicándolo al caso de uso del cáncer de mama.

Para lograr este objetivo, se realiza un estudio sobre la propia estructura del ADN, así como sobre sus problemáticas y las distintas técnicas existentes para su análisis. A continuación se realiza la preparación de los datos disponibles junto a la generación de datos de prueba, para con ellos realizar dos fases principales de análisis. Mediante el uso de algoritmos de selección de características se busca encontrar de entre los millones de atributos del genoma, cuales son los que afectan a una etiqueta analizada. Este proceso es realizado en la primera fase sobre bases genéticas, mientras que en la segunda se realiza sobre genes, presentando ambos casos distintas ventajas y desventajas que son analizadas en profundidad.

Las pruebas realizadas muestran el rendimiento de distintos algoritmos, así como su precisión, entre las cuales destaca la ofrecida por mRMR, un filtro que realiza selección de características basado en subconjuntos mediante la aplicación de la teoría de la información mutua. Con ello, se comprueba la capacidad de los métodos utilizados, así como la tecnología actual, para analizar el genoma y obtener las partes de este que influyen en distintas enfermedades.

Entre las conclusiones ofrecidas por el proyecto, se puede observar las posibilidades ofrecidas por este análisis, así como la importancia de extender el proceso de obtención de genomas con los que realizar pruebas sobre distintas patologías. Se ofrecen también nuevas posibilidades de futuro, como la utilización de los datos obtenidos para crear sistemas de detección de enfermedades y mejorar la prevención médica, así como recomendaciones en futuras extensiones a realizar sobre este proyecto.

## PALABRAS CLAVE

*Selección de características, Big data, Apache Spark, Genética*

## INDICE GENERAL

1. **Introducción………………………………………………………………………...1**
   1. Objetivos………………………………………………………………………....2
   2. Planificación y Evaluación de costes………………………………………….…3
   3. Estructura de la memoria………………………………………………………...6
2. **Campo de aplicación: genética y el cáncer de mama…………………………….8**
   1. Estructura del genoma…………………………………………………………...8
   2. Patología de estudio: cáncer de mama………………………………………….10
3. **Fundamentos Teóricos……………………………………………………………11**
   1. Big Data y ciencia de datos……………………………………….……………11
   2. Aprendizaje computacional y selección de características……………….…….16
   3. Rendimiento, complejidad computacional y computación distribuida…………22
4. **Herramientas y materiales…………………………………………………..……25**
   1. Análisis de alternativas…………………………………………………………25
   2. Metodología………………………………………………………………….…27
   3. Fundamentos tecnológicos…………………………………………………...…30
      1. Apache Hadoop…………………………………………………………30
      2. MapReduce……………………………………………….……...……..34
      3. Apache Spark.……………………………………………………...…...35
      4. Algoritmos………………………………………………………...…....42
   4. Entorno de trabajo………………………………………………………………48
   5. Datos utilizados………………………………………………………...………49
5. **Desarrollo………………………………………………………………………….50**
   1. Fases del proyecto………………………………………………………………50
   2. Los dos modelos de análisis: bases genéticas vs genes………………………...53
   3. Generación y manipulación de datos…………………………………………..55
   4. Algoritmos usados……………………………………………………………..57
6. **Resultados…………………………………………………………………………63**
   1. Medidas de evaluación…………………………………………………………63
   2. Medidas de tiempo……………………………………………………………..64
   3. Medidas de precisión…………………………………………………………...66
7. **Conclusiones y trabajo futuro …………………………………………………...68**
8. **Apéndices……………………………………………………………………….….71**
   1. Código……………………………………………………………………….….71
9. **Bibliografía……………………………………………………………………..….83**

## INDICE DE FIGURAS

* Figura 1: Diagrama de Gantt………………………………………...………5
* Figura 2: Fuentes de Big Data…………………………………….…..……13
* Figura 3: Crecimiento de datos según su estructura…………….……..…..14
* Figura 4: Esquema del ciclo de vida de análisis de datos……………….…27
* Figura 5: Arquitectura de HDFS…………………………………………...31
* Figura 6: Replicación en HDFS…………………………………………....32
* Figura 7: Componentes de Spark…………………………………………..36
* Figura 8: Ejemplo de código en Spark……………………………………..37
* Figura 9: Comparación de rendimiento y lenguaje en Spark……………...39
* Figura 10: Número de ejemplos vs Tiempo………………………………..65
* Figura 11: Número de atributos vs Tiempo………………………………..65

## INDICE DE TABLAS

* Tabla 1: Tiempos y Costes…………………………………………………..3
* Tabla 2: Función EXOR……………………………………………………19
* Tabla 3: Crecimiento en coste según complejidad y entrada………………22
* Tabla 4: Comparación de tiempos entre algoritmos………………………..64
* Tabla 5: Comparación de tiempos entre lenguajes…………………………64
* Tabla 6: Medidas de eficacia 1……………………………………………..66
* Tabla 7: Medidas de eficacia 2……………………………………………..66

## 1. INTRODUCCIÓN

En los últimos años se ha producido un boom en la utilización masiva de datos. Desde el comportamiento de las personas en redes sociales, hasta las mediciones tomadas por la multitud de dispositivos que forman el internet de las cosas, ofrecen información que puede ser analizada con el fin de extraer nuevo conocimiento. Estos nuevos tipos de datos suelen ser referidos por el término Big Data, y el proceso de extraer información de valor de ellos como ciencia de datos.

De entre las distintas fuentes que generan Big Data una de las más recientes está formada por los datos genéticos. Gracias al mapeo del primer genoma humano y al constante descenso en el precio de escanear la nuevos genomas, existe la posibilidad de utilizar esta valiosa información para comprender mejor el funcionamiento de nuestros cuerpos, así como el comportamiento que estos presentan frente a distintas patologías o substancias. Entre las patologías más conocidas que presentan una amplia influencia genética se encuentra el cáncer de mama, cuya investigación ha descubierto varios genes que afectan de forma considerable a su aparición. En este proyecto usaremos el caso de uso de esta enfermedad, así como datos genéticos, para comprobar la viabilidad y desarrollar mecanismos que permitan detectar automáticamente los genes que afectan a distintos aspectos, como es el desarrollo de enfermedades.

Con el fin de lograr este objetivo, se realizara un estudio de la estructura del genoma, para a continuación aplicar sobre los datos disponibles técnicas típicas de la ciencia de datos, entre las que se encuentran el aprendizaje computacional, y más específicamente, la selección de características. Estos procesos de selección de características buscaran encontrar entre la multitud de atributos que presenta el genoma, aquellos que son importantes para comprender la presencia de una etiqueta dada, y mediante este proceso, identificar los atributos que influyen es esa etiqueta.

La aplicación de estas técnicas se realizara mediante el uso de plataformas como Apache Spark, que presenta una eficacia demostrada para el trato con cantidades masivas de datos así como la aplicación de algoritmos interactivos sobre estos, como es el caso de los que forman parte del aprendizaje computacional. Se seleccionaran varios algoritmos de selección de características, y se comprobaran tanto su rendimiento como capacidad predictiva en el ámbito de aplicación.

Para realizar este proceso, se llevara a cabo una amplia fase formativa que ofrezca el conocimiento necesario para trabajar con las técnicas y herramientas a tratar. A continuación se procederá al desarrollo del proyecto mediante la aplicación del ciclo de vida del análisis de datos, el cual está especialmente pensado para el tratamiento y comportamiento de proyectos de ciencia de datos. Finalmente, se realizara un análisis sobre los resultados y las posibilidades de trabajo futuro que haya ofrecido el presente proyecto.

### 1.1. Objetivos

A continuación se presenta una sumarización de los objetivos que se pretende alcanzar durante el desarrollo del presente proyecto de fin de Grado:

* Obtención y generación de los conjuntos de datos necesarios.
* Aprendizaje de las estructuras utilizadas en el trabajo con datos genéticos así como la preparación de estos datos para poder analizarlos de forma eficiente.
* Formación en la plataforma Apache Spark así como en otras herramientas y metodologías relacionadas con el análisis masivo de datos.
* Realización de un estudio sobre diversos algoritmos de selección de características y su aplicación mediante el uso de tecnologías propias del análisis de datos masivos en el problema que se nos presenta.

### 1.2. Planificación y evaluación de costes

Es este apartado mostraremos la evaluación de costes del proyecto, así como la planificación de este mediante un diagrama de Gantt, que incluye tanto la línea de base inicial como el estado de esta en el momento de elaboración de este documento.

Para la realización de la evaluación de costes se ha utilizado el salario correspondiente al valor medio en el sueldo de un científico de datos, situándose este en España en torno a los 25.000 € anuales y tomando, de forma simplificada, un coste de 13 € por hora.

Es además necesaria una explicación de algunas de las decisiones tomadas al medir los costes presentados. Una de ellas se refiere a la formación realizada, pues dado que mientras ciertas empresas se desentienden completamente de los costes de esta, otras se ofrecen a correr con los costes de los cursos formativos, mientras que existen empresas que no solo corren con estos sino que también retribuyen al empleado por el tiempo invertido en este proceso, siendo esta la aproximación tomada. La segunda decisión tomada es la relacionada con el coste de los equipos usados, siendo un tema que en el ámbito laboral puede resultar más complejo, al tratar con la multitud de aspectos relacionados con el mantenimiento y/o compra de los equipos. Actualmente una opción cada vez más presente, es la de utilizar servicios remotos que ofrecen capacidades de computo en diferentes rangos de capacidad y precio, siendo esta la opción tomada para nuestros cálculos.

A continuación, se muestra la tabla de tiempos y costos incurridos durante el desarrollo que nos atañe.

|  |  |  |  |
| --- | --- | --- | --- |
| Tarea | Tiempo | Coste / H Hombre | Coste |
| Formación:  Remuneración | 200 h | 13 € | 2,600 € |
| Formación:  coste cursos |  |  | 400 € |
| Preparación del entorno de trabajo | 10 h | 13 € | 130 € |
| Desarrollo | 600 h | 13 € | 7,800 € |
| Realización de la Documentación | 220 h | 13 € | 2,860 € |
| Utilización de hardware | 1210h | 0.5 € / h | 605 € |
|  |  |  |  |
| Coste total |  |  | 14,395 € |

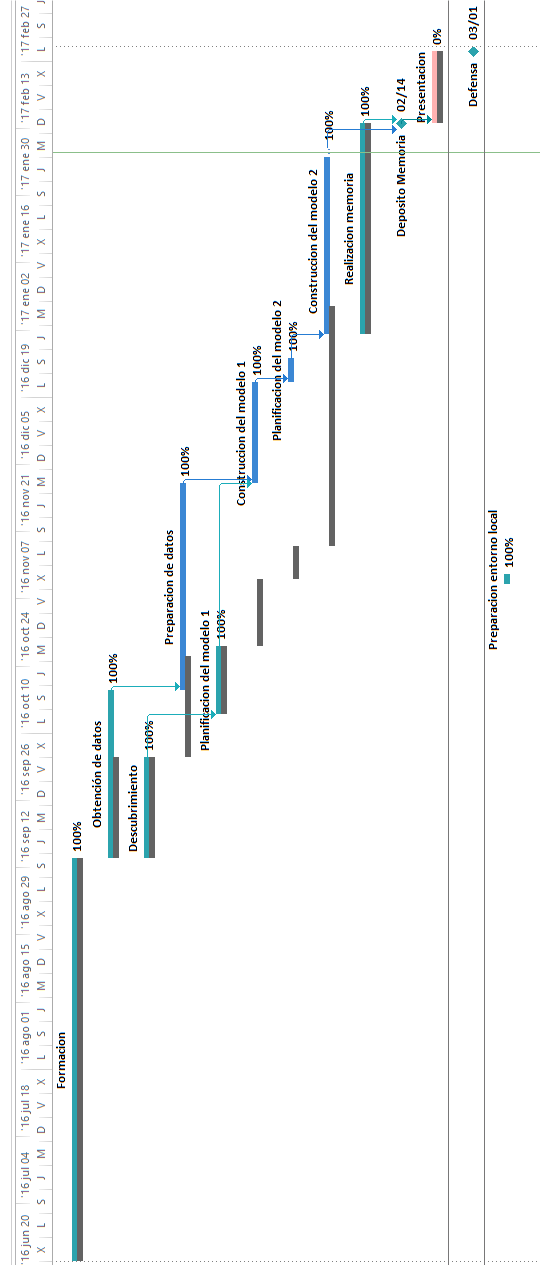
Tabla 1: Tiempos y Costes

Tras haber observado la tabla de costes pasaremos a mostrar en la página siguiente el diagrama de Gantt con la planificación, aunque el motivo de mostrarlo en este orden no es otro que el formato de este, dada la dificultad de ajustar un diagrama de este tipo a una sola página.

En este diagrama las líneas de color negro muestran la planificación inicial, mientras que las de tono azulado el desarrollo real, siendo las más claras tareas cuyo inicio se ha visto desplazado debido a retrasos en tareas anteriores. Se puede observar también como multitud de las tareas presentes han requerido periodos de tiempo muchísimo mayores a los planificados, esto fue debido a multitud de problemas provocados por el volumen de datos tratados así como la inexperiencia inicial en el ámbito, pudiendo observarse sobre todo en el aspecto de la preparación de datos, siendo esto explicado en los capítulos de desarrollo. Otros aspectos interesantes del diagrama incluyen la tarea rosada al final, que aún no ha comenzado en el momento de escribir este documento, y la tarea de preparación del entorno local, no planificada inicialmente.

Procediendo en la siguiente página, encontraran el diagrama mencionado.

Figura 1: Diagrama de Gantt.

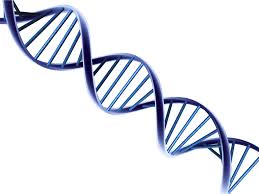


### 1.3. Estructura de la Memoria

En esta sección se enumeran los distintos capítulos de los que consta esta memoria, realizando una pequeña descripción de esta.

* **Capítulo 1, Introducción:** Inicia el contenido del proyecto con la motivación y los objetivos de este, añadiendo su planificación y una referencia a la metodología usada. Seguido por el actual resumen de las secciones de este documento.
* **Capítulo 2, Campo de aplicación:** En este apartado se trata el tema del genoma como campo de aplicación del proyecto.
* **Capítulo 3, Fundamentos teóricos:** Es este apartado, donde se abordan los conceptos teóricos necesarios para la comprensión de este proyecto.
* **Capítulo 4, Herramientas y materiales:** En él se aborda una introducción a las distintas alternativas tecnológicas disponibles para abordar las tareas a realizar, se expone el entorno de trabajo usado, y se explican las principales herramientas y algoritmos utilizados durante su desarrollo, así como la metodología seguida en este. Finalmente se incluyen los datos iniciales disponibles para ello.
* **Capítulo 5, Desarrollo:** Este apartado cuenta con una narrativa sobre el desarrollo del proyecto según la metodología usada, así como las problemáticas encontradas. Mostrándose también el proceso seguido para la generación y manipulación de los datos usados.
* **Capítulo 6, Resultados:** El capítulo de resultados expone las distintas pruebas y resultados obtenidos, así como tablas informativas y medidas de rendimiento sobre estos.
* **Capítulo 7, Conclusiones y trabajo futuro:** Tras haber completado el desarrollo, en este capítulo se exponen las conclusiones alcanzadas sobre este, así como posibles mejoras y trabajo futuro sobre las que se podría ampliar lo aquí explorado.
* **Capítulo 8, Apéndices, Código:** En este apéndice se incluye los distintos algoritmos y secuencias de ETL desarrolladas durante el proyecto.
* **Capítulo 9, Bibliografía:** Finalmente como documentación bibliográfica se incluyen las principales fuentes de información utilizadas, así como referencias a las fuentes de datos y librerías.

## 2 CAMPO DE APLICACIÓN: GENÉTICA Y CÁNCER DE MAMA.



El campo de aplicación de este proyecto se haya en el análisis de datos genéticos, y por ello, en esta sección mostramos una descripción de la estructura del genoma, así como de los aspectos más importantes, o problemáticos, al aproximarse a su análisis mediante el uso de la ciencia de datos.

### 2.1. Estructura del genoma

El termino genoma se refiere al conjunto de información genética de una célula u organismo, en el caso del ser humano este está compuesto por 23 pares de genes en el núcleo de la célula, más el genoma de las mitocondrias, unos órganos celulares encargados de la generación de energía. Estos genes están formados a su vez por una secuencia de bases fosfatadas similares a los 0s y 1s que forman un código máquina, aunque estos se presentan base cuatro. Estas bases son leídas en ternas por unos orgánulos celulares llamados ribosomas, que toman estas secuencias y las mapean a aminoácidos, produciendo cadenas de estos que reciben el nombre de proteínas.

Las proteínas fabricadas son las responsables del funcionamiento del organismo. Sus tareas van desde decidir la construcción de tejidos, hasta regular los niveles de salinidad en el entorno celular, o de controlar que él mecanismo de copia de la propia célula se efectúe correctamente. Los errores en estas proteínas son, por lo tanto, la causa de gran cantidad de enfermedades, pero también de provocar diferencias en fenotipos tales como el color de los ojos.

El ADN presenta además multitud de problemáticas en su estructura que afectaran a la cuestión de su análisis. La primera de ellas puede intuirse por lo comentado anteriormente y es que dado que el efecto final del genoma es transmitido por las proteínas, cualquier modificación en la codificación de esa proteína afectaría a su funcionamiento y por ello a las funciones que esta desempeña, pero al mismo tiempo el efecto concreto variara dependiendo del cambio producido, de en qué parte de la proteína se haya dado y de la proteína concreta.

Otra dificultad se encuentra con que en el caso de que los humanos contamos, en general, con dos copias de cada cromosoma. Cada una de las bases genéticas situadas en la misma posición son llamadas alelos y el efecto de una modificación varía según esta se produzca en uno, ambos o ninguno de estos alelos. La complejidad de este aspecto aumenta más si tenemos en cuenta que dependiendo del gen concreto, una sola variación puede manifestar prioridad, considerándose dominante, o por el contrario no la presenta, siendo recesivo, o dicho de otro modo se necesitan dos copias iguales para presentar el efecto codificado. Para más, el comportamiento dominante o recesivo puede depender de otro gen o de otro cromosoma, como es el caso de la variación que regula la calvicie, siendo dominante en hombres y recesivo en mujeres, pero no encontrándose en los cromosomas que regulan si se es hombre o mujer.

A pesar de todo lo contado anteriormente, las complicaciones continúan, y es que las bases que forman el genoma no son inyectivas. Esto quiere decir que una base genética puede codificar varios genes y por lo tanto proteínas al mismo tiempo y se debe a que en parte, el genoma se comporta de modo similar a como lo hace el código informático con el aspecto, de que se puede empezar a leer desde cualquier punto. En el fondo el código del genoma no son simplemente las instrucciones para una serie de proteínas en secuencia, sino que en él se codifican instrucciones *begin* y *end*, indicando los puntos de inicio de lectura de un gen, de ahí los múltiples puntos de inicio, junto a comentarios indicando grandes zonas a ignorar, código con especial protección en el sentido de “no tocar” y referencias a otras zonas del genoma del estilo “aquí va una cabeza, ir a: código para la cabeza”, etc. [1]

Volviendo al tema del genoma como estructura, el hecho de poder mapear por primera vez un genoma humano supuso un altísimo coste, reportado como cercano a los 2.700 millones de dólares [17], pero gracias a ello y a las diversas técnicas informáticas desarrolladas, este análisis puede realizarse ahora a mucho menor coste, que ya se encuentra en torno a los 1000 €. Como nota adicional, este proceso se realiza mediante la lectura de millones de pequeños pedazos de ADN, y mapeando estos sobre el genoma original. Esto se consigue hacer además a pesar de las pequeñas diferencias entre los genomas y siendo esas diferencias con respecto al genoma base conocidas como llamadas variantes.

Una llamada variante consiste pues, en la conclusión de que hay una base o nucleótido que varía en una posición concreta del genoma con respecto a otro genoma base. Dado que dos humanos del mismo sexo comparten entre si aproximadamente el 99.9% del genoma, el estudio de estas variantes permite investigar el distinto comportamiento de cada individuo con respecto a campos tan amplios como la aparición de enfermedades o la reacción ante ciertas sustancias.

### 2.2. Patología de estudio: Cáncer de mama

Para el desarrollo a realizar se utilizara el caso de estudio del cáncer de mama, este es el tumor más frecuente entre la población femenina, con más de 25.000 nuevos casos al año solamente en España y 1 de cada 8 mujeres desarrollándolo a lo largo de su vida. Uno de los aspectos que han llevado a la elección de esta enfermedad para orientar el desarrollo, no es otro que la investigación realizada con respecto a esta, así como a la clara influencia genética que presenta.

Para ofrecer datos concretos, se tiene que la tasa de aparición de esta patología entre la población femenina general es entorno al 12%, mientras que las mujeres con una mutación dañina en el gen BRCA1 ronda entre el 55% y 65%. Otro gen influyente es el BRCA2 que causa una tasa de incidencia del 45%, por descontar que en caso de presentarse errores en ambos genes las estimaciones rondan el 90%. En el caso de los hombres la aparición del cáncer de mama es prácticamente nula, pero en el caso de presentar ambas mutaciones su aparición se dispara hasta tasas del 10 % [16]

Cabe anotar también que estos genes presentan influencia en multitud de otras enfermedades cancerígenas, dado que son genes supresores de tumores, mediante la regulación de la división celular, que evitan la proliferación incontrolada. [19]

## 3. FUNDAMENTOS TEORICOS

Para el desarrollo y la comprensión de lo realizado en este proyecto se requieren una serie de fundamentos teóricos que se incluyen en esta sección. Estos incluyen los conceptos de Big Data y ciencia de datos, una visión del aprendizaje automático con una mayor profundización en el aspecto de la selección de características y unas nociones básicas de complejidad computacional y computación distribuida que pretenden exponer aspectos que estos campos presentan con respecto al Big Data.

### 2.2. Big Data y ciencia de datos

Nuevos datos se crean constantemente y a una velocidad cada vez mayor, siendo ejemplos los teléfonos móviles, las redes sociales, las tecnologías de captura de imágenes, etc. Todos estos y más generan datos que deben ser almacenados en algún lugar y con algún propósito. Mientras tanto, distintos dispositivos y sensores generan constantemente información que debe ser procesada en tiempo real y al mismo tiempo se suma al volumen total de nuevos datos. Simplemente mantenerse al día con esta gran cantidad de datos es difícil, pero substancialmente más difícil es analizar todos estos datos para extraer patrones e información significativa, especialmente cuando no siguen las nociones tradicionales de estructura.

Existen multitud de ejemplos del uso de estos datos, ofreciendo una visión de la importancia que este puede tener. Ejemplos de esto se encuentran desde las compañías de tarjetas de crédito que monitorizan cada movimiento para identificar transacciones fraudulentas, a las compañías telefónicas, que analizan los patrones en las llamadas de sus clientes con el fin de detectar, si estas indican un alto número de conocidos en una compañía rival, lo cual podría llevar a la potencial perdida del cliente. Mientras tanto otras compañías, como las redes sociales, utilizan los datos en sí mismos como su principal producto.

A la hora de explicar mejor en que consiste este gran influjo de datos conocido como Big Data, podemos destacar varios aspectos que se consideran características definitorias de esta:

* **El inmenso volumen de datos:** Donde en lugar de miles o millones de filas, el Big Data puede contar con billones de filas y millones de columnas.
* **La complejidad de los tipos y estructuras de datos:** Dado que el Big Data cuenta con gran variedad de nuevas fuentes de datos, desde imágenes, video y muestras de audio, hasta texto o trazas dejadas por la utilización de diversos sistemas.
* **La velocidad de generación de nueva información:** Donde el Big Data se puede describir como datos de gran velocidad, tanto generándose rápidamente, como en muchas ocasiones, siendo necesario analizar estos datos en tiempo real.

Aunque es el volumen de los datos lo que tiende a atraer más la atención, generalmente es la variedad y velocidad de los nuevos datos lo que ofrece una definición más adecuada, siendo el Big Data normalmente descrita como teniendo 3 Vs, volumen, variedad y velocidad. Más recientemente se están añadiendo a esta definición otras 2 Vs, Variabilidad, referenciando la posible inconsistencia en los datos y Veracidad, referenciando el nivel de calidad de estos. Debido a estas características, el Big Data no puede ser analizado eficientemente usando solo métodos y bases de datos tradicionales. Por ello, se requieren nuevas herramientas y tecnologías que permitan almacenar, manipular y extraer beneficio de estos grandes conjuntos de datos. Otra definición de Big Data viene dada por McKinsey en 2011 que se puede traducir como “El Big Data son datos cuya escala, distribución, diversidad y o línea temporal, requieren del uso de nuevas arquitecturas técnicas y análisis para obtener conocimiento y desbloquear nuevos recursos de valor de negocio” [4]



Figura 2 :Fuentes de Big Data

Como podemos ver en el grafico superior, existen multitud de fuentes de Big Data, mencionando algunas de ellas, las redes sociales y la secuenciación genética se encuentran entre las que presentan un crecimiento más rápido y son ejemplos de fuentes de información no tradicionales que están siendo analizadas en estos momentos. Por poner un caso más concreto de este análisis, nos encontramos con que el conjunto de los usuarios de Facebook, publicó en 2012, una media de 700 actualizaciones de estado por segundo. Estas actualizaciones pueden ser analizadas para deducir intereses o visiones políticas y mostrar a esos usuarios anuncios relevantes. Otro ejemplo viene del campo de la genética, donde la secuenciación y el mapeo del genoma de las personas, ofrece una comprensión detallada de la herencia de esta. La industria de la salud, sigue buscando en la actualidad avances en esta rama, con el objetivo de ayudar en la prevención de las enfermedades que una persona desarrollará con alta probabilidad a lo largo de su vida, para con ello eliminar o reducir el impacto de estas mediante el uso de tratamientos personalizados.

En cuanto a la estructura de los datos, el Big Data viene dado de múltiples formas, incluyendo datos estructurados y no estructurados, entre los cuales podemos recordar ejemplos como los ya mencionados: datos financieros, archivos de texto, archivos multimedia, mapas genéticos, etc. A diferencia de la información usada tradicionalmente en el análisis de datos, la mayoría estos nuevos datos son desestructurados o semiestructurados. Aunque analizar datos estructurados tiende a ser la técnica más familiar, se requieren técnicas diferentes para el análisis de los otros tipos de datos. Los siguientes son ejemplos de distintos tipos de datos según su estructura:

* **Datos Estructurados:** Estos contienen un tipo de datos definido, formato, incluyendo entre otros ejemplos datos de transacciones, archivos CSV, etc.
* **Datos Semi-estructurados:** Datos textuales compuestos por patrones reconocibles que permiten su parseo, como puede ser XML
* **Datos Quasi-estructurados:** Compuestos también por texto así como por formatos erráticos, que requieren un mayor esfuerzo y tiempo para su parseo, como por ejemplo una secuencia de clics, o de acciones realizadas por un usuario.
* **Datos desestructurados:** Sin ninguna estructura inherente pueden incluir documentos y mensajes de texto, imágenes, video, etc.

Además de su definición, podemos analizar el volumen de estos. En siguiente figura se muestran estos cuatro tipos de datos. En ella se puede ver que en torno al 80% de los nuevos datos, vienen en forma no estructurada. Es común además que estos tipos de datos estén mezclados entre sí, por ejemplo un sistema puede contener una serie de campos estructurados, como ip o usuario, acompañado de otros semiestructurados, como secuencias de clicks y no estructurados como grabaciones telefónicas, video llamadas, etc.

Figura 3: crecimiento de datos según su estructura

Otro aspecto importante es cómo realizar el análisis de estos datos, para lo que se requieren técnicas distintas a las utilizadas tradicionalmente. Normalmente, aquella rama que se encarga de analizar el Big Data, se conoce como ciencia de datos, y a continuación vamos explicar las diferencias entre esta y otros métodos como la inteligencia de negocio o BI, con el objetivo de facilitar la interpretación de este concepto.

Una forma de evaluar el tipo de análisis realizado es examinar el horizonte temporal y el tipo de aproximaciones analíticas usadas. BI hace fácil contestar preguntas y búsquedas sobre el momento actual o el pasado, como beneficios trimestre a trimestre, o como un producto vendió con respecto a otro. Estas preguntas pueden responderse agregando y agrupando datos históricos en algún modo, donde el BI ofrecerespuestas sobre cuándo o dónde los eventos ocurrieron. Por comparación, la ciencia de datos tiene a usar datos con una mirada más hacia el futuro. Esto se hace centrándose en analizar el presente para permitir la toma de decisiones informadas sobre el futuro. En lugar de agregar datos históricos con el fin observar cuando vendió un producto, un equipo de ciencia de datos puede utilizar técnicas como análisis de series temporales para predecir la respuesta futura. Además, la ciencia de datos suele presentar una naturaleza más exploratoria, enfrentándose a preguntas más abiertas. Esta visión que ofrece *insights* en la actividad actual y predicciones del futuro, generalmente centrándose más en preguntas como “¿Cómo?” y “¿Por qué?”. Además mientras BI suelen usar datos altamente estructurados, los proyectos de ciencia de datos utilizan muchas fuentes de datos, incluyendo grandes conjuntos de datos no convencionales. Dependiendo pues de las necesidades, un negocio se puede embarcar en BI para crear reportes o visualizaciones y en Ciencia de datos si necesita análisis más sofisticados y con datos más complejos.

Podríamos pues, explicar la ciencia de datos, como una extensión de la estadística, capaz de trabajar con Big Data, mediante el uso de las ciencias de la computación, en donde se incluye el aprendizaje automático y la selección de características, con el fin de extraer valor de estos datos comprendiendo el porqué de su comportamiento y extrayendo predicciones futuras de estos.

### 2.1. Aprendizaje computacional y selección de características

El aprendizaje automático es una sub rama de las ciencias de la computación en la que se crean algoritmos capaces de aprender y hacer predicciones sobre datos, mediante la creación de modelos computacionales. Entre sus distintas aplicaciones se encuentran la visión artificial, los motores de búsqueda, la traducción, el reconocimiento de voz, etc.

Las tareas abarcadas por el aprendizaje automático pueden clasificarse en tres amplias categorías:

* **Aprendizaje supervisado**: Creando modelos mediante la presentación de ejemplos de entrada junto las salidas o etiquetas deseadas, buscando que el sistema aprenda a mapear las entradas a las salidas.
* **Aprendizaje no supervisado:** Realizando procesos en donde no se presentan etiquetas y se busca encontrar estructuras en los datos.
* **Aprendizaje por refuerzo**: Que se basa en un sistema que interactúa con un ambiente dinámico recibiendo refuerzo positivo o negativo mientras navega el problema.

Otra categorización de los algoritmos de aprendizaje computacional se presenta cuando se considera la salida del sistema:

* **Clasificación:** Presentando las entradas clasificadas en clases y donde el sistema debe aprender a clasificar en estas, los nuevos datos que se le puedan ofrecer.
* **Regresión:** Se comporta de modo similar al anterior, pero la salida es continua en lugar de discreta.
* **Agrupación:** Es otra clasificación de entradas, pero a diferencia de en clasificación no se conocen los grupos de antemano, por lo que el sistema agrupa las entradas en clusters que presentan alguna similitud.
* **Reducción de dimensionalidad:** En donde se buscar simplificar las entradas mapeándolas a un espacio de menor dimensionalidad.

Como acabamos de ver, uno de los problemas del aprendizaje automático consiste en la selección de aquellas variables significativas para un problema dado de entre todos los datos disponibles. Esta es comúnmente una fase inicial del proceso antes de proceder al uso de otros métodos como clasificación, pero en casos como el que se presenta en este documento, supone un fin en sí mismo.

Podemos además distinguir dos tipos de reducción de dimensionalidad:

* **Extracción de características:** En donde de forma linear o no, se combinan los atributos disponibles para obtener otros nuevos.
* **Selección de características:** La cual busca seleccionar un subconjunto de las variables originales.

La principal premisa con respecto al uso de la selección de características es que los datos contienen muchos atributos que son redundantes o irrelevantes y que, por lo tanto, pueden eliminarse sin perder información. Esto permite simplificar los modelos, haciéndolos más fáciles de interpretar, reducir tiempos de entrenamiento para los algoritmos da aprendizaje máquina y mejorar la generalización de los modelos entrenados entre otras ventajas.

Un algoritmo de selección de características puede verse como una técnica de búsqueda sobre el espacio de características inicial, con el objetivo de encontrar el subconjunto de estas que ofrezca el mejor resultado. El algoritmo más simple sería realizar una búsqueda exhaustiva sobre los subconjuntos posibles, pero ese método es computacionalmente inviable. Por ello es necesaria la utilización de métricas de evaluación que permitan escoger cada uno de los componentes del conjunto final, siendo los principales tipos de métodos para ello *wrappers*, filtros y métodos embebidos.

Muchos de estos algoritmos utilizan rankings de variables como mecanismo principal o auxiliar por su simplicidad, escalabilidad y buenos resultados empíricos. Estos métodos utilizan un criterio para valorar cada variable con respecto a la salida de forma independiente de las demás variables. Uno de estos criterios sería el de correlación, dado por ejemplo, por el criterio de correlación de Pearson [23]:

pearsons1

Donde *cov* implica la covarianza y *var* la varianza.

Por lo tanto, la estimación de R(i) es:

pearsons2

En regresión linear, el coeficiente de determinación, R(i)2 representa la fracción de la varianza rodeando la media de y, que depende de la relación linear entre x e y, por lo que el uso de este criterio busca un ranking según lo bien que encajen linealmente ambas variables. Aunque realizar este cálculo es muy eficiente computacionalmente, solo puede detectar las dependencias lineales entre ambas variables. Para eliminar esta limitación se puede usar una función no linear, o usar pre procesados no lineales y a continuación calcular un coeficiente de correlación simple.

Como se ha visto, usando R(i)2 como criterio se crea un ranking que depende de lo bien que las variables encajen linealmente. Esta idea de seleccionar variables según su capacidad predictiva individual se puede extender a problemas de clasificación, construyendo un clasificador a partir de una única variable y usando la tasa de error para clasificar la capacidad predictiva de esa variable. Es también posible usar otros criterios que incluyan el uso de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos, pudiendo además regularse la importancia de cada uno de los tipos de errores.

Otra forma de realizar la selección de variables consiste en la utilización de la teoría de la información. Donde muchas aproximaciones usan estimaciones empíricas sobre la información mutua de cada variable con el objetivo.



Donde P(xi) y P(y) son las probabilidades de xi e y, y P(xi,y) es la probabilidad de la unión. Siendo el criterio I(i), la medida de la dependencia entre la probabilidad de la variable xi y la probabilidad de y.

La dificultad de este cálculo radica en que las probabilidades son desconocidas y su estimación es difícil. Por otra parte, en el caso de variables discretas o nominales, esto resulta más sencillo, pues la integral se vuelve una suma, permitiendo entonces estimar las probabilidades según sus frecuencias y calcular I(i) del siguiente modo.

infoth2

Hasta ahora los criterios tratados buscaban la selección de variables individualmente según su relación con la salida, pero existen variables que por sí solas no muestran relación esta relación, cuando en realidad si la tienen. Un ejemplo de esto es la función EXOR. Al no tener correlación con la salida, ambas variables son inútiles por sí mismas y nunca serían seleccionadas por uno de los métodos anteriores, pero conjuntamente si son útiles y representan la salida, en este caso, perfectamente.

|  |  |  |
| --- | --- | --- |
| Variable 1 | Variable 2 | Salida |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Tabla 2: función EXOR

Queda clara entonces la importancia de seleccionar subconjuntos de variables que presenten una buena capacidad predictiva en conjunto. En cuanto a los métodos principales utilizados para obtener estos subconjuntos tenemos, como nombramos anteriormente, wrappers, filtros y métodos embebidos. Los Wrappers utilizan aprendizaje computacional como si fuese una caja negra para obtener subconjuntos según su capacidad predictiva. Los filtros, por otro lado, seleccionan subconjuntos independientemente que son independientes del predictor elegido. Finalmente, los métodos embebidos realizan selección de variables en el propio proceso de entrenamiento, siendo normalmente específicos del algoritmo de aprendizaje computacional en el que se incluyen.

La metodología de *wrappers* ofrece una forma simple y potente de enfrentarse al problema de selección de variables, de hecho, el método de aprendizaje automático utilizado puede interpretarse como una caja negra. En su formulación más general, la metodología consiste en usar la predicción de un algoritmo cual sea, de aprendizaje máquina, para observar como de eficaz es el subconjunto. En la práctica, se necesita definir como buscar en el espacio de posibles subconjuntos, como manejar el poder predictivo del algoritmo para guiar la búsqueda y detenerla, así como que predictor usar.

A la hora de explorar el espacio de subconjuntos, una búsqueda exhaustiva puede usarse si el número de variables no es muy grande, pero el problema es NP-complejo, y la búsqueda se vuelve, con rapidez, impracticable computacionalmente hablando. Para solventar esto existen múltiples estrategias incluyendo *best-first* y algoritmos genéticos entre otros. En cuanto al rendimiento, este se suele abordar usando un conjunto de validación o *cross-validation* y algunos de los predictores más populares incluyen arboles de decisión, *naive Bayes*, *linear least-square*, y máquinas de soporte vectorial.

Los wrappers son criticados con frecuencia por parecer una solución de fuerza bruta, requiriendo cantidades masivas de computación, pero existen métodos de búsqueda eficientes que no sacrifican poder predictivo. Una de estas opciones son las estrategias voraces, siendo además robustas frente al sobreentrenamiento. Entre estas opciones, tenemos dos ejemplos en *forward selection* y *backward selection*. En el primero las variables son agregadas progresivamente en subconjuntos cada vez más grandes, mientras que en el segundo se empieza con todas las variables y se van eliminando las menos prometedoras, ofreciendo ambos métodos, subconjuntos anidados de variables.

Usando aprendizaje computacional como una caja negra, los wrappers son universales y simples, pero los métodos embebidos, que incorporan selección de variables como parte del proceso de entrenamiento, pueden resultar más eficientes en varios aspectos. Esto se debe a que pueden hacer un mejor uso de los datos disponibles, no teniendo que dividir los datos en conjuntos de entrenamiento y validación, además de encontrar la solución más rápido al evitar reentrenar el predictor desde cero para cada subconjunto probado.

El último tipo de método a tratar son los filtros, que presentan múltiples justificaciones para su utilización en la selección de subconjuntos. Uno de estos es que comparado con los wrappers, los filtros son más rápidos, aunque actualmente existen métodos embebidos que son competitivos en ese aspecto. Otro argumento es que algunos filtros, como los basados en el criterio de información mutua, ofrecen una selección genérica de variables que no está ajustada a ningún modelo de aprendizaje computacional concreto. Mientras que otra justificación es que estos se pueden utilizar como paso de pre procesado para reducir la dimensionalidad inicial.

Los filtrados basados en teoría de la información constituyen una amplia familia, presentando la justificación de que la medida de información mutua no depende de ningún proceso de predicción, pero ofrece un límite en la tasa de error de cualquier predicción para la distribución dada. Uno de estos algoritmos de filtrado es mRMR, el cual está siendo usado con frecuencia como método para identificar características de genes y fenotipos [3].

### 2.3. Rendimiento, complejidad computacional, y computación distribuida.

Antes de finalizar este capítulo, es adecuado repasar ciertos conceptos que presentan una gran influencia, así como algunas peculiaridades, en los campos del Big Data y del aprendizaje automático. Comencemos pues, con un recordatorio sobre la complejidad computacional.

La teoría de la complejidad computacional se centra en clasificar los problemas de computación según su dificultad inherente, relacionando esas clases entre ellas. Un problema de computación es entendido por una tarea que, en principio, puede resolverse por un ordenador, según la aplicación de una serie de pasos matemáticos como un algoritmo.

Para clasificar estos problemas se utiliza la notación O, representando problemas cuyo coste crece con respecto a la entrada según distintas medidas, como tiempo o memoria utilizada, de modo equivalente a distintas series numéricas. Entre estas podemos encontrar O(1) significando que ese coste es constante y no afectado por la entrada, O(n) como coste lineal, O(n2) y otros como crecimientos logarítmicos. Todos los ejemplos nombrados hasta ahora pueden además clasificarse como problemas resolubles en complejidad polinómica, pues siempre existe un polinomio que crece más rápidamente que estos. Otro caso que presenta un crecimiento mayor son las complejidades exponenciales O(2n) por poner un ejemplo, cuyo crecimiento sería muy superior y entraría en el grupo de problemas conocidos como NP, o no polinómicos. En la siguiente tabla se pueden como el crecimiento de la entrada afecta al tiempo de salida dependiendo de la clase de complejidad del algoritmo usado.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clase | n=2 | n=16 | n=256 | n=1024 |
| 1 | 1 | 1 | 1 | 1 |
| log(n) | 1 | 4 | 8 | 10 |
| n | 1 | 16 | 256 | 1024 |
| n\*log(n) | 2 | 64 | 2048 | 10240 |
| n2 | 4 | 256 | 65526 | 1048567 |
| n3 | 8 | 4096 | 16777216 | 1.07E+09 |
| 2n | 4 | 65536 | 1.16E+77 | 1.8E+308 |

Tabla 3: crecimiento en coste según complejidad y entrada

Debido a esto, queda clara la importancia de utilizar algoritmos con la menor clase de complejidad posible, pero esta es incluso mayor si tenemos en cuenta el caso del Big Data, donde n alcanza valores de millones, billones y más. Siendo común la frase sobre Big Data que dice “lo máximo que puedes hacer es leer los datos una sola vez”, representando el problema de que incluso consiguiendo algoritmos eficientes, al ser n tan grande su coste de ejecución será también intrínsecamente alto.

Para resolver problemas de tamaños tan grandes existen múltiples alternativas, entre las que destacan el procesamiento paralelo masivo, donde se busca repartir el trabajo entre miles de líneas de ejecución simultáneas, y ejecutado por ejemplo, en procesadores gráficos, y la computación distribuida, donde la carga se reparte entre múltiples equipos conectados en red.[22]

La metodología preferida a la hora de tratar con Big data es la computación distribuida, pues permite el escalado de forma continua, mediante la adición de nuevos equipos en red. Por otro lado, la opción paralela puede ser útil para ciertos problemas, pero presenta la problemática de que si el problema crece lo demasiado, la única solución sería crear equipos más potentes con capacidad para mover más hilos de ejecución al mismo tiempo, lo cual presenta una limitación en sí mismo. Por ello, vamos finalizar el capítulo con unos conceptos básicos de distribución distribuida.

La computación distribuida estudia pues, sistemas en los que distintos componentes, localizados en redes de ordenadores, se comunican y coordinan mediante el paso de mensajes, buscando una meta común. Tres características importantes en estos sistemas son la concurrencia de componentes, la falta de un reloj global que los coordine y el fallo independiente de los distintos equipos y componentes que los forman. Son por lo tanto, los siguientes aspectos, de suma importancia en los sistemas distribuidos.

* **Tolerancia a fallos:** Todos los sistemas fallan tarde o temprano, siendo mayor la probabilidad de que alguno de los componentes fallen al aumentar el número de estos. Es por ello necesario diseñar los sistemas distribuidos para que puedan recuperarse en caso de fallo de alguno de estos componentes.
* **Estructura del sistema:** Debido a que los costes de transmitir datos entre sistemas es mucho mayor a la transmisión dentro de estos, es necesario diseñar estructuras y algoritmos que minimicen la comunicación necesaria así como la distancia que esta información debe recorrer.
* **Vista limitada en cada componente:** Una de las ventajas de los sistemas distribuidos es que pueden manejar cantidades de datos mucho mayores que un solo equipo, pero para ello cada componente dispone solamente de parte de estos datos. Es por ello necesario a la hora de diseñar algoritmos distribuidos, hacerlo de forma que sea posible y eficiente trabajar solo con esa parte de los datos.

En cuanto a las arquitecturas utilizadas para la construcción de sistemas distribuidos existen multitud de opciones. A bajo nivel es necesario interconectar múltiples equipos con alguna red, mientras que a un nivel más alto es necesario que los procesos ejecutados en esos equipos se interconecten mediante algún sistema de comunicación. Los siguientes, son ejemplos de arquitecturas básicas de los sistemas distribuidos:

* **Cliente-servidor:** Esta organización consta de una serie de clientes que realizan el procesamiento distribuido y un servidor que organiza las tareas y datos de cada cliente. Este es el sistema utilizado por ejemplo por Apache Spark.
* **N capas:** En estos sistemas se mueven los procesos “inteligentes” a una capa intermedia de forma que clientes sin estado se puedan conectar a ellos, lo cual facilita el despliegue de aplicaciones. Este sistema es el usado por muchas aplicaciones web.
* **Peer-to-peer:** En estas arquitecturas no existe ninguna máquina que organice el trabajo, en su lugar todas las responsabilidades se reparten entre los distintos equipos que pueden servir tanto como clientes como servidores.

En sumario, las arquitecturas distribuidas, permiten el escalado de problemas gracias a dividir este entre múltiples sistemas, pero al mismo tiempo requiriendo una carga adicional debido al coste de las comunicaciones. Al mismo tiempo, estos sistemas presentan la necesidad de diseñar algoritmos que trabajen en cada momento con subconjuntos de los datos totales, provocando que en muchos casos los algoritmos utilizados presenten una mayor complejidad computacional. Finalmente, al ser estas arquitecturas necesarias para trabajar con Big Data, y requerir esta ultima de la utilización de algoritmos lo más óptimos posibles por el tamaño de la entrada que estos tendrán que procesar, se puede observar de donde viene la complejidad inherente a procesos como los que se desarrollan en este proyecto.

## 4. HERRAMIENTAS Y MATERIALES

### 4.1. Análisis de alternativas

Al centrarse nuestro proyecto en los campos del Big Data y el aprendizaje computacional se necesitan sistemas de almacenamiento y de procesado distribuidos, pues estos nos permiten escalar el tamaño de los problemas a los niveles necesarios. MapReduce es una opción recomendable para la manipulación de estas grandes cantidades de datos, pero está pensado para líneas de ejecución en las que los datos son leídos, modificados y escritos en una secuencia muy estricta. Además, al trabajar directamente sobre memoria secundaria, es muy eficaz para estas tareas y puede manejar grandes cantidades de datos, pero no es adecuado para los *pipelines* propios del aprendizaje computacional. Por ello son necesarias otras soluciones, como es la ofrecida por Apache Spark.

Apache Spark surgió como solución de código abierto para la realización de algoritmos iterativos sobre Big Data, y aunque no es la única opción disponible, es la más extendida. Existen también, otras opciones disponibles, como Apache Storm o Flink, aunque están orientados a análisis de datos en streaming, o DataTorrent RTS que pretende ofrecer tanto las capacidades de Spark y Storm así como las de MapReduce entre otras. Pero dado que el desarrollo de las librerías de aprendizaje automático de Spark está más avanzado, incluyendo librerías que incluyen los algoritmos que usaremos, la opción tomada es Apache Spark.

Es necesario también utilizar un sistema de almacenamiento distribuido, pero al ser Spark altamente compatible la lista de opciones es amplia, pudiéndose diseñar también un nuevo sistema que cumpla las especificaciones. La decisión tomada para este aspecto fue el uso de Apache Hadoop, ya que nos ofrece las especificaciones necesarias y tiene el plus de ser una tecnología altamente solicitada en el ámbito profesional, estando además incluida una implementación en el propio Apache Spark.

En cuanto a lenguajes de programación, Spark es completamente compatible con Java, Scala y Python. Y aunque tanto Java como Scala ofrecen un mejor rendimiento, Python ofrece una mayor facilidad para la programación de forma rápida e iterativa. Es también importante, que en Python se ofrecen múltiples cursos formativos sobre Spark, por lo que fue tomado como opción principal. Aun así ha de tenerse en cuenta que ciertos algoritmos incluidos en este trabajo serán lanzados desde Scala, por estar ya implementados en ese lenguaje y ofrecer mayor rendimiento, debiéndose a su vez mantener presente la opción de traducir el código a los distintos lenguajes disponibles.

Para el análisis de las secuencias de ADN se plantean también múltiples opciones, desde realizar el análisis basado en bases genéticas, a hacerlo con genes, o zonas más amplias del cromosoma. Esto se añade a las posibilidades de tratamiento de las distintas complicaciones presentadas en la sección correspondiente a genética. Aunque las elecciones relacionadas con este aspecto, se trataran con mayor profundidad en la sección de desarrollo del proyecto.

La última decisión a tomar en cuanto a las posibles alternativas no es otra que la metodología a seguir. Existiendo multitud de opciones que se adecuan a los distintos tipos de proyectos a realizar, en nuestro caso, se ha optado por la correspondiente al ciclo de vida del análisis de datos, el cual es utilizado con frecuencia en ámbitos relacionados con la ciencia de datos y el Big Data.

### 4.2. Metodología

Como acabamos de comentar, la metodología seguida en este proyecto es el *Data Analytics Lifecycle*, el cual fue diseñado específicamente para problemas relacionados con la ciencia de datos. Este ciclo de vida cuenta con seis fases y el trabajo puede situarse en un momento dado en varias fases a la vez. Además para la mayor parte del ciclo el movimiento entre las fases puede ser hacia delante o hacia atrás de modo que se asemeja a un proyecto real en el que nuevos descubrimientos realizados en una fase del proyecto pueden llevar al retorno a fases anteriores.

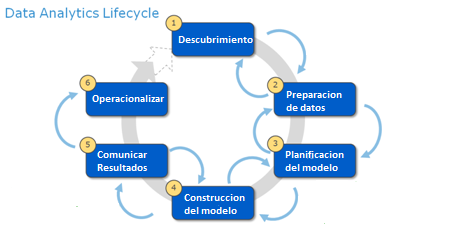


Figura 4: Esquema del ciclo de vida de análisis de datos

A continuación se muestra una descripción de las distintas fases del ciclo de vida:

* **Fase 1 – Descubrimiento:**

En esta fase el equipo de trabajo aprende el dominio del proyecto, si se han realizado proyectos similares en el pasado así como recursos disponibles, el tiempo necesario, etc. También se comienzan a realizar las primeras hipótesis y el aprendizaje de los datos.

* **Fase 2 – Preparación de datos:**

Durante la fase de preparación de datos, se realizan procesos de extracción, carga y transformación (ETL) para preparar los datos disponibles para la fase siguiente. Al mismo tiempo el equipo de trabajo debe familiarizarse con los datos a analizar.

* **Fase 3 – Planificación del modelo:**

En la fase de planificación del modelo se deciden los métodos, técnicas y flujo de trabajo que se pretende seguir durante la construcción del modelo, explorando al mismo tiempo las relaciones entre variables y la selección de aquellas que se consideran más favorables para el modelo.

* **Fase 4 – Construcción del modelo:**

El equipo debe desarrollar datasets para el testeo y entrenamiento, además de construir y probar modelos basados en la fase de planificación. Del mismo modo se puede observar si las herramientas necesarias son suficientes o se necesita el uso de otras nuevas para la ejecución de los modelos.

* **Fase 5 – Comunicación de resultados:**

A continuación el equipo debe comunicar los resultados a las partes interesadas y determinar junto a estos, sí el proyecto se considera un éxito o un fracaso según los criterios de la primera fase. Se deben identificar los descubrimientos clave, cuantificar el valor de negocio de los resultados, desarrollar documentación que sumarié los resultados, etc.

* **Fase 6 – Operacionalización:**

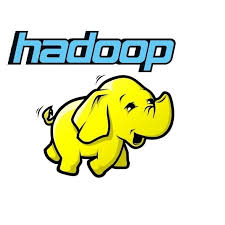
En la última fase el equipo entrega los reportes finales y documentos técnicos además es posible que el equipo lance un proyecto piloto para implementar los modelos en un entorno de producción.

Aunque se ha mostrado una explicación de la metodología seguida, en la siguiente referencia puede encontrarse información más detallada con respecto al ciclo de vida del análisis de datos, así como explicaciones en profundidad sobre cómo llevar a cabo cada una de las fases. [4]

### 4.3. Fundamentos tecnológicos

En esta sección, explicaremos las herramientas y arquitecturas utilizadas durante el desarrollo del proyecto, siguiendo las decisiones presentadas en el análisis de alternativas, para ello, trataremos el tema de Apache Hadoop, así como la metodología de Apache MapReduce que trabaja sobre este. A continuación se ofrece una introducción a Apache Spark, y a las librerías de esta plataforma utilizadas, para finalmente hablar de los fundamentos de los algoritmos de selección de características utilizados, que se basan en lo anterior y están, mayoritariamente, incluidos en librerías de Apache Spark.

#### 4.3.1. Apache Hadoop



Apache Hadoop[11]es un framework que permite el procesamiento distribuido de grandes conjuntos de datos a través de clusters de ordenadores. Está diseñado para escalar desde servidores individuales a miles de equipos, ofreciendo cada uno computación y almacenamiento local. En lugar de depender del hardware para obtener disponibilidad, la librería gestiona los fallos en la capa de aplicación, de forma que pueda ofrecer alta disponibilidad aunque los equipos sean susceptibles a fallos.

El sistema de archivos utilizado por este framework es conocido como Hadoop Distributed File System (HDFS), siendo este un Sistema distribuido diseñado para correr sobre hardware de bajo coste y altamente tolerante a fallos, proporcionando a su vez un alto rendimiento en el acceso a datos. Esta capacidad para correr sobre equipos de bajo coste, va acompañada porque HDFS está diseñado con la portabilidad en mente, siendo compatible con multitud de plataformas y fácilmente portable entre ellas.

En cuanto al aspecto de la tolerancia a fallos, hay que tener en cuenta que estos fallos de hardware son más una norma que una excepción, y dado que una instancia de HDFS puede constar de cientos de máquinas, con cada una de ellas almacenando una parte de los datos, es necesaria una rápida detección y recuperación automática de estos fallos, por lo cual esta es una de las principales metas de HDFS. Otro aspecto importante es la coherencia de los datos. Tomando esta como motivo, en HDFS un archivo creado, escrito y cerrado no puede cambiarse excepto para añadir nuevos datos al final o truncar parte de estos. Esto simplifica los problemas de coherencia y permite un alto rendimiento. MapReduce es un paradigma que encaja perfectamente con este modelo y que se tratara a posteriori.

En el ámbito del rendimiento, es importante también tener en cuenta que realizar cálculos sobre los datos es más rápido que moverlos, por lo que es necesario realizar los distintos cálculos lo más cerca posible de donde estén los datos. Para ello, HDFS proporciona interfaces para que las aplicaciones se muevan cerca de los datos que tienen que procesar.

La arquitectura que presenta HDFS es del tipo maestro/esclavo, donde cada cluster consiste en un único nodo de nombres, un maestro que controla el espacio de nombres del sistema y regula el acceso a los archivos por parte de los clientes y un número variable de nodos de datos, normalmente uno por cada nodo en el cluster. El nodo de nombres ejecuta funciones del sistema de archivos como abrir, cerrar o renombrar archivos y directorios, así como encargarse de determinar el mapeado de bloques de datos a los distintos nodos de datos. Estos últimos son responsables de servir las peticiones de lectura y escritura de los clientes.



Figura 5: arquitectura de HDFS

El NameNode y el DataNode están diseñados para funcionar sobre equipos de consumo, y aunque típicamente se lanzan sobre GNU están construidos usando java, por lo que pueden lanzarse en un amplio rango de sistemas. Por otra parte, la existencia de un NameNode único simplifica la arquitectura del sistema ya que contiene todos los metadatos, aunque al mismo tiempo puede suponer un cuello de botella.

Otro aspecto, es el espacio de nombres utilizado, concretamente HDFS utiliza una organización de archivos jerárquica similar a la de la mayoría de los sistemas de archivos, donde los usuarios pueden crear directorios y archivos dentro de estos. Además, el sistema soporta cuotas y permisos de acceso pero, aunque la arquitectura en si misma no prohíbe su implementación, esta no soporta enlaces fuertes ni débiles.

Como se ha dicho anteriormente, HDFS está diseñado para almacenar archivos muy grandes a través de varios equipos, para esto cada archivo se almacena como una secuencia de bloques y cada bloque se replica en distintas máquinas para conseguir tolerancia a fallos. Además todos los bloques de un archivo excepto el último, tienen el mismo tamaño y la aplicación puede especificar el número de réplicas, siendo todas las decisiones relativas a la localización y gestión de estas, llevadas a cabo por el NameNode.

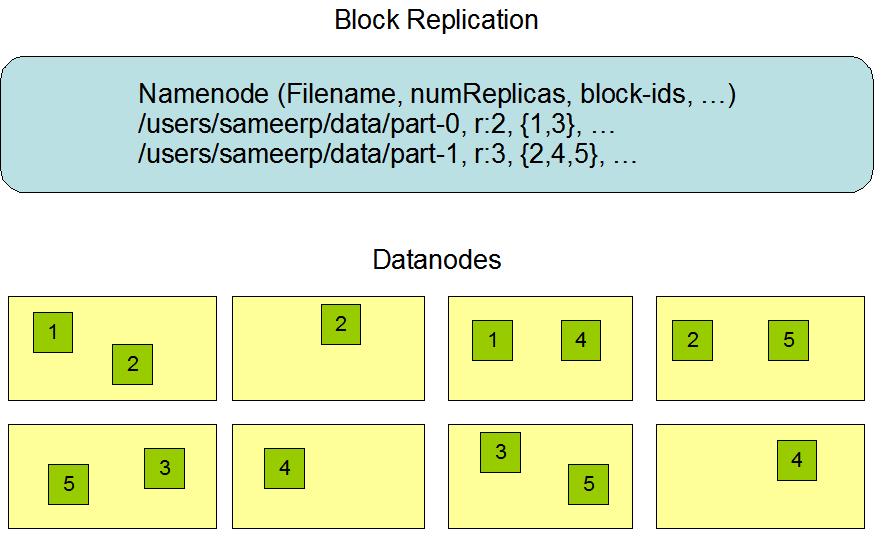


Figura 6: replicación en HDFS

La situación de estas replicas resulta crítico para el rendimiento y la fiabilidad de HDFS, las comunicaciones y transferencias de datos entre distintos racks necesitan pasar por switches que tienen un menor ancho de banda que las conexiones entre los equipos del mismo rack, por lo que mantener los datos cerca es importante para el rendimiento. Al mismo tiempo, a la hora de tratar con las réplicas es importante mantenerlos en racks separados para evitar que un fallo eléctrico en un rack haga inaccesibles todas las copias de los datos. Para ello, el NameNode utiliza un sistema que tiene en cuenta el rack en el que se encuentra cada bloque que consiste en limitar la colocación de estas a una réplica por bloque en cada rack.

#### 4.3.2. MapReduce



MapReduce [13] consiste en un modelo de programación así como una implementación asociada para el procesado y la generación de grandes conjuntos de datos de una forma paralela y distribuida sobre un cluster, pudiendo tomar ventaja de la localidad de los datos procesándolos en un lugar cercano a donde están almacenados reduciendo así la distancia que necesario desplazarlos.

En este modelo, los procesos se pueden dividir en tres etapas, un paso Map, en el que cada nodo trabajador ejecuta una función map sobre sus datos locales, almacenando la salida de forma temporal, mientras que un nodo maestro se asegura de que solo una copia de cada entrada es procesada. Un paso Shuffle, donde los nodos trabajadores redistribuyen los datos basados en las claves de salida producidos por el map, de forma que todos los datos que correspondan a una clave se localicen al mismo nodo y un paso Reduce, donde los trabajadores procesan cada grupo de datos en paralelo.

MapReduce permite el procesado distribuido de las operaciones de map y reduce siempre y cuando todos los mapeados sean independientes de los demás. Similarmente la fase de reducción puede realizarse por un conjunto de reductores siempre que cada uno reciba conjuntos de datos que no dependan de los demás conjuntos.

Cabe destacar que los programas en MapReduce no son automáticamente rápidos, siendo el mayor beneficio del modelo el uso de la función optimizada de shuffle y tener que escribir únicamente las partes de Map y Reduce del programa. En práctica, el programador debe tener en cuenta el paso de shuffle, en particular la forma de particionar los datos y el volumen de la salida de la función Map, pueden tener grandes efectos en el rendimiento.

Cuando se diseña un algoritmo en MapReduce, el autor también necesita valorar la relación entre el coste e cómputo y el coste de las comunicaciones, siendo normalmente este último el mayor cuello de botella. También se ha de tener en cuenta que muchas implementaciones de MapReduce escriben los resultados de los Mappers en memoria de forma distribuida para poder recuperarse en caso de error debiendo tenerse también en cuenta el coste que ello implica.

Otra cuestión es que para procesos en los que se desea un menor tiempo de respuesta y donde los datos cogen en la memoria de una sola maquina MapReduce no es efectivo. Del mismo modo, en situaciones en las que haya que realizar multitud de operaciones como puede ser el caso de los algoritmos de aprendizaje computacional, la carga de comunicación y de acceso a memoria secundaria sería un factor extremadamente limitante una solución para este aspecto es el uso de sistemas como Spark.

#### 4.3.3. Apache Spark

[](http://spark.apache.org/)

Apache Spark [12] es un framework de código abierto diseñado para realizar computación sobre clusters, y desarrollado por la universidad de Berkeley que posteriormente lo dono a Apache. Este framework ofrece una interface centrada en una estructura de datos llamada resilent distrubuted dataset (RDD), siendo este un multiset de solo lectura de objetos distribuidos sobre un cluster, que a su vez es mantenido de una forma tolerante a fallos. Se desarrolló como respuesta a las limitaciones de MapReduce, que forzaban un flujo de datos particular, para lo cual Spark utiliza una forma limitada de una memoria compartida distribuida.

Manteniendo los RDD en memoria principal Spark facilita la implementación de algoritmos iterativos y la realización de búsquedas sucesivas sobre los datos del RDD, pudiendo ser su latencia reducida por varios ordenes de magnitud cuando se compara con MapReduce. Entre los algoritmos iterativos mencionados se encuentran aquellos utilizados por el aprendizaje computacional, lo cual formo parte de la motivación inicial para diseñar Spark.

Para ejecutar este framework, es necesario un *cluster manager* y un sistema de almacenamiento distribuido. Como manager es soportado un modo local o standalone, el cual viene dado por un cluster nativo de Spark, así como YARN o Mesos. Para el sistema de almacenamiento se incluyen opciones como Hadoop distributed file system (HDFS), MapR-FS y cassandra entre otros, pudiendo implementarse también una solución personalizada. En cuanto a su estructura interna, el framework de Spark puede dividirse en varios componentes, el principal siendo Spark Core y junto a este se encuentran Spark SQL, Spark Streaming, MLlib y GraphX.

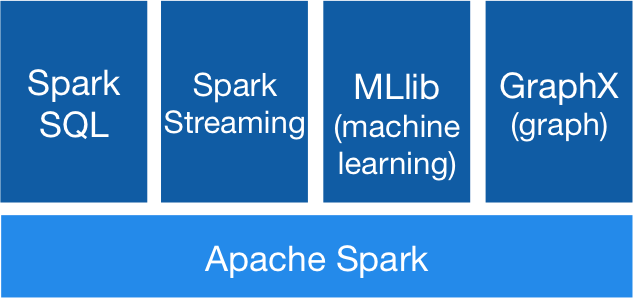


Figura 7: componentes de Spark

* **Spark Core:**

El núcleo de Spark es la base del proyecto, ofreciendo el manejo de las tareas distribuidas y su planificación. A su vez expone funcionalidades básicas de entrada y salida mediante una interface de programación disponible en Java, Python, Scala y R que se centra en la abstracción de los RDD. El funcionamiento de esta interface consiste en la aplicación de ciertas modificaciones como map, filter o reduce sobe RDDs , pasando funciones a al núcleo de Spark, que se encarga de planificar las transformaciones necesarias para su ejecución de forma paralela en el cluster. Estas operaciones, junto a algunas otras como la unión, toman RDDs de entrada y producen nuevos RDDs, siendo estos inmutables y ejecutándose todas las operaciones de forma perezosa. Dado que se sigue la pista a la secuencia de operaciones que se realizan sobre los RDDs, en caso de pérdida de datos estos pueden ser reconstruidos.

Además de la funcionalidad relacionada con los RDDs, Spark ofrece dos formas restringidas de variables compartidas, las variables de broadcast, de solo lectura y utilizadas para datos que necesitan estar disponibles en todos los nodos, y los acumuladores que actúan de forma similar, pero pudiendo incrementarse de forma controlada.

Es importante tener en cuenta la peculiaridad ofrecida por el hecho de que los datos de los RDD se en memoria, combinado con la estructura distribuida de spark, y esta es que los distintos trabajadores del cluster no pueden tener acceso a todos los datos, esto dio lugar a la necesidad de la creación de las variables compartidas pero estas solo deben usarse para pequeñas cantidades de información, dado que aunque un RDD puede ocupar Terabites o incluso más, esta memoria se encuentra repartida entre todos los trabajadores y el intento de traer más información a un solo equipo que la que puede almacenar provocaría un error. Es por ello también importante diseñar los algoritmos de modo que puedan trabajar únicamente con los datos que se encuentran en ese nodo concreto.

Un pequeño código de ejemplo seria:

EjemploRDD=(sc

*#Esta línea carga el archivo de ejemplo y lo parsea línea a línea en un RDD*

.textFile(“archivoDeEjemplo”)

*#Esta línea crea un nuevo RDD separando el string que contenía en una lista*

.map(lambda x: split(x,”,”) )

*#Esta línea pide que se mantengan los datos en memoria para futura referencia, aunque se hará cuando se calcule y recordemos que es perezoso*

.cache()

)

*#La función collect hace que se devuelvan los resultados al nodo central o driver, lo cual a su vez va forzar la evaluación de las líneas anteriores, pero, dado que collect pide que se almacenen todos los datos en un solo nodo dará error si estos no cogen en la memoria del driver, por ello collect no debería usarse en entornos de producción.*

Print(EjemploRDD.collect())

Figura 8: ejemplo de código en Spark

* **Spark SQL**

Spark SQL es un componente que se sitúa sobre el núcleo de Spark y proporciona una abstracción llamada DataFrames, que ofrece soporte para datos estructurados y semiestructurados, así como acceso a base de datos y funciones típicas de estas mediante un lenguaje específico del dominio disponible en Scala, Java y Python.

Una de las ventajas del uso de DataFrames cuando este es posible, es que el rendimiento apenas depende del lenguaje utilizado, pudiendo seleccionar el que resulte más atractivo sin preocuparse por las diferencias de rendimiento entre estos.

* **Spark Streaming**

Spark Streaming es un componente diseñado para ejecutar análisis de datos en streaming recibiendo pequeños bloques de datos y realizando transformaciones de RDD sobre estos bloques. Está pensado para que el mismo código que se usa en modo bach pueda ser usado en streaming pero tiene el inconveniente de introducir una latencia equivalente al tiempo de captura de este bloque de datos, a diferencia de otros sistemas que procesan evento a evento.

* **MLlib Machine Learning Library**

Spark MLlib es un framework distribuido de aprendizaje computacional ejecutado sobre Spark Core, tomando las ventajas de la memoria distribuida de Spark y presentando un rendimiento hasta 10 veces superior a otros métodos que trabajan sobre memoria secundaria y ofreciendo a su vez una gran escalabilidad. Muchos algoritmos comunes de aprendizaje máquina y estadística han sido ya implementados y están incluidos en MLlib lo que simplifica el diseño de pipelines de aprendizaje máquina.

En cuanto a estadística podemos encontrar algoritmos de estadística descriptiva, correlaciones, test de hipótesis, generación de números aleatorios, etc. Mientras que en el campo del aprendizaje computacional encontramos clasificación y regresión, filtrado, clustering, reducción de dimensionalidad, extracción de características, etc.

* **GraphX**

GraphX es un framework distribuido para el procesamiento de grafos basado en Apache Spark. Al utilizar RDDs y ser estos inmutables los grafos también lo son y por lo tanto su uso no está adaptado a grafos que necesiten actualizarse. GraphX proporciona dos APIs separadas para la implementación de algoritmos paralelos. Una abstracción de Pregel, que podría definirse como una tecnología de computación iterativa sobre grafos, y una API más general del estilo de MapReduce.

Para concluir, y ahora que conocemos los distintos componentes que lo forman, realizaremos una comparativa entre los distintos lenguajes utilizados con apache Spark. Como se ha visto anteriormente Spark ofrece la mayoría de sus funcionalidades en varios lenguajes de programación, cada uno con sus ventajas y desventajas a tener en cuenta, principalmente que Scala, siendo el lenguaje nativo de Spark ofrece un mayor rendimiento, mientras opciones como Pyspark ofrecen un entorno más iterativo para la programación pero un menor rendimiento. Esta diferencia de rendimiento puede observarse tanto en código genérico que se ejecute junto a Spark como en aquel directamente aplicado sobre los RDDs, mientras que el caso de los Dataframes, presentan un rendimiento más cercano entre lenguajes.

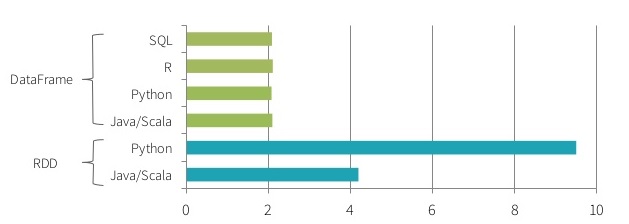


Figura 9: Comparación de rendimiento y lenguaje en Spark



Dado el uso extensivo de PySpark como lenguaje de programación, y en especial de la librería Mllib, es necesario hacer un análisis en mayor profundidad de la propia librería así como de los componentes usados. Como hemos explicado anteriormente, este paquete presenta multitud de métodos usados comúnmente en aprendizaje computacional dividiéndose a su vez en varios sub módulos:

* **Módulo mllib.feature:** Este módulo contiene las funciones necesarias para la manipulación de atributos, como normalizado y escalado. También cuenta con funciones para calcular el idf, usado en recuperación de información, así como funciones de selección de características. Entre estas últimas está incluido el ChiSqSelector que es usado en este proyecto, mientras que los otros algoritmos de selección de características usados mRMR y Relief-F, están implementados como parte de este módulo, pero no están incluidos en la distribución por defecto de este en el momento de realización de este trabajo, pudiendo encontrarse respectivamente en las siguientes referencias [8] [9].
* **Módulo mllib.regression:** Este módulo incluye diferentes modelos de regresión, y junto a estos la estructura de datos LabeledPoint, que es usada en este proyecto, representando un punto en un espacio multidimensional, con una etiqueta y un grupo de atributos.
* **Modo mllib.linealg:** El módulo de algebra lineal, contiene distintas funciones y tipos de datos, como matrices o vectores, con ejemplos como vectores densos y dispersos. Dentro de este, existe un módulo para algebra lineal distribuida, que incluye estructuras y funciones para matrices distribuidas y matrices de filas entre otras. Estas se usaron en parte del proyecto por su eficiencia pero hay que tener en cuenta que no pueden ser utilizadas en modo local.
* **Módulo mllib.utils:** Este módulo de utilidades incluye funcionalidades como la carga de módulos java, así como utilidades propias para el machine learning como la carga y escritura de distintos formatos de archivo. En el caso presente se usan loadAsLibSVMFile y saveAsLibSVMFile, para la carga y almacenaje de archivos de LabeledPoints , que en forma de RDD de estos, es la entrada más común entre los algoritmos de aprendizaje computacional de MLlib.

Además de los ya mencionados, MLlib incluye otros muchos subpaquetes que no son utilizados en este proyecto, algunos de ellos incluyen modelos de aprendizaje automático como clasificación, clusterización y regresión o sistemas de evaluación entre otros. Para más información se recomienda la siguiente referencia. [14]

Durante el desarrollo del proyecto, se usaron también otras librerías de Python, destacando NumPy [7]. Este es un paquete altamente optimizado que se utiliza especialmente para el cálculo científico, incluyendo matrices n-dimensionales, funciones matemáticas y algebraicas importantes, e integración con código en C/C++ y Fortran. Concretamente en este proyecto se hace uso de estas matrices así como operaciones básicas sobre ellas y del componente de generación de números aleatorios, que resultan más útiles en nuestro caso específico que las versiones ofrecidas por MLlib. Existe además una conexión importante entre Numpy y la versión de MLlib para PySpark, ya que dada la eficiencia de esta librería y su uso extendido, muchos de los métodos de MLlib se programaron permitiendo recibir como entrada los tipos de datos propios de la librería Numpy.

#### 4.3.4. Algoritmos

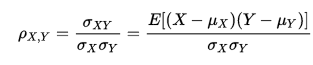
En esta sección, se mostraran los fundamentos de aquellos algoritmos de selección de características utilizados. Entre ellos tenemos el algoritmo ChiSquare, que está incluido en MLlib, así como RELIEF-F y mRMR que están incluidas en librerías adicionales, que a su vez están implementadas como parte de MLlib. Por el contrario, el caso del coeficiente de Pearson, no viene incluido, pero su implementación en Spark se puede realizar de forma eficiente y sencilla.

Además de esto podemos clasificar los algoritmos en dos tipos principales, siendo rankers monovariable el que utiliza el coeficiente de Pearson[23], ChiSquare[3,20] y Relief-F[2], y multivariable mRMR[10]. Pasamos pues a la presentación en este orden:

* **Pearson’s product-moment correlation coefficient:**

En estadística, el coeficiente de correlación de Pearson es una medida de la relación lineal entre dos variables, siendo a diferencia de la varianza, independiente de la escala de las estas.

Matemáticamente esta se define como:



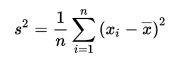
Dónde:

* **σxy** es la covarianza entre **X** e **Y**
* **σx** es la desviación típica de la variable **X**
* **σy** es la desviación típica de la variable **Y**

Por aclaración añadiremos que la desviación típica es la raíz cuadrada de la varianza, y esta a su vez se define en su caso continuo como:



Mientras que en el caso discreto se puede definir como:



Y finalmente la covarianza entre dos variables X e Y se define:



En nuestro caso, se realizara este cálculo sobre RDDs, y dado que la implementación no viene incluida, se añade a continuación el proceso resumido seguido en su implementación.

Algoritmo 9:

Cargar RDD inicial

Calcular una lista con la media de cada variable del RDD

Calcular una lista con las desviaciones típicas de cada variable del RDD

Calcular una lista con el coeficiente de cada columna del RDD con la etiqueta

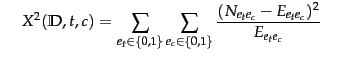
Ordenar la lista y obtener las variables más significativas

* **ChiSquare:**

El segundo algoritmo de selección de caracteristicas usado es el basado en ChiSquare, siendo este parte de la librería estándar de mllib. Y para su ejecución, lanzado sobre un RDD de LabeledPoints y recibiendo como parámetro el número de atributos a seleccionar. Esta función se basa en el test estadístico Chi2 que hace uso la distribución del mismo nombre. Calculando este estadístico entre cada variable y objetivo, podemos deducir si estas son independientes o no.

Matemáticamente, dos eventos A y B son independientes si P(AB) =P(A)\*P(B) o de forma equivalente si P(A|B) = P(A) y P(B|A) = P(B).

En selección de características los dos eventos son la ocurrencia del termino y la ocurrencia de la clase, y podemos realizar un ranking de los términos con respecto al resultado del siguiente calculo:



Donde N es la frecuencia observada en D, E es la frecuencia esperada, y et y ec se definen como 0 si la variable es binaria y vale 0, u es nominal y no está presente, y 1 si es binaria y vale 1, o está presente siendo nominal.

* **RELIEF-F:**

El algoritmos Relief-f , está incluido en una librería externa, que incluye una implementación distribuida para Spark tanto de este como de otros algoritmos, y aunque requiere el uso de mllib no está incluida en esta por defecto.

RELIEF-F es un estimador heurístico construido sobre el algoritmo RELIEF, que trata eficientemente con conjuntos de datos incompletos y con ruido así como con problemas mutilase. Funciona localizando los vecinos más cercanos para cada instancia de la misma clase y la opuesta, actualizando los pesos para cada atributo adecuadamente.

Más concretamente el algoritmo recorre un conjunto de instancias, encontrando para cada instancia sus k vecinos más cercanos pertenecientes a la misma clase, llamados aciertos cercanos, y los k vecinos más cercanos de clase diferente, que son llamados fallos cercanos.

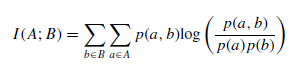
Cuando todos los vecinos han sido encontrados, se actualiza el peso para cada atributo, restando la distancia media ponderada (mediante la distancia Manhattan) de cada acierto, y sumando la distancia media ponderada de cada una de las otras clases. Al calcular estas distancias, estas se ponderan además por la probabilidad de la clase.

En la siguiente referencia [2], puede encontrarse más información sobre los distintos algoritmos en la librería utilizada, incluyendo RELIEF-F, así como el pseudocodigo de una implementación paralela de este.

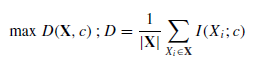
* **mRMR (fast-mRMR):**

El último algoritmo de selección de características usado es el mRMR, incluido en una librería adicional escrita en Scala y disponible públicamente en [8]. Aunque no se incluirá el pseudocodigo de este en este documento, tanto este, como un análisis en mayor profundidad puede encontrarse en [10]

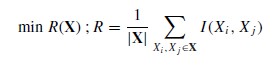
El algoritmo mRMR es considerado un filtro multivariable, usándose para realizar un ranking de la importancia de un conjunto de atributos para una tarea de clasificación dada. El ranking es calculado según la relevancia que cada uno de estos atributos presenta con respecto a la salida, al mismo tiempo que se penaliza la redundancia de estos atributos. El principal objetivo es encontrar la máxima dependencia entre un conjunto de atributos X y la clase c, usando para ello información mutua y se representa del siguiente modo.



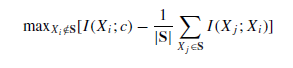
Dado que el cálculo de este criterio de máxima dependencia resulta extremadamente costoso, sobre todo en espacios de alta dimensionalidad, un modo alternativo es determinar el criterio de máxima relevancia, que consiste en buscar aquellos atributos que cumplen lo siguiente:



El problema de seleccionar atributos según este criterio es que puede implicar mucha redundancia, y para contrarrestar esto se añade el siguiente criterio de mínima redundancia:



La combinación de ambos criterios da el criterio mRMR que en la práctica se puede implementar como un algoritmo voraz.



En versión original de mRMR, el principal cuello de botella es el cálculo de la información mutua entre los posibles grupos de elementos. La versión utilizada en este proyecto, Fast-mRMR, utiliza el número de atributos a seleccionar para limitar el número de comparaciones entre atributos, y dado que mRMR es considerado como un algoritmo de ranking, esta búsqueda no afecta al resultado final, siempre que el interés se presente en esos n primeros atributos, siendo este el caso en el que nos encontramos.

### 4.4. Entorno de trabajo

Tras haber explicado las distintas herramientas y metodologías utilizadas se muestra a continuación un resumen del entorno de trabajo:

* **Apache Spark:** Como plataforma de desarrollo.
* **HDFS:** Como sistema de archivos para Spark, tanto en modo cluster como en modo standalone.
* **PySpark:** Usando Python3 como lenguaje principal de desarrollo.
* **Scala:** Como lenguaje secundario para algunas librerías.
* **Jupyter Notebook:** Como entorno local e iterativo de programación y ejecución con PySpark
* **Eclipse:** Como entorno de programación en Scala
* **Databrick community edition:** Como entorno adicional online de programación así como cluster distribuido de Spark sobre HDFS.

Además de esto, los siguientes son los datos del sistema local:

* **Windows 10**: Como sistema operativo local, siendo todo la anterior compatible también con Unix.
* **Equipo personal**:
  + Procesador Intel i7-4700MQ a 2.4GHz
  + 16 GB RAM

Es importante mencionar, que Apache Spark está diseñado para su uso en entornos distribuidos, pero en este proyecto se hace uso tanto de ese modo normal de funcionamiento como del modo local. Al mismo tiempo hay que tener presente que el código local, es ejecutable en casos distribuidos y esta optimizado para ello.

### 4.5. Datos utilizados.

Terminando el capítulo de materiales, pasaremos a enumerar los datos iniciales con los que se abordó el proyecto, viéndose más detalles sobre su uso en la sección de desarrollo. Estos incluyen:

* **Datos de genomas:** Obtenidos del ftp de *1000 genomes proyect* [5], de donde se descargaron 24 archivos VCF, correspondientes cada uno a un cromosoma, y que cuentan con las llamadas variantes de 2504 genomas. El tamaño total comprimido de estos archivos es de 16.2 GB y su ratio de compresión en torno a 50-1.
* **Mapeado de genes y posiciones:** Este archivo contiene los puntos de inicio y fin de cada gen mapeado hasta la fecha, compuesto por un archivo en formato TSV obtenido de UCSC [15], ocupando 1,4 MB y contando con las posiciones de 62251 genes.
* **Datos sobre el cáncer de mama:** En cuanto al cáncer de mama, usaremos las tasas de influencia de distintos genes en la patología, incluyendo estos la tasa de aparición normal así como las tasas modificadas según la influencia de cada uno de esos genes. [16]

## 5. DESARROLLO

En el presente capitulo se abordara la explicación del proceso de realización de este proyecto, comenzando con una fase de formación y recorriendo el proceso seguido mediante la metodología del ciclo de vida del análisis de datos. A continuación, se entrara en una subsección que pretende explicar en mayor profundidad ciertos aspectos de los tipos de análisis realizados en relación al tratamiento de la genética. Continuaremos con una explicación del proceso de manipulación y generación de datos, y terminaremos entrando en detalle en los principales algoritmos desarrollados para ello.

### 5.1. Fases del proyecto

Para la realización de este proyecto era necesario el aprendizaje de las tecnologías y metodologías necesarias poder abordarlo. Por ello, durante la fase inicial de este se realizaron una serie de cursos de formativos que, junto al seguimiento de diversos medios bibliográficos, pretendían servir como base para él mismo. La mayoría de estos cursos fueron tomados mediante la plataforma EDX, e incluyeron cursos sobre Big Data, así como Machine Learning y Python para complementar los conocimientos adquiridos durante el grado en esos campos. A continuación, se completó un conjunto de 3 Cursos de Apache Spark y Big Data ofrecidos por la universidad de Berkeley de forma online, en las que se utilizaban Databricks y PySpark, por una duración total de 160 horas.

Una vez completados estos cursos comenzaría la fase de descubrimiento del ciclo de vida del proyecto. Esta incluyó la búsqueda de datos de genomas, el estudio de los diferentes archivos, así como la adquisición de conocimientos relacionados con el campo de aplicación. Otro aspecto tratado durante esta fase inicial fue el inicio de la exploración de los posibles métodos de análisis a realizar, así como de los algoritmos de selección de características disponibles que pudiesen resultar eficaces para cumplir los objetivos propuestos.

Entre los problemas encontrados en esta etapa, se haya tanto el volumen de datos a obtener como el hecho de que, a aunque estos aumentan rápidamente, los datos de genomas completos disponibles son aun limitados. No siendo posible obtener genomas de pacientes con cáncer de mama haciéndose necesario utilizar generación de datos para simularlos.

Una vez se hubieron obtenido, se podía comenzar con la fase de preparación de datos. Siendo necesario comenzar la carga de los distintos archivos en Spark, así como la preparación de diversos procesos de ETL, con los que seleccionar los datos adecuados de aquellos disponibles y dándoles un formato que resultase adecuado.

Durante este proceso se encontraron multitud de problemas y cuellos de botella. Uno de ellos fue el inesperado coste de realizar la transposición de RDDs con los datos genéticos, que podrían entenderse como matrices de gran tamaño, con miles de filas y millones de columnas. Para solventar este problema fueron examinados distintos métodos, desde el uso inicial de transformaciones RDD del modo más optimizado posible, a combinaciones de estas con las matrices distribuidas ofrecidas por MLlib. Aunque finalmente, debido a las limitaciones de los entornos hardware utilizados aunque no de así de Spark, se tuvo que optar por la generación de datos, al menos durante esta fase inicial.

Tras completar las fases anteriores, entraríamos en la etapa de planificación del modelo, en la que sería necesario planificar como serían estos. Durante este proceso se encontro la posibilidad de explorar dos tipos de análisis, uno basado en bases genéticas otro sobre los genes que estas forman. Este aspecto es relativamente complejo, y para no entorpecer la explicación sobre las distintas fases, se abordara en su propia subsección inmediatamente después de esta. Las conclusiones de ello fueron que se debía realizar primero el análisis de bases genéticas usando distintos algoritmos disponibles, con la intención de probar su rendimiento en la mayor dimensionalidad que ofrecía esta opción. Una vez realizado ese análisis, se podría planificar un nuevo modelo y realizar análisis sobre genes.

Otro punto de esta fase de la metodología incluye la selección de aquellos algoritmos que utilizaríamos para probar el funcionamiento del modelo, donde se incluyen, como ya habíamos comentado, el selector ChiSquare, el coeficiente de correlación de Pearson, Relief-F y *Minimum Redundancy Maximum Relevance (*MRMR) .

Con el modelo planificado era el momento de comenzar la fase de construcción de este, pero para ello era primero había de completarse la preparación de los datos necesarios. Como habíamos observado, las limitaciones impuestas por los entornos de trabajo disponibles, se optó por la generación de automática de datos de prueba. Este proceso consto a su vez de dos fases, pues primero era necesario generar conjuntos de datos que simulasen genomas reales, para a continuación aplicar un algoritmo, que usando las distribuciones conocidas de la patología de estudio, añadiese los valores necesarios al conjunto anterior. Con estos los datos de prueba listos era el momento de probar los distintos algoritmos, midiendo tanto el rendimiento como la eficacia de con conjuntos de datos de distintas dimensiones.

Completadas las pruebas y observados los resultados, debía volverse a la fase de planificación del modelo con el objetivo de realizar el segundo bloque de análisis. Siendo este flujo algo esperado en este ciclo de vida, en el que los retrocesos y la superposición entre fases, resulta común. El retorno a esta etapa está marcado por el paso de bases genéticas al análisis de genes. Esto incluyo la toma de decisiones sobre cómo tratar diversos aspectos de su comportamiento, así como el diseño de las funciones y estructuras que permitiesen su mapeado. Como beneficio adicional de este proceso se encuentra la gran reducción en la dimensionalidad del problema.

Una vez más, se había retornado a la fase de construcción del modelo. Tras los resultados anteriores, había quedado claro cuál era el algoritmo más interesante y es que en el caso de uso presente aquellos algoritmos que se centraban en la selección de características de forma individual no podían competir con la precisión ofrecida por mRMR. Contando con que gracias a la reducción de dimensionalidad del paso a genes, los entornos utilizados eran capaces de trabajar con los datos de genomas reales. Se crearon pues los procesos ETL necesarios, incluidas funciones y estructuras para realizar el mapeado necesario, y aunque es cierto que ahora estos procesos podían completarse, seguían requiriendo tiempos de procesado de en torno a las 15 horas por cromosoma. Con estos archivos listos, se añadieron a ellos datos generados según las distribuciones de nuestra patológica de estudio, pudiendo verse resultados de las distintas etapas en el capitulo correspondiente.

El ciclo de vida utilizado presenta dos últimas etapas. La definición de la etapa de comunicación de resultados encaja perfectamente con el desarrollo de este documento, así como de la futura defensa que lo acompañará. Por el contrario, la fase de despliegue puede considerarse como parte del trabajo futuro a realizar.

### 5.2. Los dos modelos de análisis: bases genéticas vs genes

En esta subsección trataremos en mayor profundidad los aspectos del desarrollo relacionados con el análisis de datos genéticos, donde cabe destacar que las decisiones a tomar no son triviales y presentan multitud de problemáticas. Entre ellas nos encontramos con cuestiones que van desde cómo tratar con los múltiples alelos y los genes recesivos y dominantes, hasta las diferencias genéticas de género. Siendo otra decisión que afecta también a los aspectos anteriores el hecho de tratar con bases genéticas o genes.

Sobre este último aspecto, en el caso de la utilización de bases genéticas tenemos la ventaja de mantener toda la información disponible, dado que aunque dos bases pertenezcan al mismo gen y puedan afectar a la misma patología, la alteración final en la proteína codificada será distinta, y del mismo modo, diferente el efecto y la intensidad de este. Al mismo tiempo esta elección presenta puntos negativos, como el hecho de tener que trabajar con una dimensionalidad muchísimo mayor y al mismo tiempo perder la capacidad de generalización, necesitándose además conjuntos de datos con un numero de muestras muchísimo mayor. Siendo esto último debido a que si pretendemos encontrar todas las bases que afectan a nuestro objetivo, sería necesario al menos, que todas estas relaciones estén presentes y sean apreciables, y teniendo una diferencia media entre genomas de solo el 0.1%, podemos hacernos a la idea de lo que esto supondría.

Por otro lado, la opción de mapear estas bases a genes elimina estos dos problemas, ofreciendo una dimensionalidad muchísimo más abarcable, pudiendo tomar como cota superior los 100.000 atributos. Este proceso favorece también la generalización, dado que como se mencionó con anterioridad, un error en un gen dado afectara, de un modo u otro, al comportamiento de ese gen, y por lo tanto a la las funciones reguladas por la proteína que codifica. Como punto negativo, esto también significaría que el sistema seria menos preciso, dado que sería imposible detectar cual sería la variación exacta provocada por una mutación simplemente conociendo que esta se encuentra en algún punto de un gen concreto. Aunque como contrapunto, esto no tendría importancia si lo que buscamos es únicamente saber que la etiqueta de salida es influenciada por la existencia de mutaciones en ese gen, sino que sería una cuestión que aparecería en el momento de detectar la presencia de la enfermedad a partir de esta y eso podría solucionarse con otros métodos como por ejemplo el uso de análisis en fases, detectando con el sistema aquí desarrollado que genes presentan influencia en la salida, y a posteriori analizando las bases que forman esos genes influyentes, que variaciones concretas ejercen que efecto.

Existen algunas otras cuestiones a tener en cuenta, siendo una de ellas la relación entre genes dominantes y recesivos, dado que esta afecta también a la interpretación, tanto con bases, como con genes. Esto se refiere más específicamente a los múltiples alelos que forman el genoma. Normalmente, podríamos almacenar si este alelo esta modificado en un caso, ambos o ninguno, pero al combinar las bases en genes, necesitamos decidir si interpretar de forma distinta que ambas copias del gen tengan una modificación distinta, o que la modificación sea la misma para ambos genes.

Muchas de estas decisiones son aun concepto de debate entre la comunidad científica, pero en nuestro caso podemos suponer que incluso ignorando esas diferencias, los algoritmos de detección de características deberían detectar igualmente que existe alguna influencia. Por ello, se ha optado por una versión simplificada, filtrando los valores en la última fase para tener solamente en cuenta si el gen presenta modificaciones o no, e ignorando los alelos.

### 5.3. Generación y manipulación de datos

En la presente sección ofreceremos una explicación de los procesos de generación de datos, así como las transformaciones realizadas sobre ellas, para terminar con la presentación de los distintos algoritmos utilizados para ello.

Comenzamos pues, con datos generados para le realización de pruebas, donde contamos con dos casos principales

* **Generación de datos de genomas:** Este proceso utiliza la librería numpy de Python con la función de generar RDDs que simulen genomas en el formato necesario para los algoritmos de selección de características.
* **Generación de datos del cáncer de mama:** El segundo ejemplo de generación de datos, es llevado a cabo mediante un algoritmo que haciendo uso de las distribuciones conocidas sobre esta enfermedad, genera datos que corresponden a estas. Este algoritmo se utiliza luego en una función map sobre un RDD para añadirlos a cada fila datos generados específicamente para ella.

A continuación, trataremos el proceso de carga y transformación de datos, el cual supuso el mayor cuello de botella del proyecto. La primera fase de estos ETL consiste en cargar los datos de los archivos VCF, para a continuación parsearlos a listas que incluyen la posición de una base genética y los valores de alteración o no para cada uno de los genomas incluidos. Una vez realizado este proceso, era necesario realizar la transposición del RDD resultante, lo cual suponía tanto tener todos los datos del cromosoma específico cargados en memoria y realizar la transposición de este. Los problemas se tornan más graves si tenemos en cuenta el tamaño de cada uno de estos archivos, que sin estar comprimidos alcanzan volúmenes superiores a los 20GB y que calcular la traspuesta de este RDD supone un alto computacional, al que hay que añadir los posibles costes de comunicaciones en caso de un entorno distribuido.

El uso de transformaciones RDD para realizar esta tarea parecía la solución lógica, pero las pruebas realizadas al respecto daban como resultado tiempos de ejecución excesivamente grandes, normalmente no llegando a completarse las tareas debido a los límites en la capacidad de los entornos utilizados. Ante esta problemática se probó también el uso de matrices distribuidas construidas a partir de RDDs de coordenadas. Este componente de MLlib ofrecia mejores rendimientos, pero su utilización no es posible en modo local, y el cluster distribuido no aceptaba, en su version gratuita, archivos superiores a 200 MB. De todos modos, de esto se extrajo de realizar una primera fase de transformación en coordenadas seguido de un filtrado para eliminar aquellas entradas de valor 0 y construyendo con el resto el RDD traspuesto. Esta opción ofrecía mejor rendimiento que la primera aproximación, pero seguía suponiendo demasiada carga a la hora de trabajar con los archivos de datos reales. Por ello, fue necesario el uso de datos generados para el análisis de bases genéticas que simulasen datos ya transformados. El siguiente paso del ETL, consistió en el mapeado de este RDD con la función de generación de datos correspondientes al cáncer, dejando listos los archivos necesarios para poder probar los algoritmos de selección de características.

Tras haber completado la primera etapa, se pasó a la siguiente fase, la cual implicaba el análisis de genes. Para ello fue necesario diseñar e implementar nuevas funciones. En este caso las funciones necesarias eran aquellas que realizaban la extracción de las posiciones de genes, la creación de una estructura de datos para almacenarlos y del mapeado de posiciones de bases a las listas de genes que son afectados por estas posiciones. Con estas funciones completadas se pudo proceder a realizar transformaciones sobre los datos originales, realizando el mapeado a genes, filtrando aquellos valores que no pertenecían a ningún gen, y explosionando las restantes para separar las posiciones que afectaban a varios genes a la vez en entradas diferentes del RDD resultado. Luego las entradas del mismo gen, se combinaron entre ellas usando la función or, elemento a elemento. Este proceso redujo los datos a niveles computables en el entorno local, pero los tiempos necesarios eran del entorno de las 15 horas por archivo. Finalmente los resultados se almacenaron en memoria para futuro uso.

El último proceso de ETL realizado, consistió en la carga de los archivos generados en el paso anterior, para agruparlos y mapearlos, como en el caso anterior, para valores restantes, almacenando nuevamente los resultados para proceder a la realización de pruebas.

### 5.3. Algoritmos usados.

Para completar el capítulo se mostrara a continuación el seudocódigo simplificado de los principales algoritmos utilizados para la generación de datos y las transformaciones de estos. A estos habría que añadir los algoritmos de selección de características usados en la fase de pruebas, que fueron ya explicados en el capítulo correspondiente a herramientas y materiales.

* **Generación de RDDs aleatorios de prueba.**

Este algoritmo, recibe el número de columnas y filas deseado y devuelve un RDD de listas de 0s y 1s, con la distribución deseada, habiéndose seleccionado un 5% de 1s.

Algoritmo 1: Generación de RDDs de valores aleatorios

Seleccionar nº de elementos por fila

Seleccionar nº de filas

Crear lista de 0s del mismo tamaño que el nº de filas

Crear un RDD con esa lista

Mapear cada fila de ese rdd a una lista aleatoria de 0s y 1s de tamaño igual al nº de elementos por fila con la distribución indicada.

Devolver el RDD resultado

* **Generación de datos según distribuciones del cáncer de mama.**

Para la generación de los datos correspondientes al cáncer de mama, se utiliza una función que recibiendo una tupla de probabilidades, genera una lista valores aleatorios con respecto a esas probabilidades.

Algoritmo 2: Generación de datos según distribución de entrada

Crear lista de valores aleatorios entre 0 y 1 del tamaño deseado.

Modificar los dos primeros valores según la probabilidad de alteración.

Modificar el tercer valor según la probabilidad indicada para esos valores 0 y 1

Modificar el cuarto valor según el corte de edad.

Modificar el quinto valor según los valores 3 y 4

Devolver la lista resultado.

El resultado del algoritmo es una lista en el que los valores se corresponden con:

1. Gen BCRA1 modificado
2. Gen BCRA2 modificado
3. Si la persona desarrollará cáncer
4. Edad de la persona, discretizada.
5. Etiqueta, presencia o no de cáncer.

En este caso existe la cuestión lógica de porque no se incluyen las diferencias en género en estas pruebas, siendo la respuesta que ofrecerían el mismo caso de test que ofrece el valor de edad, manteniendo la etiqueta si fuese 1, o femenino, y dividiendo la tasa de aparición entre 10, si fuese 0 o masculino.

* **Generación de diccionario de genes.**

Esta función, se encarga de generar una estructura de datos para el futuro mapeado de posiciones a genes.

Algoritmo 3: Generación del diccionario de genes

Cargar Archivo de posiciones de genes

Seleccionar de este archivo posición inicial, final e índice del gen

Generar lista ordenada de posiciones inicial y gen

Generar lista ordenada de posiciones final y gen

Crear dos índices para las listas

Crear lista de salida

Crear diccionario auxiliar

Para cada posición de la lista de posiciones iniciales:

Actualizar índice de posiciones iniciales

Para cada posición entre el índice de posiciones finales y el de iniciales:

Si termina algún gen antes de la posición inicial actual:

Insertar la posición de terminación y los genes que

terminan ahí en la lista de salida.

Eliminar del diccionario los genes terminados.

Ajustar el índice de la lista de posiciones finales para la seguir

desde donde termino el último gen.

Si la última posición de la lista de salida, es menor a la posición de la

lista de posiciones iniciales en la que estamos:

Insertar la posición de inicio y los genes que había hasta ahí en el

diccionario.

Añadir gen que empieza en la posición actual al diccionario.

En resumen, este algoritmo recorre dos listas ordenadas del mismo tamaño, generando una salida en la que, en cada posición que se produce un cambio, se indican los genes que son afectados por todas las posiciones anteriores, desde el último cambio. Además realiza este proceso con complejidad lineal O(n)

* **Mapeado de posiciones a genes**

El siguiente algoritmo, consiste en una versión modificada de Búsqueda Binaria, en el que si el valor no existe se devuelve el inmediatamente superior. Las entradas son un diccionario ordenado, y un clave a buscar.

Algoritmo 4: Mapeado de posición a genes

Si diccionario contiene la clave buscada:

Devolver su valor

Mínimo = 0

Máximo = tamaño entrada

Repetir

Pivote= punto medio entre máximo y mínimo

Si máximo y mínimo están pegados o son el mismo:

Devolver máximo

Si el valor en la entrada en la posición del pivote es menor al buscado:

Mínimo=pivote

Sino:

Máximo = pivote

Una opción de mejora de este algoritmo, sería el uso de diccionarios anidados. Realizando una descomposición del valor a buscar a modo de mascara y usando la primera parte de la máscara para seleccionar un diccionario interno en tiempo constante, y el resto para lanzar el algoritmo sobre un n que vendría dado por el nº de claves en ese diccionario interno.

* **Carga, filtrado y mapeo a genes del archivo de cromosomas**

A continuación se muestra el algoritmo utilizado para mapear genes a cromosomas, utilizando el diccionario generado anteriormente y la función de mapeo de posiciones a genes. Además de esto realiza filtrados para eliminar los valores correspondientes a ADN no codificante y realiza la combinación de los genes mediante funciones or.

Algoritmo 5: Carga, filtrado y mapeo a genes de archivos de cromosomas

Crear RDD con el archivo deseado

Filtrar filas de cabecera

Para cada fila seleccionar posición y lista de variaciones de cada genoma

Para cada fila, mapear la posición a la lista de genes que pertenece

Eliminar las filas que codifican ADN “basura” (ningún gen)

Generar para cada fila una fila por cada gen que modifique

Realizar una función OR, entre las filas que afecten al mismo gen

Devolver Resultado

* **Agrupación de RDDs**

El siguiente algoritmo simplemente une todos los archivos creados anteriormente en un solo RDD de mayor tamaño, permitiendo seleccionar que cromosomas queremos usar.

Algoritmo 6: Agrupación de rdds

Declarar RDD vacío:

Por archivo a cargar:

Si RDD está vacío:

Crear RDD con el archivo y asignarlo al RDD declarado

Sino:

Crear RDD con el archivo y unirlo al RDD almacenado

Finalmente nos encontramos con el algoritmo o transformación necesaria para realizar la transposición de los RDDs de genes.

* **Transposición de RDD**

Este último algoritmo se encarga de realizar la transposición de RDDs, mediante la separación de estos en coordenadas para reformarlos nuevamente del modo deseado. Para ello utiliza reducciones por clave que ofrecen mayor rendimiento que las agrupaciones y realiza filtrados intermedios para mejorar este aún más. Cabe recordar, que es exactamente este punto el que ofrece el mayor cuello de botella cuando se trata directamente con bases genéticas.

Algoritmo 8: Transposición de RDD de cromosomas

Añadir índices de posición a cada fila del RDD de entrada

Para cada fila:

Añadir índices de posición a cada valor de la fila

Mapear cada valor de la fila a una nueva fila de coordenadas –valor

Para cada fila de coordenadas invertirlas

Unir todas las filas que comparten la coordenada deseada en una sola fila

Ordenar cada fila internamente según la otra coordenada

Eliminar las coordenadas

## 6. RESULTADOS

### 5.1. Medidas de evaluación

A la hora de evaluar los resultados obtenidos, es importante el uso de diversas medidas de evaluación que se adapten tanto a las metodologías usadas como al caso de estudio desarrollado. En primer lugar se toma la medida de tiempo como muestra del rendimiento de los distintos algoritmos de selección, mientras que para comprobar la eficacia de estos se utilizara la precisión. A continuación se entra en mayor detalle en ambas medidas.

* **Eficiencia, Tiempo:**

A la hora de medir la eficiencia de los algoritmos tomaremos medidas de tiempo para los algoritmos de selección de características usados, estas medidas serán tomadas en el sistema local, pues el otro entorno utilizado presenta peor estabilidad debido a las diferencias en la carga de trabajo de este. En cuanto a la ejecución de los distintos algoritmos se utilizara Scala, ofreciéndose también una comparativa entre los tiempos de este lenguaje y Python. Pare esto último se ha utilizado el algoritmos ChiSquare, cuya implementación se incluye en las librerías por defecto para ambos lenguajes.

* **Eficacia, Precisión:**

Los algoritmos usados son lanzados con el objetivo de recuperar un subconjunto n de atributos, siendo n el número de atributos que afectan a la salida, y que al ser generados es conocido de antemano. Por ello, para la medida de eficacia deseamos saber cuántos de esos atributos son incluidos en el subconjunto devuelto, tomando la precisión a n como medida de eficacia. En cuanto a los datos utilizados, fueron utilizados conjuntos de prueba de distintos tamaños en ambos ejes, generándose para cada conjunto los valores influyentes y sus etiquetas siguiendo las funciones descritas anteriormente.

Para una mayor definición de los datos utilizados, los conjuntos incluyen en la fase de testeo con bases genéticas una distribución del 5% de alteraciones. Mientras que en la segunda fase se trata de datos reales.

### 5.2. Medidas de Tiempo

A continuación se presenta la tabla de tiempos de los distintos algoritmos de selección de características:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Muestras N | Atributos M | Pearson’s | ChiSquare | Relief-F | mRMR |
| 1 K | 1 K | 12 s | 0.8 s | 114 s | 5 s |
| 1 K | 10 K | 16 s | 4 s | 169 s | 29 s |
| 10 K | 10 K | 64 s | 35 s | Error | 139 s |
| 1 K | 100 K | 66 s | 200 s | Error | 123 s |
| 2 K | 100 K | 128 s | 573 s | Error | 268 s |
| 1 K | 2 K | 20 s | 1 s | 108 s | 9 s |
| 1 K | 20 K | 30 s | 5 s | 174 s | 35 s |
| 1 K | 50 K | 46 s | 22 s | 276 s | 70 s |
| 2 K | 1 K | 16 s | 0.5 s | 108 s | 4 s |
| 10 K | 1 K | 17 s | 2 s | 384 s | 14 s |
| 20 K | 1 K | 21 s | 3 s | 1440 s | 27 s |

Tabla 4: Comparación de tiempos entre algoritmos.

De estos datos se puede concluir que el mejor escalado es el ofrecido por el cálculo del índice de correlación de Pearson, seguido muy de cerca por fast-mRMR. Además podemos observar que los tiempos ofrecidos por la implementación usada del algoritmo Relief-F, presentándose a su vez un límite en el volumen de datos que este puede recibir, aunque esto último se debe a la implementación concreta ya hace uso de variables compartidas que en Spark están limitadas en tamaño.

La siguiente tabla ofrece una comparación entre los tiempos de ejecución en Scala y Python:

|  |  |  |  |
| --- | --- | --- | --- |
| Muestras N | Atributos M | ChiSquare Python | ChiSquare Scala |
| 1K | 1K | 7.5 s | 0.8 s |
| 1K | 10 K | 44 s | 4 s |
| 10K | 10 K | 205 s | 35 s |
| 1K | 100 K | 1740 s | 200 s |
| 2K | 100 K | 3900 s | 573 s |

Tabla 5: Comparación de tiempos entre lenguajes.

Los siguientes gráficos muestran la relación de tiempos según la variación en número de ejemplos y numero de atributos respectivamente, manteniendo el otro eje fijo a 1. El eje X esta en millares y el Y en segundos.

Figura 10: número de ejemplos vs Tiempo

Figura 11: número de atributos vs Tiempo

### 5.3. Medidas de Precisión.

Se pasa ahora a mostrar las mediciones de eficacia de los distintos algoritmos, mediante el uso de la precisión a n, donde n es el número de variables significativas. Empezaremos por los resultados sobre bases genéticas generadas.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Pearson | ChiSquare | Relief-F | mRMR |
| Datos de Test | Precisión | Precisión | Precisión | Precisión |
| 1Kx1K | 1 | 1 | 0.75 | 1 |
| 1Kx10K | 0.75 | 0.75 | 0.5 | 1 |
| 1Kx2K | 1 | 1 | 0.75 | 1 |
| 1Kx20K | 0.75 | 0.75 | 0.5 | 0.75 |
| 1Kx50K | 1 | 1 | 0.25 | 1 |
| 2Kx1K | 1 | 1 | 0.75 | 1 |
| 10Kx1K | 1 | 1 | 0.25 | 1 |
| 20Kx1K | 1 | 1 | 0.75 | 1 |
| 10Kx10K | 1 | 1 | NaN | 1 |
| 1Kx100K | 0.5 | 0.5 | NaN | 0.75 |
| 2Kx100K | 1 | 1 | NaN | 1 |
| Promedios | 0.91 | 0.91 | 0.56 | 0.95 |

Tabla 6: Medidas de eficacia 1.

Los datos anteriores presentan una alta dimensionalidad, pero aquellos que representan variables no significativas se generaron de forma aleatoria. Para observar el funcionamiento con datos reales mostramos a continuación mediciones tomadas sobre genes en la segunda fase de análisis.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Pearson | ChiSquare | Relief-F | mRMR |
| Datos de Test | Precisión | Precisión | Precisión | Precisión |
| Chr1 | 1 | 1 | 0 | 1 |
| Chr13 | 1 | 1 | 0.75 | 1 |
| Chr2 | 1 | 1 | 0 | 1 |
| Chr14 | 1 | 1 | 0 | 1 |
| Chr10-11 | 1 | 1 | 0 | 1 |
| Promedios | 1 | 1 | 0.15 | 1 |

Tabla 7: Medidas de eficacia 2.

Como se puede observar, el algoritmo que peor resultado ofrece es Relief-F, dejando claro que no presenta un buen rendimiento en el tipo de datos usados. Los índices de correlación de Pearson y ChiSquare presentan un rendimiento bastante bueno, pero al tratarse de filtros monovariable, solo detectan aquellos atributos que son identificables en solitario. Por otro lado mRMR detecta los efectos que los distintos atributos ofrecen en conjunción con otros, detectando casos como el efecto de la edad o el sexo, en las situaciones en las que el atributo en sí mismo no muestra relación con la salida, ofreciendo los mejores resultados de todos los algoritmos probados.

## 7. CONCLUSIONES Y TRABAJO FUTURO

Como se ha visto a lo largo del proyecto, el Big Data y la ciencia de datos ofrecen la posibilidad de extraer valioso conocimiento de fuentes de datos que hace poco resultaban inabarcables. Las nuevas tecnologías y métodos desarrollados para el tratamiento de estos datos son potentes, pero requieren a su vez de amplios periodos de aprendizaje, así como de profesionales que dominen multitud de campos. La necesidad de contar con conocimientos en plataformas como Spark, computación distribuida, estadística, aprendizaje máquina y otros muchos aspectos se acumula, ayudando a explicar la escasez de personas cualificadas para la realización de estos procesos. Esto se acumula a la gran cantidad de valor que los datos de distintos ámbitos puede contener, pero también a que las diferencias entre cada análisis realizado hacen, que junto a todos las técnicas anteriores, sea necesario adquirir un profundo conocimiento en el ámbito de aplicación de cada análisis a realizar.

Uno de estos campos de conocimiento es, como se ha visto, el de la genética. Este ámbito presenta multitud de problemáticas, entre las que se encuentran las comunes al Big Data, como es la alta dimensionalidad, pero también la complejidad de los datos en sí mismos. Entre ellos se encuentran la relación entre bases, genes y proteínas, el cómo distintos cambios en estas pueden ofrecer distintos resultados, la presencia de múltiples alelos y como tratarlos, la superposición de genes, el ser dominante o recesivo incluso dependiendo de otro gen, etc. Todos estos aspectos se acumulan y ayudan a explicar el porqué, de que el análisis de genes, no sea aun la norma.

A pesar de esto el horizonte brilla con fuerza, el mapeado del genoma de los individuos es cada vez más asequible, siendo también más fácil encontrar los datos necesarios para su análisis. Las empresas y servicios médicos muestran su interés en utilizar estas capacidades, y los procedimientos seguidos en el desarrollo de este proyecto muestran que es posible realizar estos procesos.

Los resultados obtenidos muestran como las distintas técnicas utilizadas, permiten aproximarse desde múltiples direcciones al análisis del ADN, y de cómo la tecnología actual está preparada para ello. Entre los distintos algoritmos utilizados se ha visto la potencia de los distintos tipos de métodos de selección de características, especialmente la precisión ofrecida, en este campo de estudio, por aquellos diseñados para la selección de subconjuntos mediante información mutua. El algoritmo mRMR no solo ha ofrecido grandes resultados en cuanto a precisión, sino que en su implementación fast-mRMR ha demostrado a su vez su buen rendimiento en cuanto a tiempo.

Se ha podido ver también las diferencias entre la aproximación desde el punto de vista de las bases genéticas y el de los genes, analizándose las ventajas y desventajas de cada uno. El análisis de genes nos ofrece mayor generalización, siendo óptimo para detectar aquellos que presentan influencia en aspectos como el aquí estudiado, mientras que el análisis de bases puede ofrecer la precisión de conocer exactamente que mutación es la que provoca el efecto concreto a buscar. La combinación de ambos procesos ofrece incluso más ventajas, pudiendo comenzarse con el análisis de genes tanto para obtener sus beneficios, como para reducir el tamaño del problema antes de analizar las bases. A su vez, el proceso inverso permitirá analizar genomas en busca de los posibles problemas que estos presentan, pudiendo ofrecer tratamiento preventivo y personalizado a todos aquellos que se beneficien de estos sistemas.

### 7.1. Trabajo Futuro

El trabajo futuro a realizar con respecto a este proyecto es muy amplio, mostrándose parte de estas a continuación, pero al mismo tiempo, una necesidad para el avance del proceso es la ampliación de los datos de genomas disponibles, que ya está siendo llevada a cabo por multitud de organizaciones y empresas.

* **Extensión del sistema:** Una de las futuras ampliaciones de este proyecto reside en la adición de distintos tipos de dato al proceso. Aspectos como la edad, el historial médico, el peso, los valores correspondientes a análisis de sangre o datos sobre la alimentación. Todos ellos pueden tener influencia en distintos aspectos de la salud y podrían ser combinados con el análisis del ADN para obtener un mayor valor.

Esto incluye la creación de un sistema de almacenamiento capaz de contener todos los datos necesarios, de recibir las aportaciones de nuevos mapeados de ADN y de ampliarse rápidamente para aceptar nuevos tipos de datos. Ejemplos como Big Table de Google ofrecerían las características necesarias para realizar este proceso. [18].

* **Sistema de detección de patologías:** Otra futura extensión consiste en la utilización de los resultados obtenidos por el análisis de genomas, en la creación de un sistema que utilice estos para, recibiendo un genoma como entrada, ofrecer un informe de todas las enfermedades, alergias o aspectos importantes que se deban tener en cuenta con respecto a ese genoma. Ofreciendo así las capacidades de prevención que el ámbito medico necesita.
* **Optimización de procesos de transformación de datos:** Las transformaciones necesarias para el análisis de los datos utilizados han supuesto, con diferencia, el mayor cuello de botella del proyecto, al menos en cuanto a computación se refiere. Esto requiere soluciones que incluyen tanto la optimización del proceso, como el almacenaje de los datos en forma que su selección y utilización requiera el menor número de transformaciones posible.

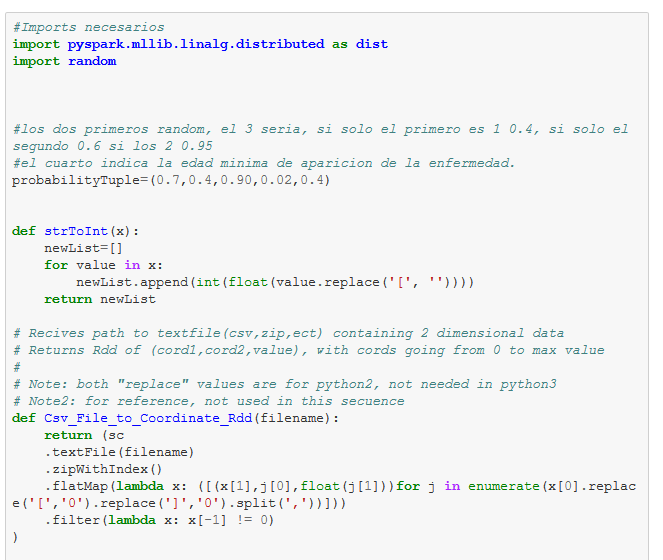
## **8. APÉNDICES**

Se incluye a continuación una copia del código utilizado para referencia, estando disponible también en forma digital como acompañamiento a esta memoria.

### 8.1. Código

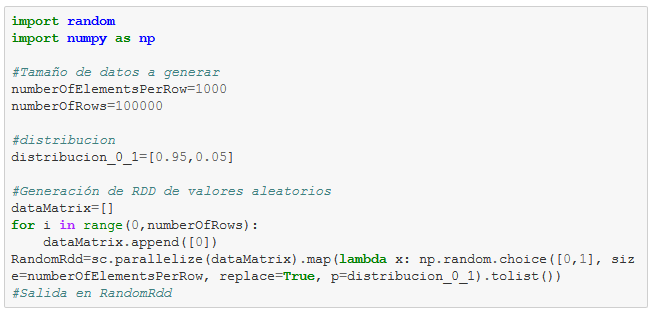
#### Fase 1: Creación de datos

La primera celda incluye las declaraciones de funciones auxiliares así como la función de generación de correspondientes a variables significativas para añadir al RDD base.





La siguiente celda incluye el código de generación de RDDs aleatorios.



La siguiente casilla utiliza el Rdd aleatorio en RandomRdd, genera los nuevos valores para cada fila y escribe la salida en la ruta de destino.

****

#### Fase 2: Creación del diccionario de genes

Este código genera un diccionario ordenado para mapear posiciones a listas de genes afectadas por esta.





Las siguientes funciones realizan el mapeo de posiciones a listas de genes usando el diccionario anterior

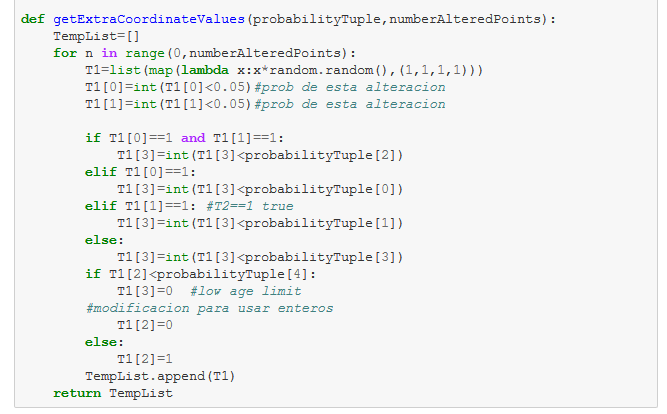
****

#### **Fase 2: Secuencia de preparación de datos reales**

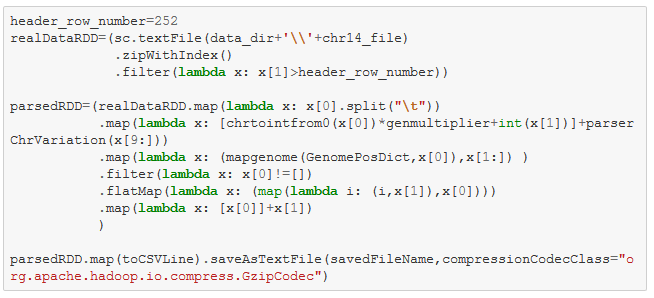
En la primera casilla se incluyen imports, rutas y funciones auxiliares

****

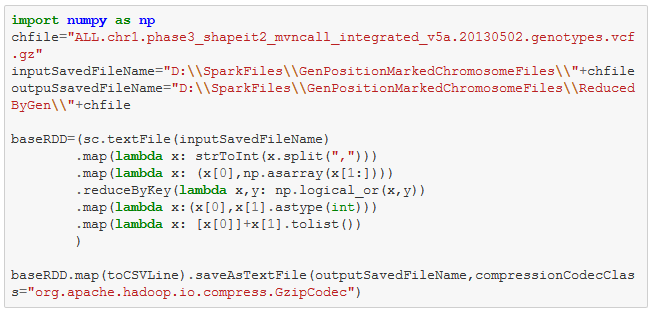
****

****

En la siguiente casilla se cuenta con la carga de archivos VCF, su parseo, filtrado y mapeado, guardando el resultado



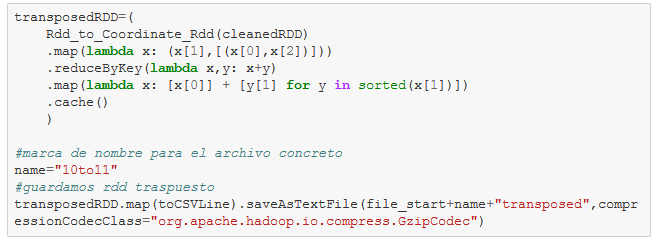
La siguiente casilla carga los datos generados anteriormente, realizando fase de reducción y guardando el resultado

****

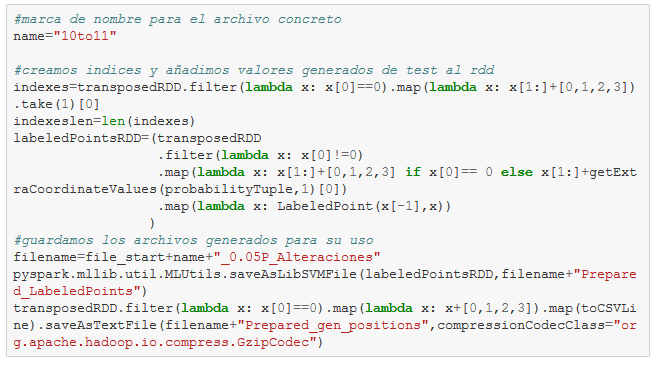
A continuación, se cargan los archivos deseados, preparados anteriormente, para combinarlos en un solo RDD

****

A continuación se realizará la transposición del RDD

****

Finalmente se ejecutan las últimas conversiones de datos y almacenamos los resultados

****

#### **Extra: Implementación del coeficiente de correlación de Pearson sobre Rdds**

La siguiente celda muestra la implementación del coeficiente de Pearson sobre RDDs, así como una función para chequear los valores seleccionados.



## **9. BIBLIOGRAFÍA**

[1] Bert Hubert(2016, Enero 20), **“**DNA seen through the eyes of a coder” [online]. Disponible: https://ds9a.nl/amazing-dna/

[2] Carlos Eiras-Francoa, Verónica Bolón-Canedoa, Sabela Ramosb, et al, “Multithreaded and Spark parallelization of feature selection filters”, Journal of Computational Science, VOL. 17, Part 3, 2016, pp. 609–619

[3] [Christopher D. Manning](http://nlp.stanford.edu/%7Emanning/), [Prabhakar Raghavan](http://theory.stanford.edu/%7Epragh/) y [Hinrich Schütze](http://www.cis.uni-muenchen.de/personen/professoren/schuetze/), *Introduction to Information Retrieval*, Cambridge University Press. 2008.

[4] David Dietrich, Barry Heller, Beibei Yang, “Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data”, EMC Education Services, 2015

[5] IGSR (2017, Febrero 6), 1000 Genomes Project data [online], Disponible: http://www.internationalgenome.org/

[6] Isabelle Guyon, Andre Elisseeff, “An Introduction to Variable and Feature Selection”, Journal of Machine Learning Research 3, 2003, pp. 1157-1182

[7] NumPy developers, (2016),“NumPy”[online],Disponible: http://www.numpy.org

[8] Sergio Ramírez (2015, Junio 1), libreria fast-mRMR [online], Disponible: github.com/sramirez/fast-mRMR

[8] Sergio Ramírez (2016, Septiembre 22 ), librería infotheoretic [online], Disponible: https://github.com/sramirez/spark-infotheoretic-feature-selection

[10] S. Ramírez-Gallego,I. Lastra,D. Martínez-Rego, et al ,“Fast-mRMR: Fast Minimum Redundancy Maximum Relevance Algorithm for High-Dimensional Big Data”, INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 00, 2016, pp. 1–19

[11] The Apache Software Foundation. (2017, Enero 26), “Apache Hadoop”[online], Disponible: http://hadoop.apache.org/

[12] The Apache Software Foundation. (2017, Enero 4),”Apache Spark”[online], Disponible: http://spark.apache.org/

[13] The Apache Software Foundation. (2013, Abril 8),”MapReduce Tutorial”[online], Disponible: hadoop.apache.org/docs/r1.2.1/mapred\_tutorial.html

[14] The Apache Software Foundation. (2017, Enero 26), “PySpark 2.0 Documentation” [online], Disponible: http://spark.apache.org/docs/2.0.0/api/python/pyspark.mllib.html

[15] UCSC, genome.ucsc.edu (2013, Diciembre), Datos de posiciones de genes (UCSCgenPfam.txt.gz)[online], Disponible: http://hgdownload.cse.ucsc.edu/goldenpath/hg38/database/

[16] US National Cancer Institute, (2015, Abril 1),“BRCA1 and BRCA2: Cancer Risk and Genetic Testing”[online], Disponible: www.cancer.gov/about-cancer/causes-prevention/genetics/brca-fact-sheet

[17] US National Human Genome Research Institute(2003,Abril 14),” The Human Genome Project Completion: Frequently Asked Questions”[online], Disponible: www.genome.gov/11006943/human-genome-project-completion-frequently-asked-questions/

[18] Wikipedia contributors, (accedido 2017, Febrero 11), "Bigtable," *Wikipedia, The Free Encyclopedia,* [online] Disponible: https://en.wikipedia.org/w/index.php?title=Bigtable&oldid=762217700

[19] Wikipedia contributors, (accedido 2017, Febrero 11), "BRCA1," *Wikipedia, The Free Encyclopedia,* [online] Disponible: https://en.wikipedia.org/w/index.php?title=BRCA1&oldid=764531206

[20] Wikipedia contributors, (accedido 2017, Febrero 11), "Chi-squared test," *Wikipedia, The Free Encyclopedia,* [online] Disponible: https://en.wikipedia.org/w/index.php?title=Chi-squared\_test&oldid=753998755

[21] Wikipedia contributors, (accedido 2017, Febrero 11), "Computational complexity theory," *Wikipedia, The Free Encyclopedia,* [online] Disponible: https://en.wikipedia.org/w/index.php?title=Computational\_complexity\_theory&oldid=761785410

[22] Wikipedia contributors, (accedido 2017, Febrero 11),, "Distributed computing," *Wikipedia, The Free Encyclopedia,* [online] Disponible: https://en.wikipedia.org/w/index.php?title=Distributed\_computing&oldid=762887036

[23] Wikipedia contributors, (accedido 2017, Febrero 11), "Pearson correlation coefficient," *Wikipedia, The Free Encyclopedia,* [online] Disponible: https://en.wikipedia.org/w/index.php?title=Pearson\_correlation\_coefficient&oldid=764630851