Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Maestría en Explotación de Datos y Descubrimiento de Conocimiento

Predicción de secuencias de inserción en genomas bacterianos utilizando algoritmos de machine learning.

Tesis presentada para optar al título de Magíster de la Universidad de Buenos Aires en el área de Explotación de Datos y Descubrimiento de Conocimiento

Autor: Lic. Miguel Ángel Barros

Director: Dr. German M. Traglia

Co-Director: Dr. Andrés Iriarte

Lugar de trabajo: Departamento de Desarrollo Biotecnológico, Instituto de Higiene, Facultad de Medicina, Universidad de La República, Montevideo, Uruguay

Fecha de defensa: 25 de Julio 2023.

Título de Tesis: Predicción de secuencias de inserción en genomas bacterianos utilizando algoritmos de Machine Learning.

Lugar de trabajo: Departamento de Desarrollo Biotecnológico, Instituto de Higiene, Facultad de Medicina, Universidad de La República. Montevideo, Uruguay

Buenos Aires, 2023.

Director: Dr. German M. Traglia

_____

Co-Director: Dr. Andrés Iriarte

Agradecimientos

A mi esposa, Andrea.

A mis hijos, Felipe y Salvador, mis pequeños héroes.

A mis padres y hermana.

A mi amigo y director, el Dr German Traglia.

Al Dr. Andrés Iriarte por el espacio y sus devoluciones.

Resumen

Las secuencias de inserción (IS) son elementos genéticos móviles que tienen la capacidad de desplazarse desde una determinada región del genoma hacia otra. Las IS son una fuente de variabilidad genetica que podrían brindar rasgos adaptativos a distintas especies bacterianas, por ejemplo: resistencia antibiotica. Sin embargo, la identificación de las IS no es una tarea simple dado los rasgos genéticos variables entre los distintos tipos existentes.

El objetivo del presente trabajo fue desarrollar un *software* basado en algoritmos de aprendizaje automatizado que permita identificar IS sobre diferentes especies de genomas bacterianos. Para lo cual, para entrenar los diferentes algoritmos de clasificacion, se trabajó con un dataset inicial compuesto por 8.223 secuencias aminoacídicas de IS y 8.223 secuencias aminoacídicas pertenecientes a otro tipo de estructura protéica (las cuales se denominaron non-IS). Las primeras se obtuvieron de bases de datos específicas de IS, como ISFinder. En tanto que el resto de las secuencias fueron descargadas de la base PDB, *Protein Data Bank.*

Los clasificadores evaluados fueron seis: Regresión Logística, Support Vector Machines (SVM), Stochastic Gradient Descent (SGD), Xtreme Gradient Boosting (XGBoost), Random Forest y Light Gradient Boosting Machine (LGBM). Para validar el rendimiento del modelo, se incluyó una etapa adicional, validación, en la cual se ejecutaron a los algoritmos sobre datos que a los que dichos clasificadores no habian sido expuestos con anterioridad. Estos datos de validación correspondieron a cinco genomas bacterianos de referencia: *Escherichia coli* K-12, *Salmonella enterica* serovar Typhi CT18, *Acinetobacter baumannii* AYE, *Staphylococcus aureus* Newman y *Pseudomonas aeruginosa*.

El clasificador que mostró el mejor rendimiento fue XGBoost, el cual obtuvo valores de 93.9% en Sensitividad, 94.1% en Especificidad y 94% en Accuracy en la etapa de testing, demandando 15 segundos de tiempo de cómputo en un ordenador portátil. El posterior análisis, mediante BLAST, sobre los falsos positivos producidos durante la clasificacion, demostraron que el modelo desarrollado fue capaz de identificar nuevas IS con un elevado nivel de precisión.

Abstract: Insertion Sequence prediction on bacterial genomes via Machine Learning algorithms

Insertion sequences (IS) are genetic elements capable to move itself from a certain DNA region to another one. ISs are considered as a source of genetic variability that provides adaptative features to different bacterial species, such us antibiotic resentence, among others. Nevertheless, IS identification is not a simple process due to the high genetic variability among different types of these genetic elements

The main goal of this study was to develop a software based on machine learning allowing the identification of IS on different species of bacterial genomes.

To accomplish that task, an initial dataset composed of 8,223 amino acid sequences belonging to IS (retrieved from IS-Finder repository) and 8,223 amino acid sequences from another type of protein structure (which were called non-IS) was utilized to train the different classifiers.

Six classifiers were evaluated: Logistic Regression, Support Vector Machines (SVM), Stochastic Gradient Descent (SGD), Xtreme Gradient Boosting (XGBoost), Random Forest, and Light Gradient Boosting Machine (LGBM). To validate the performance of the model, an additional stage was included referred as validation. Along this phase, the pull of trained classified was executed on new datasets where those algorithms had not been previously exposed. These new datasets consisted five in bacterial aminoacidic sequences from reference organisms. Validation datasets come from *Escherichia coli* K-12, *Salmonella enterica* serovar Typhi CT18, *Acinetobacter baumannii* AYE, Staphylococcus aureus Newman y *Pseudomonas aeruginosa*.

The classifier that showed the best performance was XGBoost, which obtained values of 93.9% in Sensitivity, 94.1% in Specificity and 94% in Accuracy in the testing stage, demanding 15 seconds of computing time on a laptop.

The subsequent analysis, using BLAST, on the false positives produced during the classification, demonstrated that the developed model was capable to detect new IS with a high level of precision.

Indice general

Introducción

Estructura del genoma bacteriano

La información genética esencial para la vida de la bacteria está contenida en una molécula de ácido desoxiribonucléico (ADN) de doble cadena y circular, cerrado por enlaces covalentes. Dicha molécula se denomina cromosoma bacteriano. Muchas bacterias poseen, además, ADN extracromosómico, el cual es también circular y cerrado, denominado ADN plasmídico por estar contenido en los plásmidos. Así, las bacterias poseen información genética para muchas funciones que no son esenciales para la célula en condiciones normales de crecimiento.

La arquitectura del genoma, que consta de un cromosoma, más uno o múltiples replicones grandes adicionales, se denomina genoma dividido o genoma multipartito. Curiosamente, los estudios han observado repetidamente para los genomas multipartitos que no solo cada molécula de ADN está físicamente separada, sino que cada molécula también tiene propiedades distintas, como diferencias en el uso de codones (la proporción de codones sinónimos que se usan), contenido de GC (porcentaje del ADN compuesto por guanina y citosina) y abundancia relativa de dinucleótidos (la frecuencia con la que aparece cada par de nucleótidos en la secuencia de ADN). Por otro lado, la arquitectura del cromosoma bacteriano se puede dividir en, genoma central y genoma accesorio. El genoma central contiene: los genes conservados, la especie bacteriana (presencia en al menos 99% de los genomas de la especie) y el genoma accesorio, que son los genes variables entre los diferentes aislados bacterianos de una especie determinada. La conservación de los genes del genoma central no necesariamente los convierte en esenciales para la viabilidad bacteriana.

La organización de los genomas procarióticos no es estocástica, sino que su organización refleja algún propósito funcional o regulador (Junier 2014, Rocha 2008, Lawrence 2003). Por ejemplo, las enzimas para cada paso de una ruta biosintética o catabólica generalmente están codificadas

por un solo operón y, a menudo, se colocalizan en el cromosoma con su regulador (Rocha 2008). La ubicación cromosómica de un gen puede influir en su nivel de expresión (Bryant y cols 2015) y, al menos en especies de replicación rápida, en el número de copias del gen (Dryselius y cols 2008). Además, existe una tendencia general a que los genes bacterianos se enriquezcan en la cadena principal para evitar colisiones frontales entre las maquinarias transcripcional y replicativa del ADN (Rocha 2004). Dada la naturaleza estructurada de los genomas procarióticos, es poco probable que la estructura del genoma multipartito simplemente represente una peculiaridad evolutiva y, en cambio, presumiblemente esté moldeada por presiones selectivas. Las distintas partes del genoma multipartito bacteriano se pueden clasificar en: replicones, el cual se define como cualquier molécula de ADN que posee una naturaleza especifica, cromosoma (cromosoma primario), plásmidos y megaplásmidos, crómidos y cromosomas secundarios (diCenzo y cols 2017).

Brevemente, el cromosoma bacteriano se refiere al replicón principal o primario. El cromosoma es siempre el replicón de mayor tamaño que contiene la mayoría de los genes esenciales. Los plásmidos y megaplásmidos son considerados replicones secundarios que contiene genes no centrales y no esenciales, y son dispensables para la viabilidad celular en la mayoría de los ambientes. La adquisición de plásmidos se lleva a cabo por medio de mecanismos de transferencia horizontal (Transformación, Transducción y Conjugación) (diCenzo y cols, 2017). Su firma genética difiere significativamente del cromosoma en el contenido de GC y composición de dinucleótidos. Por otro lado, los crómidos son considerados replicones que son intermediarios entre plásmidos y cromosomas. El sistema de replicación es similar al de los plásmidos o megaplásmidos, pero puede tener controles adicionales que integran su replicación dentro del ciclo celular. A diferencia de los plásmidos (sin genes centrales), los crómidos contienen al menos un gen esencial o central para la viabilidad celular y generalmente tiene una firma genómica con mayor similitud a la observada en el cromosoma bacteriano. Por ultimo los cromosomas secundarios son replicones secundarios que presentan una gran similitud con los crómidos. La diferencia entre ellos es que la aparición del crómidos es un evento reciente, mientras que los cromosomas secundarios son replicones secundarios que posiblemente se encuentren presentes desde la diferenciación de la especie bacteriana o especiación (Harrison y cols, 2010).

Elementos genéticos móviles como fuente de variabilidad genética en la evolución de genomas bacterianos

El gran potencial de las bacterias para el intercambio de la información genética a través de la transferencia horizontal de genes (THG), ha sido reconocido como un factor importante en su adaptación genética y evolución (Brigulla y cols., 2010). Hacia el año 1970 se determinó que la adquisición de diversos genes, que suministran a la bacteria mecanismos para una mejor adaptación a distintos ambientes, está mediada por diferentes elementos móviles, como ser plásmidos, transposones, secuencias de inserción, integrones y cassettes génicos, capaces de transferir su material de un organismo a otro (Cambray y cols., 2010).

La THG, es la responsable de la movilización de elementos genéticos y del ADN bacteriano entre bacterias, y posee un profundo impacto en la evolución bacteriana. La eficiencia con la cual las bacterias incorporan información genética, refleja su capacidad para adaptarse a los cambios ambientales. Varios mecanismos están involucrados en el intercambio de material genético, pero dicho material genético transferido, puede o no, ser incorporado al nuevo genoma (Mazel y cols., 1999). Hoy en día, es bien reconocida la importancia de los mecanismos de THG, por ser los principales responsables de la amplia diseminación de genes de resistencia antibiótica en bacterias patógenas y ambientales (Chen y cols., 2005, Dantas G. y cols. 2008, Sommer y cols., 2010). Mientras que los plásmidos participan en la diseminación de la resistencia antibiótica a nivel celular, los transposones/secuencias de inserción promueven el intercambio a nivel molecular, y es así como se comenzó a asociar a la multirresistencia antibiótica con la THG (Bonomo y cols., 2007).

La THG comprende diversos mecanismos que permiten la adaptación de “nuevos” genes dando lugar a la formación de genomas heterogéneos y dinámicos (diCenzo y cols, 2017). Para que el “nuevo” gen pueda ser adquirido es necesario que se inserte en el genoma bacteriano y se exprese. Hasta el momento, se conocen cinco eventos involucrados en la THG: endosimbiósis, fusión celular, transformación, conjugación y transducción. Estos tres últimos fenómenos son característicos de las bacterias. En la transformación, algunas bacterias (denominadas competentes) son capaces de incorporar ADN exógeno proveniente de otras bacterias. La propiedad de captar ADN del exterior, conservarlo e interaccionar, se denomina competencia. El mecanismo de transducción es la transferencia de ADN de una bacteria a otra por medio de un

bacteriófago. Por último, el mecanismo de conjugación se basa en el intercambio unidireccional de información genética desde una bacteria donante hacia otra receptora mediante un contacto entre ambas. Es importante notar que muchas bacterias poseen información genética contenida en moléculas distintas a las del cromosoma bacteriano, dichos elementos genéticos se denominan plásmidos. En la conjugación, son los plásmidos los que se transmiten de este modo con mayor frecuencia. La capacidad de conjugación depende de la presencia de plásmidos conjugativos en la bacteria, los cuales contienen los genes necesarios para llevar a cabo tal mecanismo. Cabe notar, además, que generalmente estos tipos de plásmidos poseen la maquinaria molecular para la transferencia de su propio material genético. No obstante, el plásmido es capaz de integrarse al cromosoma bacteriano y, al momento de conjugar, se transferirá no solo a sí mismo, sino también a los genes cromosómicos que se encuentren tras él (Betancour y cols., 2017).

Entre los elementos genéticos móviles que participan en forma directa de la THG y poseen una gran importancia en la diseminación de la resistencia antibiótica, podemos nombrar a los plásmidos, las secuencias de inserción y los cassettes génicos.

Secuencias de inserción, elemento genético móvil de amplia distribución en genomas bacterianos

Los elementos transponibles o genes saltarines son secuencias de ADN que tienen la capacidad intrínseca de moverse desde una determinada región del genoma hacia otra en la que, posteriormente, se integrará. Barbara McClintock identificó los transposones (TE, del inglés, transposable elements) o genes saltarines (McClintock B, 1956) e introdujo el concepto en su estudio de análisis de inestabilidad genética en la pigmentación del maíz (McClintock B, 1956).

Los TE contribuyen de manera significativa, tanto en el tamaño, como en la diversidad del genoma. En mamíferos, se ha demostrado mediante estudios de secuenciación que, entre un tercio y la mitad del genoma deriva de TEs. Esta abundancia hace que tengan un impacto considerable en la evolución de los genomas de mamíferos (Platt y cols., 2018). Los TE juegan, además, un rol clave en: la estructuración de los cromosomas, regulación de expresión génica, adaptación, evolución (Orozco-Arias y cols., 2019) así como en la composición del centrómero

(segmento central del cromosoma) en plantas (De Castro Nunes y cols., 2018). Además, los TE han sido encontrados en todos los géneros bacterianos en los cuales han sido buscados. Estos elementos juegan un papel central en la evolución, proporcionando mecanismos para la generación de variabilidad, y, en conjunción con los sistemas de transferencia de ADN, participan de la rápida diseminación de material genético entre bacterias (Hall y cols., 1989, Syvanen 1984)

Desde el punto de vista estructural, los TE poseen en sus extremos pequeñas secuencias repetitivas e invertidas denominadas IR, las cuales, en la mayoría de los casos, son idénticas o casi idénticas entre sí. Estas regiones poseen relevancia, dado a que son reconocidas por la proteína (o complejo proteico) denominada transposasa para llevar a cabo el desplazamiento (Rao y cols., 2000, Streips y Yasbin, 2002, Craig 1996). Es importante notar que los TE codifican ésta proteína, la cual promueve y facilita la propia transposición de éstos elementos.

Diversas clasificaciones han sido utilizadas para hacer referencia a los distintos transposones, algunos los clasifican en función de su estructura, otros en base a la función, etc. Si nos referimos a la clasificación teniendo en cuenta su estructura, se clasifican como secuencias de inserción (IS, del inglés, Insertion Sequence) a los elementos transponibles más sencillos y pequeños (0,5-2 Kb). Las IS son elementos genéticos móviles que pueden moverse de un segmento de ADN a otro a través de reacciones de escisión e integración independiente de la recombinación homóloga. Pueden movilizarse desde el cromosoma a un plásmido o viceversa, dentro de la misma molécula de ADN. Son unidades autónomas que solo portan el gen que codifica para la transposasa posibilitando solo la movilización del mismo. A nivel estructural, una IS posee en sus extremos las repeticiones terminales invertidas características de los transposones (Figura 1). Es importante notar que las IS fueron los primeros elementos transponibles reconocidos en las bacterias. Como ejemplos puede citarse a los siguientes tipos de secuencias de inserción: IS1, IS2 y IS3 (Mahillon y Chandler 1998, Mahillon y cols., 1999).

Las secuencias de inserción han sido agrupadas en 29 familias de acuerdo a los siguientes criterios: 1) similitud en la organización genética (disposición de los marcos de lectura abierta), 2) identidades marcadas o similitudes en sus Transposasas (dominios comunes o motivos), 3) características similares de sus extremos (terminal de IR), y 4) el destino de la secuencia de nucleótidos de sus sitios blanco (la generación de una duplicación objetivo directo de longitud determinada) (Mahillon y cols.,1998). La importancia de la clasificación en familias de secuencias de inserción radica en la posibilidad de determinar el origen de las secuencias de inserción como así también el blanco donde se insertaría en el genoma bacteriano.

Los elementos IS son los que codifican para la actividad transposasa que es la responsable tanto de crear un sitio blanco como de reconocer los extremos del transposón.

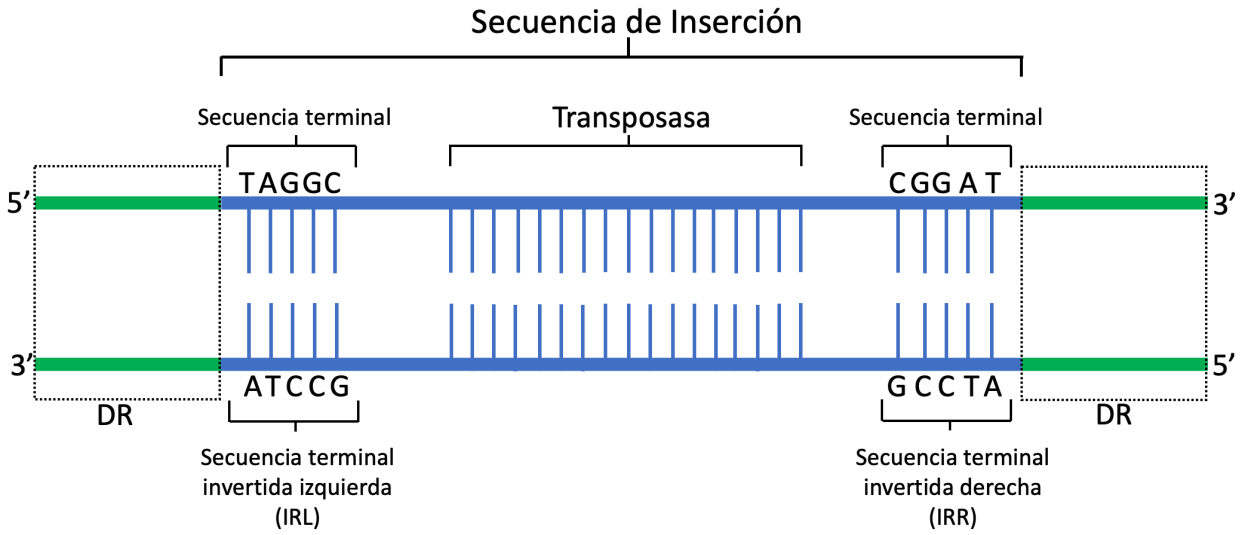

Figura 1 Estructura molecular de una secuencia de inserción.

Bases de datos genómicas

Una base de datos de genomas puede ser definida como un repositorio de secuencias de ADN que corresponden a diferentes tipos de especies de organismos. Dichas bases de datos están diseñadas y mantenidas dentro servidores usando diversos sistemas operativos, como así también diferentes protocolos de transferencia e interfaces de usuario. Las secuencias almacenadas son generadas, reportadas y actualizadas por investigadores del campo de la genética, biología, biotecnología y disciplinas afines mediante la utilización de tecnologías que posibilitan la determinación del orden de nucleótidos de una secuencia de ADN (Carroll y cols., 2000). Un gran número de tecnologías de secuenciación y mapeo han sido desarrolladas en principio aplicadas a genomas de organismos modelo como *Escherichia coli*, *Saccharomyces cerevisiae*, *Drosophila melanogaster, Caenorhabditis elegans* o *Mus musculus* pero actualmente extendidas a un conjunto de organismos vivos muy vasto. Cabe destacar que los proyectos de secuenciación sobre organismos procariotas han sido más diversos que en organismos

eucariotas (Hide, 2009). A grandes rasgos, las bases de datos ómicos pueden ser descritas en función del criterio del tipo de molécula: ADN genómico (genómica), ARNm o proteínas (proteómica). A su vez, se existen bases de datos que recolectan los datos ómicos para grupos taxonómicos particulares que presentan relevancia en el campo de la salud, industria, etc. Entre ellas se pueden destacar la base de datos Burkholderia.com (https://www.burkholderia.com/) o WormBase (https://wormbase.org), entre muchas otras y a modo de ejemplo. Además de la secuencia, las bases de datos tienen información sobre los roles biológicos, las funciones moleculares, la estructura de las proteínas, los dominios funcionales, entre otros aspectos que hacen a la anotación de las secuencias.

La anotación del ADN de un genoma consiste en la identificación estructural de genes, regiones codificantes y motivos, así como la identificación funcional de esas regiones; y tiene como objetivo final la identificación y etiquetado de todas las características relevantes en una secuencia del genoma. La relación de diferentes secuencias genómicas puede abordarse mediante estudios comparativos, y en este sentido, las bases de datos genómicas juegan un rol fundamental.

Estas bases varían en cuanto a tamaño de almacenamiento, tipo de conectividad y complejidad. El centro nacional de información biotecnológica o NCBI (del inglés National Center of Biotechnology Information) y “Ensembl genome” el cual fue fundado por Wellcome Trust, son los repositorios más grandes que existen actualmente. Ambos están centralizados en el almacenamiento de secuencias y tienen la capacidad para alojar genomas de diversas especies. NCBI es una de las bases de datos más antiguas y sirve como referencia. Esta base integra la información existente de varias otras bases de datos y de diferentes tipos.

Machine learning como potencial herramienta para el análisis genético en la era de las "omicas"

Dado al gran uso en el campo de la medicina, y en todas las ciencias biológicas en general, el análisis de secuencias proteicas ha sido ampliamente estudiado en la última década (Barve y cols., 2013; Chen y cols., 2010; Cong y cols., 2013; Machado y cols., 2013; Carregari y col., 2013; Liu y cols., 2013). Dicho tipo de estudio generalmente ayuda a caracterizar las secuencias proteicas *in silico* y permite la predicción tanto a nivel estructural como funcional de una proteína. Esto ha promovido el desarrollo de bases de datos y repositorios de datos de proteínas como "Protein Information Resource" (PIR, https://proteininformationresource.org/), Protein Data Bank (PDB, https://www.rcsb.org/), UniProt (Universal Protein Resource, https://www.uniprot.org/), entre muchos otros. De este modo, lograr una explotación eficiente de la información disponible en dichas bases de datos se ha convertido en un desafío de gran relevancia. Por otro lado, Baldi y Brunak (2001) propusieron que la anotación de secuencias tiene un rol crucial en el análisis. Por ejemplo, los miembros cuya secuencia aminoacídica que se agrupan en una misma superfamilia de proteínas, están relacionadas evolutivamente y son, funcional y estructuralmente, afines entre sí. Generalmente, dos secuencias aminoacídicas son clasificadas dentro de la misma categoría, como homólogas, si luego de ser sometidas a un análisis de secuencia, éstas presentan una identidad mayor a la esperada por azar. En la práctica, para definir homología se han definido valores de corte de identidad mayores a 40% y una zona de incertidumbre entre 30% y 40% (Rost 1999). La lógica subyacente de un algoritmo de búsqueda de similitud de secuencias radica en comparar, en este caso, la secuencia de una proteína desconocida (o target) contra otra ya conocida y luego cuantificar la similaridad entre ambas secuencias. Han sido reportados diversos algoritmos de clasificación funcional a lo largo de la última década. Entre los cuales se pueden mencionar solo a modo de ejemplo, iPro-Clas (proteininformationresource.org/), SAM (por sus siglas en inglés Sequence Alignment and Modeling Software System, compbio.soe.ucsc.edu/sam.html), y MEME (por sus siglas en inglés Multiple Expectation Maximization for Motif Elicitation, www.cbil.upenn.edu/EpoDB/release/version_2.2/meme/intro.html). Sin embargo, es importante notar que los algoritmos enlistados son costosos en términos de tiempo de cómputo requerido al momento de realizar las comparaciones de proteínas target contra otras existentes, sobre todo cuando la base de datos es extensa y la longitud de la proteína objetivo es elevada. Una alternativa viable suele ser el algoritmo BLAST (por sus siglas en inglés Basic Local Alignment

Search Tool) el cual directamente aproxima alineamientos que optimizan la medida de similaridad local (Altschul y cols., 1990). De este modo, en base a la similitud a nivel de la secuencia, éste método permite asignar a una proteína incógnita (denominada *query*), un determinado nombre o clasificarla dentro de una familia o clase de proteínas previamente determinada.

Con el objetivo de realizar una clasificación funcional sobre secuencias proteicas se han aplicado diversos algoritmos del campo de aprendizaje automatizado (también conocido como *machine learning*) o estadística bayesiana. Entre dichos algoritmos, se pueden mencionar Support Vector Machines (SVM), árboles de decisión y redes neuronales. En 2006, Borro y colaboradores reportaron el uso de clasificadores bayesianos para la predicción de tipos de enzimas a partir de secuencias (Borro y cols., 2016). Por otro lado, Lee y colaboradores han reportado métodos de ensamble basados en árboles de decisión como Random Forest (Lee y cols., 2009). Amidi y colaboradores en el año 2016, implementaron una combinación de dos clasificadores SVM y Nearest Neighbor (Amidi y cols., 2016) con el fin de clasificar secuencias.

En cuanto al procesamiento de las secuencias aminoacídicas, para ser tratadas como variables independientes, Yang y colaboradores (Yang y cols., 2008) han utilizado métodos basados en segmentación de palabras con el fin de extraer atributos que posteriormente son usadas como valores de entrada para el algoritmo SVM, que luego llevará a cabo la clasificación. Por otro lado, Caragea y colaboradores han reportado la implementación de funciones de hashing con el objetivo de reducir las dimensiones del vector que representa las secuencias aminoacídicas y luego también aplicar el clasificador SVM (Caragea y cols., 2012).

Los clasificadores de tipo no lineales generalmente tienen un mejor desempeño ante un problema de clasificación funcional de proteínas (Gupta y cols., 2019). Dentro de este grupo se incluye a los métodos basados en árboles de decisión, SVM y las redes neuronales o NN (del inglés Neural Networks). La explicación radica en la naturaleza de la representación de los datos. Las características o *features* (atributos) que se extraen de secuencias aminoacídicas están distribuidas en un espacio altamente dimensional y es de tipo no lineal. Para este tipo de datos con características complejas, no es una tarea trivial hallar un modelo de clasificación satisfactorio.

Representación de secuencias aminoacídicas y extracción de atributos

Wu y colaboradores han reportado el uso de metodologías basadas en “Procesamiento del Lenguaje Natural” (NLP, del inglés Natural Language Processing) con el fin de obtener una representación matemática de las secuencias de aminoácidos correspondientes a cada proteína (Wu y cols., 1995). El abordaje consistió en el uso de funciones específicas que permiten obtener n-gramas, y luego, la implementación de métodos algebraicos con el objetivo de reducir la dimensionalidad en la matriz original compuesta de n-gramas.

Un n-grama está formado por secuencias de entidades lingüísticas según el orden en el que aparecen en un texto. En este caso en particular, un n-grama está dado por los grupos de aminoácidos que están dispuestos en una cadena aminoacídica. En la actualidad hay disponibles aplicaciones que permiten ser implementadas sobre secuencias de textos y que generan matrices de n-gramas como resultado. Un ejemplo, es la función CountVectorizer disponible en el framework (conjunto de librerías relacionadas con *machine learning*) Scikit-learn en el lenguaje de programación Python (Pedregosa y cols., 2011).

El método Descomposición en Vectores Singulares (inglés SVD, “Singular Value Decomposition”) ha sido uno de los grandes desarrollos aportados por el álgebra lineal moderna, jugando un rol clave en un rango importante en aplicaciones de modelado y simulación. Algunos ejemplos son el modelado de la expresión génica (Alter y cols., 2000), análisis de interacción océano-atmosfera (Wallace y cols., 1992), *data-unfolding* (desdoblamiento) de alta energía en física (Hoecker y cols., 1996), sistemas de recomendación (Sarwar y cols., 2002), y simulación con datos de MRI (Magnetic Resonances Imaging) (Calamante y cols., 2000). El método SVD ha sido adoptado, además, en minería de texto en tareas como Indexación Semántica Latente (Latent Semantic Indexing) (Deerwester y cols., 1990) dentro del campo de la Recuperación de la información (IR, del inglés, Information Retrieval). Este último abordaje, se basa en el orden implícito de la estructura generada por los términos presentes en los respectivos documentos, con el fin de mejorar la detección de documentos relevantes que pueden o no, contener los términos que se deseen buscar.

En SVD, una matriz de datos de entrada original es descompuesta en un set de K factores ortogonales a partir de los cuales, la matriz original puede ser aproximada por combinación lineal.

Dicho de otro modo, una matriz A, de tipo m x n, puede ser representada como el producto de tres matrices diferentes (Figura 2), las cuales tienen diferentes restricciones. En primer lugar, la matriz U, ortogonal, de tamaño m x k, que contiene los vectores singulares(*s*) izquierdos (left singular vectors), los cuales representan a las columnas de la matriz de entrada (n) o k valores más largos de s. La matriz sigma, de tipo k x k, es la matriz que contiene los valores singulares, es diagonal, no negativa y alberga todos valores nulos excepto en su diagonal. Dichos elementos singulares están ordenados de manera decreciente, el valor singular más elevado se localiza en primer lugar en el ángulo superior izquierdo (cuadrante II del plano cartesiano). El número de valores singulares o s-values están determinados por k. Por último, la matriz V transpuesta, es ortogonal, y sus columnas contienen los valores singulares derechos (right singular vectors) y está representada como n x k (Ecuación 1, Figura 2).

$$A\ (m \times n) = U\ (m \times k) * \Sigma\ (k \times k) * V^{t}\ (n \times k)$$

Ecuación 1. Función matemática de SVD

El parámetro a tener en cuenta es el valor de K, el cual representa la reducción de dimensiones, o número de dimensiones finales que se busca obtener. Dicho valor es obtenido de modo heurístico.

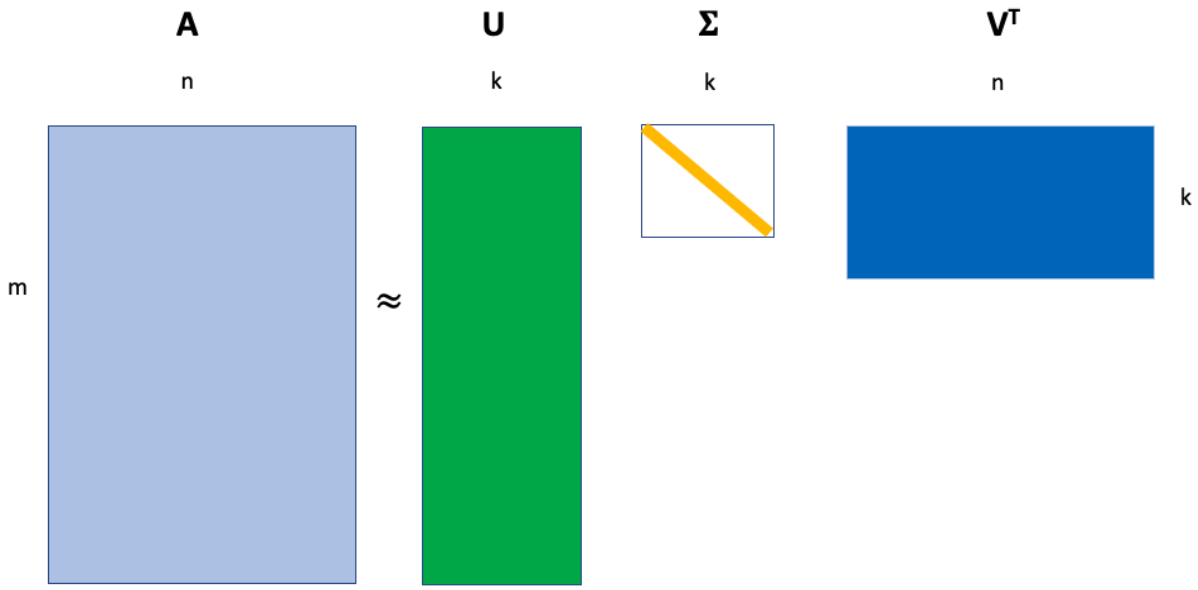

Figura 2. Representación esquemática de la técnica SVD

En el campo de la genómica, Wu y colaboradores implementaron la metodología SVD con el fin de generar un método de codificación de secuencias y de lograr comprimir una larga y rala matriz de entrada que consistió en vectores generados por los n-gramas (Wu y cols., 1995). El resultado de la metodología SVD fue una matriz factorizada y comprimida que a su vez hizo posible reducir la complejidad del algoritmo clasificador basado en redes neuronales.

Por otro lado, Stuart y colaboradores reportaron el uso de métodos de alineamiento de genomas basados en SVD con el fin de comparar secuencias aminoacídicas de nueve genomas eucarióticos completos (Stuart y cols., 2004). Dicha implementación hizo posible la identificación y definición de un gran número de motivos y familias de genes. Por último, Gomide y colaboradores implementaron el algoritmo SVD para llevar a cabo análisis estructurales en proteínas a partir de datos de interacciones hidrofóbicas (Gomide y cols., 2009).

Estado del arte de la clasificación de IS

Habitualmente la clasificación de elementos transponibles (TE), se realiza mediante herramientas bioinformáticas basadas en criterios de homología (Vendrell-Mir, 2019; Ewing 2015) Dicho abordaje consiste en comparar la nueva secuencia (secuencia objetivo) contra otras secuencias las cuales han sido ya previamente identificadas como TE. Si bien este método ha sido ampliamente aceptado, posee algunas desventajas dentro de las cuales se puede mencionar que la homología entre secuencias omite ciertas propiedades bioquímicas, las cuales son útiles al momento de la clasificación Vendrell-Mir, 2019; Ewing 2015).

Estudios recientes se han centrado en la clasificación jerárquica de los TE (Loureiro, 2013, Santos 2016, Schietgat, 2018). Por ejemplo, Schietgat y colaboradores, desarrollaron un marco de trabajo (framework) cuyo objetivo fue clasificar los TE a nivel estructural. En dicho trabajo, los autores además de diseñar el framework basado en algoritmos de aprendizaje automatizado (como el algoritmo de clasificación *Random Forest*), incorporaron un estudio comparativo con otras dos metodologías de identificación de TE, Repeated Masker (Smit y cols., 2010) y Censor (Jurka y cols., 1996). Estos dos últimos abordajes con los que se hizo el contraste son métodos basados en búsqueda de similitud de secuencias y comparaciones de a pares (*pairwise comparison*) entre una secuencia incógnita (query) y una secuencia proveniente de una base de datos de referencia mediante algoritmos como BLAST. De este modo Schietgat y colaboradores, reportaron una mejora significativa en términos de la capacidad de detección de TE mediante el uso de métricas de rendimiento de la clasificación (que serán posteriormente definidas) como Sensibilidad o *Recall* como es denominado en su publicación.

Nakano y colaboradores (Nakano y cols., 2018) han reportado la implementación de redes neuronales en el problema de clasificación jerárquica sobre TE. En dicha publicación, los autores trabajaron sobre dos conjuntos de datos de distinto origen que constaron de secuencias de ADN repetitivo. Estos datos fueron organizados jerárquicamente en base al criterio propuesto por Wicker y colaboradores (Wicker y cols., 2007). En términos de los algoritmos implementados, de modo similar al trabajo publicado por Schietgat y colaboradores (Schietgat y cols, 2018), Nakano y colaboradores además de incluir su abordaje de clasificación mediante redes neuronales incluyeron el algoritmo basado en búsqueda de similitud mediante BLASTn. Sobre ambos conjuntos de datos, los autores reportaron valores superiores, a los publicados en el trabajo de Schietgat, en términos de Sensibilidad y Precisión.

En relación a las secuencias de inserción (IS), que, como se explicó en párrafos anteriores, son un tipo de elemento transponible (TE), no hay, hasta el momento, bibliografía disponible que haga referencia a su clasificación mediante el uso de métodos de aprendizaje automatizado. Huda y colaboradores (Huda y cols., 2014), reportaron el uso de un flujo de trabajo (o *pipeline*) mediante el cual es posible la identificación de IS. No obstante, en dicha publicación, los autores reportan la integración de bases de datos públicas como ISFinder *(*Siguier y cols., 2016*)* y softwares de anotación de genes bacterianos como Prokka (Seemann., 2014), BASys (Van Domselaar y cols., 2005) y Prodigal (Hyatt y cols., 2009). De este modo, en su publicación los autores llevaron a cabo comparaciones entre las diferentes herramientas de detección de IS. Cabe notar, que dicho trabajo solo se focaliza en una especie bacteriana en particular; *Pseudomonas aeruginosa*.

Es por ello que, en el presente trabajo, proponemos la implementación de algoritmos de clasificación basados en aprendizaje automatizado con el objetivo de identificar y clasificar IS en diversas especies bacterianas. Además de lograr de la identificación de IS, se buscará que dicho algoritmo reduzca el costo computacional y tiempo de ejecución al llevar a cabo la identificación de ISs con respecto al algoritmo BLAST. Dicho algoritmo se basa en el alineamiento de secuencias bajo el criterio de similitud entre ellas.

Objetivo Principal y Específicos

El presente trabajo tiene como principal objetivo diseñar e implementar un algoritmo de clasificación basado en aprendizaje automatizado, que nos permita clasificar e identificar potenciales IS sobre un gran volumen de genomas bacterianos. Dicho clasificador estará integrado en un *software* bioinformático para su uso en el campo de la genómica computacional.

Para lograr implementar el *pipeline*, los desafíos tanto a nivel metodológico como técnico se enlistan a continuación a modo de objetivos específicos:

- Alcanzar una adecuada representación de las secuencias aminoacídicas (tanto IS como de las otras proteínas) en los conjuntos de datos.
- Obtener atributos solidos de las secuencias aminocídicas
- Implementar un estimador capaz de identificar una potencial secuencia de inserción con un grado de exactitud aceptable, comparándolo con el algoritmo de alineamiento de secuencias BLAST.
- Desarrollar módulos específicos para cada etapa del *pipeline* e integrarlos en un programa.
- Identificar las IS evitando una alta demanda computacional (respecto a BLAST) y tiempos no superiores a las dos horas.

Materiales y Métodos

Conjuntos de datos

Tres conjuntos de datos o *datasets* fueron incluidos en el presente trabajo. El primer *dataset* consistió en una matriz de tipo "*m x n*" conformada por 16456 registros y 3 columnas. Las columnas representan la descripción de la proteína, su respectiva secuencia aminoacídica y la variable dependiente, respectivamente. El esquema del conjunto de datos está representado en la Tabla 1. La variable dependiente, denominada en *el dataset* como "target", representa la presencia o ausencia de secuencias de inserción (IS) y la misma fue de tipo binaria. Donde los valores 1 y 0, codifican para la presencia o ausencia de IS respectivamente o, dicho de otro modo, si el valor es 1 se trata de una IS, de otro modo, el valor es 0. Cabe notar que la distribución de la variable dependiente fue de tipo balanceada, es decir, los datos estaban compuestos equitativamente por IS y secuencias pertenecientes a otro tipo de macromoléculas, denominadas non-IS. En cuanto a las instancias correspondientes a IS (8228 secuencias de aminoácidos), éstas fueron descargadas de la base de datos ISFinder (Siguier y cols., 2016*),* un repositorio que alberga exclusivamente datos correspondientes a IS bacterianas correspondientes a todas las especies.

Dado a que, uno de los objetivos específicos del trabajo fue entrenar un clasificador de *machine learning* para estimar y predecir IS, fue necesario complementar el *dataset* con secuencias que no correspondieran a dicha población. Para llevar a cabo la tarea se descargaron secuencias aminoacídicas correspondientes a diferentes tipos de proteínas procariotas, que no fuesen IS. Para lo cual se recurrió al repositorio público PDB (del inglés Protein Data Bank, www.pdb.org) (Berman y cols., 2000). El criterio de selección, fueron proteínas que no fuesen recombinasas o transposasas. Es decir, este tipo de proteínas fueron excluídas. Dado a que el esquema del *dataset* de IS de partida estaba conformado por la anotación y las secuencias aminoacídicas, solo se tuvieron en cuenta estos dos campos en los datos de proteínas descargados del PDB.

De este modo, sobre el total de datos descargados se seleccionaron 8228 secuencias de un modo determinístico, con el fin de crear un conjunto de datos final de trabajo en el cual la variable dependiente presentase una distribución balanceada, totalizando 16456 instancias.

Tabla 1. Esquema del conjunto de datos de trabajo para entrenamiento y testeo.

Nombre de columna	Definición	Tipo de dato
Anotación	Referencia al tipo de proteína	String
Secuencia	Secuencia aminoacídica	String
Target	Variable dependiente	Integer

El segundo conjunto de datos consistió en cinco genomas de especies bacterianas que son considerados de referencia y se los denominó genomas de referencia. Dichos genomas corresponden a las especies *Escherichia coli*, *Acinetobacter baumannii*, *Salmonella enterica*, *Staphylococcus aureus* Newman y *Pseudomonas aeruginosa.* En cuanto a la estructura de los dichos datos, el esquema estuvo definido por una secuencia, y su respectiva anotación. Este segundo conjunto de datos fue utilizado en la segunda etapa del diseño experimental, denominada “validación”, la cual será posteriormente detallada.

Por último, el tercer conjunto de datos estuvo conformado por una colección completa de genomas bacterianos de la especie *E. coli.* Dicho *dataset*, consistió en un esquema similar al de los genomas de referencia y fué el utilizado en la etapa de Escalamiento y Despliegue que se detallara posteriormente.

Diseño experimental

En el presente trabajo, el desarrollo del software se ha subdividido en dos etapas principales; en la primera, la preparación y selección del algoritmo de clasificación, y en la segunda, el despliegue (*deploy*) y escalamiento del algoritmo sobre una especie bacteriana en particular que incluye número elevado de genomas completos.

Diseño experimental: primer etapa

La primer etapa del estudio estuvo enfocada en seleccionar el algoritmo que presentase mayor robustez y los valores más elevados en términos de métricas de evaluación de clasificación (descritas en la próxima sección). De este modo, la primera etapa se subdividió en tres fases: *Training* (entrenamiento), *Testing* (testeo) y Validación (sobre genomas de referencia). Donde, las primeras dos fases, se desarrollaron sobre el conjunto de datos de partida, en tanto que la etapa de validación se llevó a cabo sobre genomas de referencia. Dichos genomas, son datos a los cuales los algoritmos nunca fueron expuestos previamente.

Así, inicialmente los esfuerzos se centralizaron en el procesamiento de los datos de entrada. Es decir, en el tratamiento de las secuencias aminoacídicas y en su representación. Luego, se hizo foco en entrenar los diferentes estimadores sobre éste conjunto de datos generado (primer *dataset*). A esta etapa se la denominó “training”.

Posteriormente, en la etapa de “testing”, por medio de la metodología de validación cruzada en cinco campos, o “*five-fold cross-validation*”, se llevó a cabo la evaluación de los resultados de la clasificación generada por los algoritmos incluidos en el estudio (Figura 3).

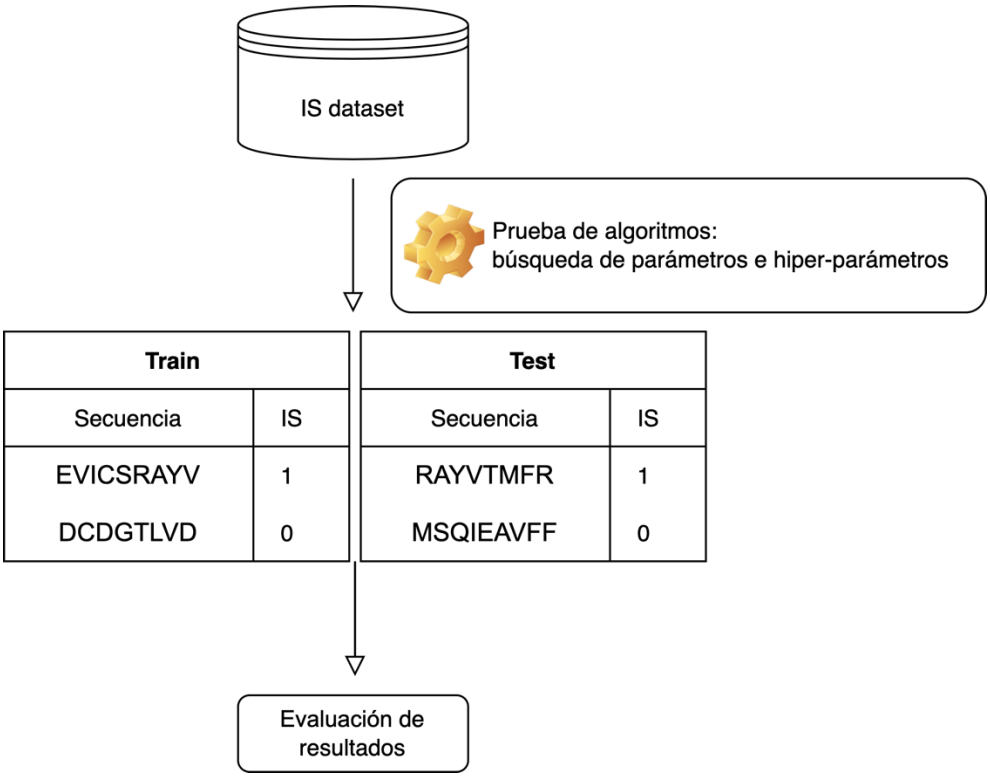

Figura 3. Ejecución de los diversos clasificadores sobre el *dataset* de partida.

En la última etapa de la primera fase, se incluyeron cinco genomas de referencia y, sobre las proteínas anotadas en estos genomas, se ejecutaron los diversos algoritmos de clasificación. A estos genomas se los denominó de “referencia” debido a que sus anotaciones han sido validadas y curadas en diferentes reportes mediante ensayos experimentales y son ampliamente utilizadas como referencia para diversos estudios microbiológicos (Figura 4). Estos genomas fueron descargados de la base de datos GenBank (www.ncbi.nlm.nih.gov/genbank/). A continuación, se detallan los cinco genomas bacterianos de referencia y su número de acceso de la secuencia nucleotídica en GenBank:

- *E. coli* K-12 (Numero de Acceso: NC_000913.3).
- *A. baumannii* AYE (Numero de Acceso: CU459141).
- *S. enterica* serovar Typhi CT18 (Numero de Acceso: AL513382.1).
- *S. aureus* Newman (Numero de Acceso: AP009351.1).
- *P. aeruginosa* PAO1 (Numero de Acceso: AE004091.2).

El archivo correspondiente a cada genoma fue descargado en formato FASTA, que es el tipo de formato más utilizado para representar, tanto secuencias de nucleótidos como de aminoácidos. Este tipo de archivo está conformado por una línea que hace referencia a la descripción y otra línea donde se encuentra la secuencia. El comienzo de un archivo FASTA está delimitado por el símbolo matemático “>”, a partir del cual se hace referencia a la anotación o descripción de la secuencia o anotación, la cual puede incluir: el nombre, número de acceso, código Entrez, etc. Es importante notar que fue necesario crear funciones específicas dentro del software a modo de procesar estas secuencias FASTA con el fin de ser representadas en un matriz, siguiendo el esquema detallado en la Tabla 1.

Por último, la robustez de los resultados obtenidos por los clasificadores se estudió mediante las métricas de rendimiento en clasificación. Dado a que, uno de los objetivos planteados en el presente trabajo fué identificar IS a un tiempo de ejecución aceptable, no solo se tuvieron en cuenta las métricas para evaluar la clasificación sino el tiempo de cómputo requerido por cada estimador para realizar dicha tarea.

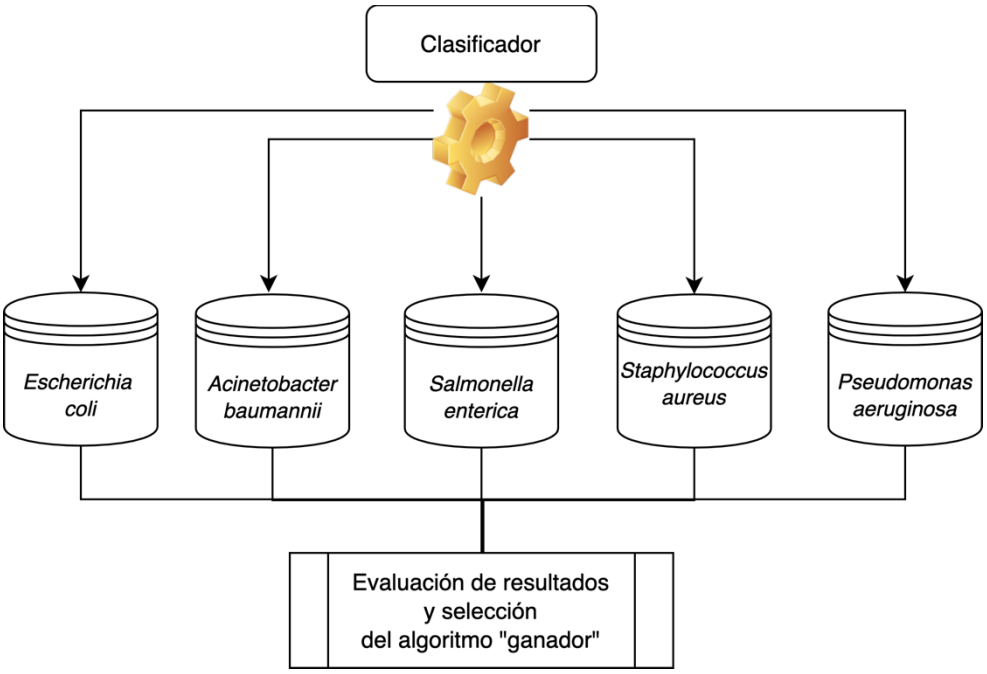

Figura 4. Etapa de validación: ejecución de los clasificadores sobre los genomas de referencia y selección del mejor clasificador.

Diseño experimental: segunda etapa

En cuanto la etapa escalado y despliegue, dentro de la gama de algoritmos incluidos en los experimentos en la primer fase, aquel estimador con mejores métricas de clasificación será el seleccionado para posteriormente ser ejecutado sobre una gran cantidad de genomas de especies de interés (Figura 5). En el presente trabajo la especie de interés estudiada en esta última etapa fue *E. coli*. Para lo cual, se descargaron 1197 genomas completos de dicha especie desde la base GenBank. Como parte del software desarrollado, se diseñó un módulo específico para la descarga de dichos genomas bajo protocolo FTP. Tanto la etapa de escalamiento como el diseño del *scraper* de genomas (descarga automatizada de secuencias genómicas y la información asociada a ella), se describirán en más detalle en secciones subsiguientes. Una vez que todos los genomas de una especie se encuentran en el entorno local (ordenador portátil), donde cada genoma está localizado en su directorio específico previamente generado, el algoritmo recorre cada directorio y realiza una ejecución (clasificación) asignando la clase estimada a cada proteína codificada por cada gen.

Las métricas de rendimiento (o en inglés, “performance”), son también implementadas sobre esta etapa. De este modo, es posible cuantificar el rendimiento del algoritmo a nivel de genoma de una determinada especie.

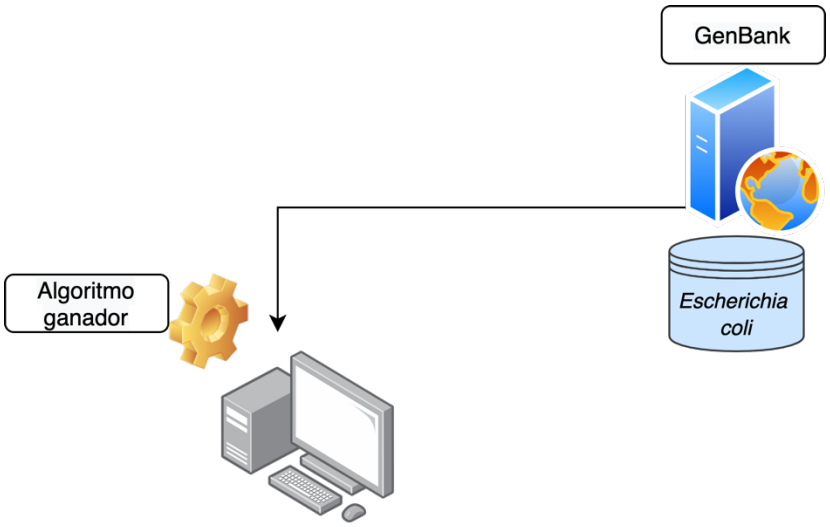

Figura 5. Etapa de escalado, en la cual el algoritmo de clasificador ganador (con mejores métricas) es desplegado sobre el genoma de *Escherichia coli*.

Procesamiento de secuencias aminoacídicas: extracción de atributos.

Como primera medida, para llevar a cabo el tratamiento de las secuencias, se trabajó con el método CountVectorizer dentro del *framework* Scikit-learn (Pedregosa y cols., 2011). Por definición, dicho método realiza una transformación de los datos de entrada, en una matriz que representa el recuento de términos, o, dicho de otro modo, la frecuencia relativa de *tokens* sobre los datos de entrada. Como resultado, se genera una matriz rala, que puede ser denotada matemáticamente como *m x n*. Donde el término "m" representa las instancias y "n" las columnas. Cada instancia pertenece a las distintas secuencias y las columnas representan los términos o k-meros (sub-secuencias de caracteres o *substrings*) generados por el algoritmo CountVectorizer. Cabe notar que los términos en dicha matriz son luego representados en valores numéricos enteros. Dentro de CountVectorizer, hay dos funciones principales instanciadas: ***fit*** en la cual se ajusta el método, previamente instanciado en el programa informático, sobre los datos de entrada. En esta etapa es donde tiene lugar la construcción del vocabulario. En el presente caso, el método se centra sobre un *dataframe* compuesto de secuencias de aminoácidos. El segundo método en el algoritmo CountVectorizer, es el denominado ***transform***, el cual genera la secuencia-término.

Es importante notar que el algoritmo CountVectorizer, está conformado por una serie de parámetros que son necesarios optimizar. En el presente trabajo se hizo foco en tres de los mismos:

- ***n gram range:*** el cual está conformado por la tupla “min_n” y “max_n”, que representa los limites máximo y mínimo que va a definir el rango del n-grama a extraer.
- ***analyzer:*** hace referencia al tipo de atributos que se busca extraer, es decir si el n-grama será a nivel palabra o carácter.
- ***max_df:*** computa la frecuencia del término obtenido al momento de la construcción del vocabulario. En dicho parámetro se especifica un punto de corte o *threshold*, a partir del cual se ignoran aquellos términos o entidades que superen dicho umbral. De este modo, se logran remover aquellos términos que no aporten relevancia dado a su elevada frecuencia.

Descomposición en valores singulares

Con el objetivo de alcanzar una representación numérica de las secuencias de aminoácidos presentes en todos los conjuntos de datos utilizados en el presente trabajo, fue necesario implementar abordajes que permitan transformar dichos datos de entrada (*strings*) hacia matrices numéricas. Es decir, de secuencias conformadas por caracteres a matrices compuestas de números. Este proceso se llevará a cabo mediante la “vectorización” de dichos *strings*, es decir, transformando las secuencias de aminoácidos en entidades representadas por un valor numérico (*int*). No obstante, dado que la matriz generada de dicha vectorización es de tipo rala, es necesario implementar métodos de reducción de dimensiones con el objeto de sintetizar la información generada.

Es por ello que se implementó la metodología de reducción de dimensiones denominada Descomposición de Valores Singulares (SVD, en Inglés Singular Value Decomposition). En la SVD, una matriz de entrada original es descompuesta en un set de K factores ortogonales a partir de los cuales, la matriz original, puede ser aproximada por combinación lineal.

Las matrices que se generan son tres, la **matriz U**, ortogonal, de tamaño *m x k*, dicha matriz contiene los vectores singulares(*s*) “izquierdos” (left singular vectors), los cuales representan a las columnas de la matriz de entrada (n) o *k* valores más grandes de *s*. La **matriz sigma** (Σ, de tipo *k x k*, la cual alberga los valores singulares, y concentra todos valores nulos excepto en su diagonal. Dichos elementos singulares están ordenados de manera decreciente, el valor singular más elevado se localiza en primer lugar en el ángulo superior izquierdo (cuadrante II del plano cartesiano). El número de valores singulares o *s-values* están determinados por *k*. Por último, la **matriz V** transpuesta, es ortogonal, y sus columnas contienen los valores singulares “derechos” (right singular vectors) y está representada como *n x k*.

En el presente trabajo se implementó una variante de SVD, denominado Randomized-SVD disponible en el paquete Scikit-learn. Dicho algoritmo fue diseñado por Halko y colaboradores (Halko y cols., 2011) con el objetivo de generar una aproximación de rango menor (*low-rank approximation*) a partir de una matriz rala de gran tamaño. Por lo tanto, en vez de aplicar la metodología SVD de una forma determinística sobre una matriz de entrada, la cual es computacionalmente costoso requiriendo un uso intensivo de memoria, la variante Randomized-SVD comienza generando un sub-espacio pequeño y de manera aleatoria, y posteriormente proyecta la matriz de entrada original en dicho subespacio. Así, las características más importantes de la matriz de entrada son condensadas en el subespacio aleatorio de menor tamaño. Este subespacio proyectado se transforma en una alternativa viable para aproximar la factorización o descomposición de la matriz evitando un alto costo computacional.

Métricas de evaluación de rendimiento en clasificación

En la etapa de la preparación del algoritmo, dado a que la distribución de la variable dependiente, de tipo binaria, fue balanceada, la métrica seleccionada para determinar rendimiento en clasificación fue *Accuracy* (o Exactitud) (Tabla 2). Por otro lado, teniendo en cuenta que el objetivo del presente trabajo es implementar un algoritmo capaz de predecir y clasificar a una secuencia aminoacídica como secuencia de inserción bacteriana, se decidió incluir las medidas de Sensibilidad y Especificidad. La determinación de estas dos métricas indica la eficiencia, la confiabilidad y reproducibilidad de la clasificación aplicada. En la Tabla 2 se describen las definiciones de dichas métricas. Para una mejor interpretación, la abreviación TP hace referencia

a resultado verdadero positivo (*True Positive*), TN a verdadero negativo (*True Negative*), FP a falso positivo (*False Positive*) y FN a falso negativo (*False Negative*).

La medida de F1 (también referenciada como F1 score), también fue incluida como métrica de evaluación en la fase de clasificación sólo sobre los genomas de referencia. Dicha métrica se puede definir como la media armónica entre la medida de Precisión y Sensibilidad (o *Recall*). Otra métrica incluida fue la curva ROC, la cual además nos permitió observar dicho rendimiento. La curva ROC es un gráfico de dos dimensiones el cual ilustra el desempeño de un clasificador conforme se va combinado el valor de corte que discrimina una variable dicotómica (en el caso del presente trabajo es, 1 o 0). En el gráfico, el eje X (o variable independiente) representa la tasa de falsos positivos para el test predictivo, en tanto que el eje Y (variable dependiente) es el valor de la tasa de verdaderos positivos. Cada punto en la curva es un par de datos VP/FP para un valor de punto de corte configurado en la prueba predictiva. La implementación de la curva ROC pudo llevarse a cabo mediante la función *sklearn.metrics.roc_curve*, disponible en Scikit-learn.

Tabla 2. Métricas de evaluación de performance en clasificación.

Métrica	**Definición**
Accuracy	TP + TN / TP + TN + FP + FN
Sensibilidad	TP / TP + FN
Especificidad	TN / TN + FP
F1	2* (P * R) / (P + R)

Algoritmos de clasificación

Los algoritmos de clasificación que se incluyeron en la primera etapa del desarrollo de la aplicación fueron Regresión Logística (Freedman, 2009), *Support Vector Machines* (SVM) (Cortes y Vapnik, 1995), *Stochastic Gradient Descent* (SGD) (Bottou, 1991) y métodos basados de ensambles. En este último grupo los algoritmos incluidos fueron *Xtreme Gradient Boosting*

(XGBoost) (Chen y Guestrin, 2016), *Random Forest* (Breiman, 2001) y *Light Gradient Boosting Machine* (LGBM) (Ke y cols., 1997).

Búsqueda de valores en parámetros

El valor óptimo de los diversos parámetros correspondientes a los clasificadores con los cuales se experimentó en el presente trabajo fue estimado mediante búsqueda aleatoria (o randomizada) en grilla. Ha sido demostrado empírica y teóricamente que, la búsqueda de valores de los parámetros de manera aleatoria es más eficiente que de tipo secuencial en grilla (Bergstra y Bengio, 2012) Específicamente, se hizo uso del método "GridSearch" disponible en el framework Scikit-learn.

Considerando el procesamiento de las secuencias de partida y el orden en el *pipeline* propuesto, los parámetros a optimizar fueron: sobre el método CountVectorizer: "n gram range", "analyzer", "max_df", y, para SVD, la búsqueda de valores óptimos se hizo sobre el parámetro "K". En cuanto a los algoritmos de clasificación, los parámetros a optimizar se describirán a continuación, en función del algoritmo:

i) Regresión logística: solver, penalty, max_iter y C.
ii) SGD: loss, penalty, alpha y max_iter.
iii) SVM: C, kernel y gamma.
iv) *Random Forest*: n_estimators, min_samples_split, min_samples_leaf, max_features', max_depth, criterion y bootstrap.
v) LightGBM: objective, boosting_type, alpha, colsample_bytree, lambda_l1, lambda_l2, max_depth, min_child_samples, min_child_weight, n_estimators, num_leaves y subsample.
vi) XGBoost: objective, alpha, subsample, n_estimators, min_child_weight, max_depth, learning_rate, gamma y colsample_bytree.

Para ayudar a la comprensión de cada parámetro se describirá a continuación cada uno de ellos.

- *Solver:* hace referencia al tipo de algoritmo implementado dentro del clasificador cuyo objetivo es minimizar el error durante la tarea de clasificación.
- *Penalty:* es el tipo de regularización.

- *max iter*: hace referencia al *número* de iteraciones efectuadas por el *solver* hasta alcanzar convergencia.
- *C:* parámetro utilizado para reducir la regularización.
- *Loss*: parámetro mediante el cual se determina la función de loss.
- *Alpha:* constante que actúa como multiplicador del termino de regularización.
- *Kernel:* tipo de función o kernel utilizado.
- *Criterion*: método usado para cuantificar la pureza del nodo.
- *Boostrap*: método para seleccionar la fracción de instancias usadas durante la etapa de entrenamiento.
- *N_estimators:* número de estimadores.
- *Min samples split:* número de muestras requeridas para dividir un nodo interno.
- *Min samples leaf:* especifica la cantidad mínima de instancias requeridas para ser un nodo hoja.
- *Max_features leaf*: cantidad de atributos seleccionados de manera automática.
- *Max depth:* profundidad de los estimadores (árboles de decisión).
- *Objetctive:* función objetivo.
- *Boosting type:* tipo de boosting empleado.
- *Lambda 1 y Lambda 2*, tipos de regularización.
- *min_child_samples*: datos minimos necesitados en un nodo hoja.
- min_child_weight: suma minima de ponderaciones en la instancia.
- *num_leaves:* número de hojas.
- *Subsample:* proporcion de sub-muestreo.
- *Learning rate:* velocidad de aprendizaje.
- *Gamma:* parámetro de regularización.
- *colsample_bytree*: fraccion de atributos (seleccionados aleatoriamente) que serán usados en por cada árbol en la fase de entrenamiento.

Selección del algoritmo de clasificación

Se realizó un estudio comparativo entre todos los algoritmos de clasificación en función del rendimiento mediante las métricas de clasificación, sobre cada uno de los genomas de referencia. Además de las métricas descritas anteriormente, como Especificidad y Sensibilidad, se llevó a cabo un estudio de la distribución sobre las secuencias catalogadas como verdaderos positivos (o TP, del inglés “True Positives”). Esto, permitió observar con mayor nivel de granularidad, a nivel de familias de las IS, el rendimiento de cada algoritmo de clasificación, aumentando la consistencia al momento de seleccionar el algoritmo. Otro criterio que se consideró, además de la performance en clasificación, fue el tiempo de cómputo. Es decir, el tiempo en segundos que demandó cada algoritmo al momento de efectuar las clasificaciones. Este criterio tiene igual ponderación que la performance y la razón subyace en que el clasificador está integrado en un software que clasifica en tiempo real las secuencias de un determinado genoma. Si bien el clasificador entrenado se almacena en formato *pickle* (objeto serializado), el mismo demanda un determinado tiempo para ejecutar las clasificaciones. Otro punto a considerar es el número de secuencias que contiene cada genoma que se requiere clasificar. En el presente trabajo, el clasificador seleccionado se ejecutó sobre un conjunto de genomas completos de la especie *E. coli* el cual consta de más de mil secuencias codificantes para proteínas.

Calibración

Luego de seleccionado el algoritmo a utilizar, se procedió a implementar la etapa de calibración. En su mayoría, los algoritmos experimentados en el trabajo constan de un método que permite obtener las probabilidades generadas en la clasificación. Cabe notar que, por defecto, el algoritmo establece un punto de corte en el valor de 0,5. Así, en función de dicho umbral y de las probabilidades asignadas a la instancia, se determina la clase predicha. De modo que probabilidades con valores encima de dicho punto representan la clase positiva, en este caso secuencia IS (valor igual a 1). En tanto que, valores de probabilidad debajo de 0,5, asignan secuencias non-IS (0).

Es importante mencionar que las probabilidades producidas pueden ser interpretadas como un nivel de confianza. No obstante, dichas probabilidades deben surgir de un estimador

adecuadamente calibrado. Como contrapartida, un estimador no calibrado, es menos estable ante errores en situaciones donde los datos a clasificar presentan un fuerte desbalance en términos de la distribución de la variable dependiente. En el presente trabajo, dicho escenario tiene lugar cuando el clasificador es ejecutado sobre los genomas de referencia, en donde la frecuencia de IS presentes es considerablemente baja.

El tipo de calibración implementado en el presente trabajo fue mediante regresión isotónica (Zadrozny, 2012). Dicho abordaje es de tipo no paramétrico, en donde el objetivo es ajustar por etapas, una función no decreciente y constante. En otros términos, el ajuste de la función es de tipo escalonado. Este método puede ser considerado como una forma generalizada de *binning* o discretización con la ventaja de que no se requiere de un número predeterminado o límites en cuanto al tamaño de los contenedores o *bins* (Boström, 2008).

Despliegue y escalamiento

En la segunda etapa, el trabajo se focalizó en detectar nuevas IS sobre genomas de interés. Para llevar a cabo esta tarea, el algoritmo de clasificación puesto a punto en la primera fase del estudio se ejecutó sobre secuencias correspondientes a genomas bacterianos de especies conocidas que fueron descargados del repositorio GenBank (Figura 5). Con el fin de poder implementar la clasificación de manera consecutiva sobre cada genoma descargado, se desarrollaron módulos que se describen a continuación.

1.Descarga automatizada de genomas y procesamiento.

Una vez entrenado y calibrado el estimador sobre el conjunto de datos de partida (Conjunto de secuencias de IS y non-IS), el *software* se centralizó en la clasificación e identificación de potenciales nuevas IS. Para ello el clasificador se ejecutó sobre secuencias de genomas bacterianos descargadas del repositorio GenBank.

Para la obtención de dichos genomas se procedió a la descarga de los archivos FASTA mediante el protocolo de tipo FTP (del inglés, File Transfer Protocol) sobre el servidor de GenBank. Para lo cual, se diseñó un módulo específico dentro del *software* el cual permitió automatizar dicha

tarea. Una vez descargados todas las proteínas codificantes de los genomas, en formato FASTA y comprimidos de extension .gz. , el programa tiene las instrucciones para descomprimir el archivo, crear un directorio con el nombre del genoma (denominado “Strain”), y dentro de ese directorio convertir desde formato FASTA a una matriz de tipo *data frame* . El esquema de datos resultante consiste en: la descripción de la secuencia, su anotación y la secuencia aminoacídica. El tipo de datos, para todos los campos, es de tipo *string* (compuesto por caracteres*)* y el esquema resultante puede observarse en la Tabla 3.

Tabla 3. Ejemplo del esquema resultante en datos de los genomas.

Description	Annotation	Sequence
QKB78294.16 phosphogluconae phosphatase	6-phosphogluconate	MSQIEAVFFDCDGTLVDSEVIC SRAYVTMFR
QCN07788.1IS66-like element ISEc8 family transposase	IS66	MIPLPSGTKIWLVAGITDMRNG FNGLA

2.Procesamiento de secuencias y clasificación

Luego de generado el *dataset* correspondiente a cada genoma como resultado del paso previo, el *script* recorre dichos datos y, basándose en la anotación asignada por defecto por los investigadores, se asigna un valor de 1 o 0 a una nueva columna denominada target (la cual corresponde a la variable dependiente). Cabe notar que la anotación de un genoma consiste en la asignación funcional de un gen (secuencia codificante para una proteína determinada) a una secuencia con una característica particular.

Así, se asigna el valor de 1, si la anotación corresponde a una secuencia de inserción (IS) y 0, si no lo es (non-IS). El *dataset* resultante que, a su vez, será la matriz de entrada para el clasificador, se detalla en la Tabla 4.

Tabla 4. Ejemplo del dataset con la clase asignada basado en la anotación por defecto. Este conjunto de datos corresponde a un genoma específico y es donde se centra el clasificador en la etapa siguiente.

Description	Annotation	Sequence	Target
QKB78294.16 phosphogluconae phosphatase	6-phosphogluconate	MSQIEAVFFDCDGTLVD SEVICSRAYVTMFR	0
QCN07788.1IS66-like element ISEc8 family transposase	IS66	MIPLPSGTKIWLVAGITD MRNGFNGLA	1

3.Selección de IS potenciales

En esta etapa se hizo foco en los falsos positivos (FP), potenciales IS, resultantes de la clasificación sobre cada genoma. Es decir, en aquellas secuencias que el algoritmo clasificó como 1, identificándose como secuencias de inserción (IS), pero que fueron anotadas inicialmente como otro tipo de secuencias (non-IS). Se tuvo interés en los FP, dado a que la anotación de genomas es un proceso con una alta variabilidad. Por último, los FP de todos los genomas fueron recolectados y compilados en un *dataset* único, que luego será usado como archivo de entrada para la etapa siguiente.

4.Búsqueda de homología de secuencias

A las potenciales IS identificadas en el paso anterior fueron sometidas a un análisis de búsqueda de similitud mediante el software BLAST, que se basa en un alineamiento local para establecer el grado de similitud. El objetivo de esta etapa es cotejar las IS identificadas mediante el algoritmo y el resultado arrojado mediante a alineamiento. Es decir, a las secuencias que el algoritmo catalogo como positivo para IS, se las sometió luego a un análisis mediante BLAST con criterios configurados para detectar secuencias de Insersión.

Como fue detallado anteriormente dicho algoritmo BLAT requiere de secuencias “patrón” usadas como referencia al momento de comparar la secuencia objetivo o query. En este caso se utilizó como referencia la base de datos ISFinder, la cual alberga secuencias de inserción bacterianas.

El algoritmo BLAST fue ejecutado bajo parámetros específicos que se implementaron a modo de puntos de corte (*threshold*):

a) cobertura de alineamiento sobre el *“subject”* (base de datos ISFinder) mayor o igual al 70%.

b) identidad a nivel aminoacídico (grado de semejanza entre secuencias) mayor o igual al 30%.

Estos parámetros nos permitieron identificar nuevas IS en un amplio rango de identidad (Traglia y cols., 2018). Cabe notar, éstos valores de parámetros son criterios estándares para IS. Para realizar las ejecuciones del algoritmo BLAST se utilizó el entorno de trabajo o *framework* denominado BioPython (Cock y cols., 2009). Dentro de dicho entorno se hizo uso del módulo específico Bio Blast Applications (Camacho y cols., 2009).

Tabla 5. Esquema de dataset de salida como resultado del alineamiento mediante BLAST.

Nombre de variable	Tipo de dato	Descripción
query	*string*	Secuencia objetivo
sequence_matched	*string*	Secuencia presente en la base de datos que presento una determinada identidad frente a la secuencia blanco o incognita
id_aminoacid_perc	*float*	Porcentaje de identidad aminoacidica
aligment_lenght	*integer*	Longitud de alineamiento
mismatch	*integer*	Cantidad de diferencias aminoacidicas resultantes del alineamiento

gaps	*integer*	Número total de gaps
start_query	*integer*	Inicio del alineamiento de la secuencia que se busca alinear
end_query	*integer*	Final del alineamiento de la secuencia que se busca alinear
start_subject	*integer*	Inicio de la secuencia de referencia presente en la base de datos
end_subject	*numeric*	Final de la secuencia de referencia presente en la base de datos
e_value	*float*	Parámetro que describe el número de aciertos “esperados” por aleatoriedad cuando se consulta una base de datos especifica
bit_score	*float*	Tamaño requerido de una base de datos de secuencia en la que la coincidencia (match) podria encontrarse por casualidad
slen	*numeric*	Loingitud de la secuencia de referencia
coverage_perc	*float*	Porcentage de covertura

Cada una de las etapas descritas a lo largo del escalamiento y despliegue, se puede apreciar en el diagrama de flujo de la Figura 6.

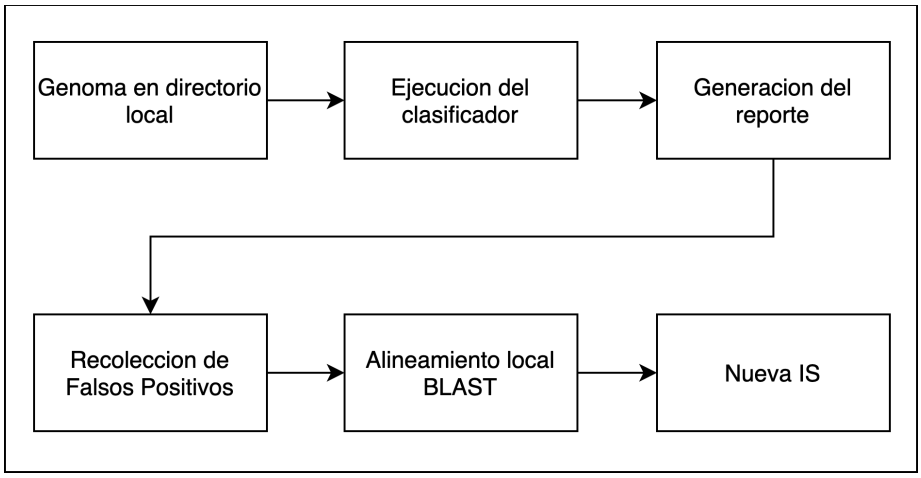

Figura 6. Pipeline en la etapa de escalamiento y despliegue

Lenguaje computacional

El presente trabajo se llevó a cabo en lenguaje Python (*Python Software Foundation. Python Language Reference*, versión 3.8.5 disponible en www.python.org). Scikit-learn se utilizó como marco de trabajo en donde se desarrollaron las metodologías de *machine learning* implementadas (Pedregosa y cols., 2011). Por último, la librería Biopython (Cock y cols., 2009) nos permitió realizar ejecuciones del algoritmo BLAST, para alineamiento de secuencias, dentro del entorno Python.

Arquitectura del software

Se crearon módulos para llevar a cabo las diferentes etapas lo que permite trabajar de modo ordenado, manteniendo de forma simple la trazabilidad de errores. Como resultado de la estrategia la descarga masiva de genomas, el procesamiento de los genomas, la ejecución del clasificador (puesto a punto) y el alineamiento por BLAST en los potenciales falsos positivos, estuvieron alojados en diferentes módulos.

Disponibilidad del código del flujo de trabajo

Los *scripts* generados para la implementación del flujo de trabajo estarán disponibles de forma pública y libre en el repositorio GitHub (https://github.com/Mangel1782/IS_target_Ecoli).

Resultados

Análisis descriptivo de secuencias de inserción

El conjunto de datos de trabajo estuvo compuesto por 8228 secuencias aminoacídicas correspondientes a secuencias de inserción (IS) y 8228 que no pertenecían a secuencias de inserción (non-IS). Las 8228 IS, estuvieron distribuidas en 29 familias. Además, se definió como "tipo" a las diferentes secuencias basadas en la variabilidad existente la secuencia aminoacídica

Con el fin de estudiar la distribución de los tipos de IS, se establecieron tres categorías en función de su frecuencia relativa sobre los datos. Así, los valores que determinan las categorías "baja"," media" y "alta" fueron obtenidos de modo arbitrario, en base a la distribución de su frecuencia y se detallan en la Tabla 6. Así, se puede observar que la categoría "baja" se conformó de tipos de IS que estuvieron presentes por única vez, la categoría "media" consistió en tipos con frecuencia de entre dos y cuatro, y la categoría "alta" estuvo representada por aquellos tipos con frecuencia mayor o igual a cinco (>=5).

Tabla 6. Descripción y distribución de las categorías establecidas para las enzimas o proteínas pertenecientes al grupo de IS

Frecuencia	Categoría	Distribución (%)
1	baja	3670 (69)
2-4	media	1617 (30)
>= 5	alta	32 (1)

De este modo, las secuencias que corresponden a una único tipo y familia a fueron 3670 (69%). En tanto que 1617 (30%) IS estuvieron representadas entre 2 y 4 familias, y, finalmente, 32 (1%) secuencias estuvieron distribuidas entre más de 5 familias. En la Figura 7 se puede observar la distribución de los tres tipos de categorías.

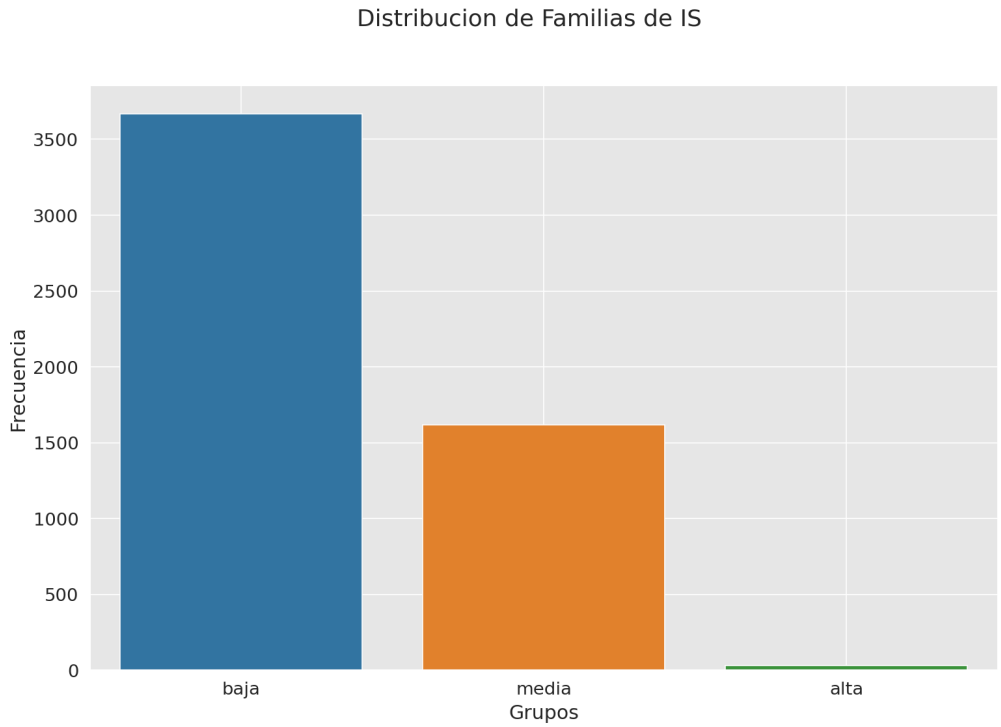

Figura 7. Distribución de los grupos de familias de IS en función de su frecuencia en el conjunto de datos.

A modo de ejemplo, dentro de los tipos con alta frecuencia se pueden mencionar ISArsp9, ISCausp2 y ISCausp3 (Figura 8). En los tipos de frecuencia media se pueden enunciar ISSme3, ISBth5, y ISSce1respectivamente. Por último, dentro los tipos de baja frecuencia se identifican las secuencias IS-LL6, ISNaoc3 y ISNaoc6.

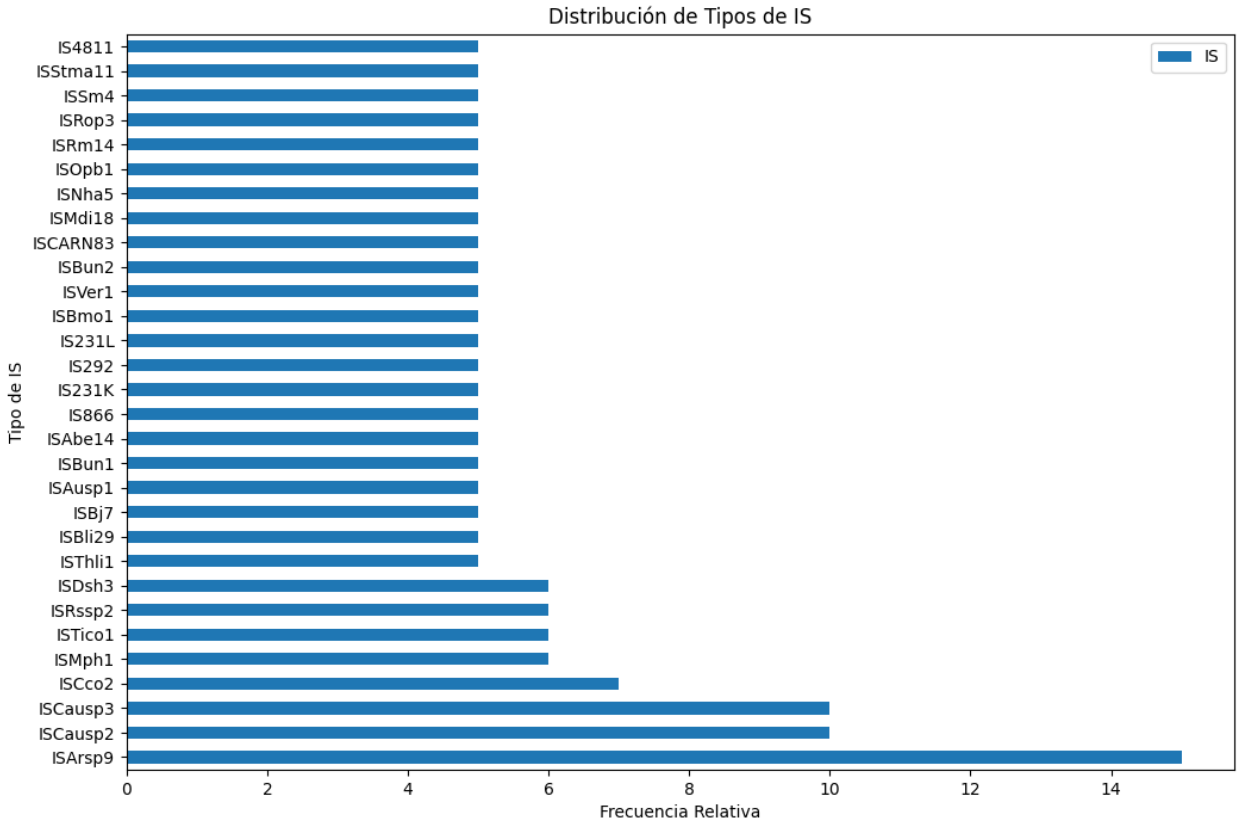

Figura 8. Distribución de tipos de secuencias de inserción presentes en el conjunto de datos de partida.

Variabilidad en la longitud de residuos de aminoácidos en las IS

La longitud de residuos aminoacídicos en cada proteína o secuencia de inserción no es constante, tanto dentro como entre las familias. Por éste motivo se estudió la variabilidad en términos de la longitud. Para un mejor entendimiento, en éste análisis descriptivo se tuvo en cuenta las categorías de IS definidas en la sección anterior (en función de la frecuencia). En la Tabla 7 se detallan los estadísticos descriptivos basados en medidas de tendencia central sobre la variable “longitud de aminoácidos”.

Tabla 7. Medidas de tendencia central en para la variable longitud de aminoácidos. Dichas medidas se estudiaron bajo el criterio de la categoría de familia de IS a la que pertenecían.

Categoría de familia de IS	Media	Desvío Estándar	Mediana
Baja	373,1	83,7	376
Media	269,7	136,5	273,5
Alta	214,4	156,3	175,5

Continuando con el análisis descriptivo inicial, se estudió la distribución de la longitud de aminoácidos sobre cada categoría. De este modo, el estadístico descriptivo utilizado para cuantificar normalidad de distribución fue el Test de Shapiro, con un nivel de significancia de 0,05 (α = 0,05). Así, en cuanto al estudio de distribución del número de aminoácidos en la categoría baja, la variable para el recuento de aminoácidos sigue una distribución de tipo no gaussiana (p<0,05). En cuanto a la asimetría de la distribución, el valor de skewness fue de -0,21. En referencia a la forma de la curva, el estadístico de curtosis arrojó un valor de 0,11 (Figura 9). En la categoría media, también se observó una distribución no gaussiana (p<0,05) con un sesgo o skewness de 0,32 y una curtosis de -0,42 (Figura 9). En la categoría de frecuencia alta, si bien se observó una distribución de tipo no Gaussiana, (p<0,05), el valor de skewness observado fue de 2,38 y el valor para curtosis fue de 9,42.

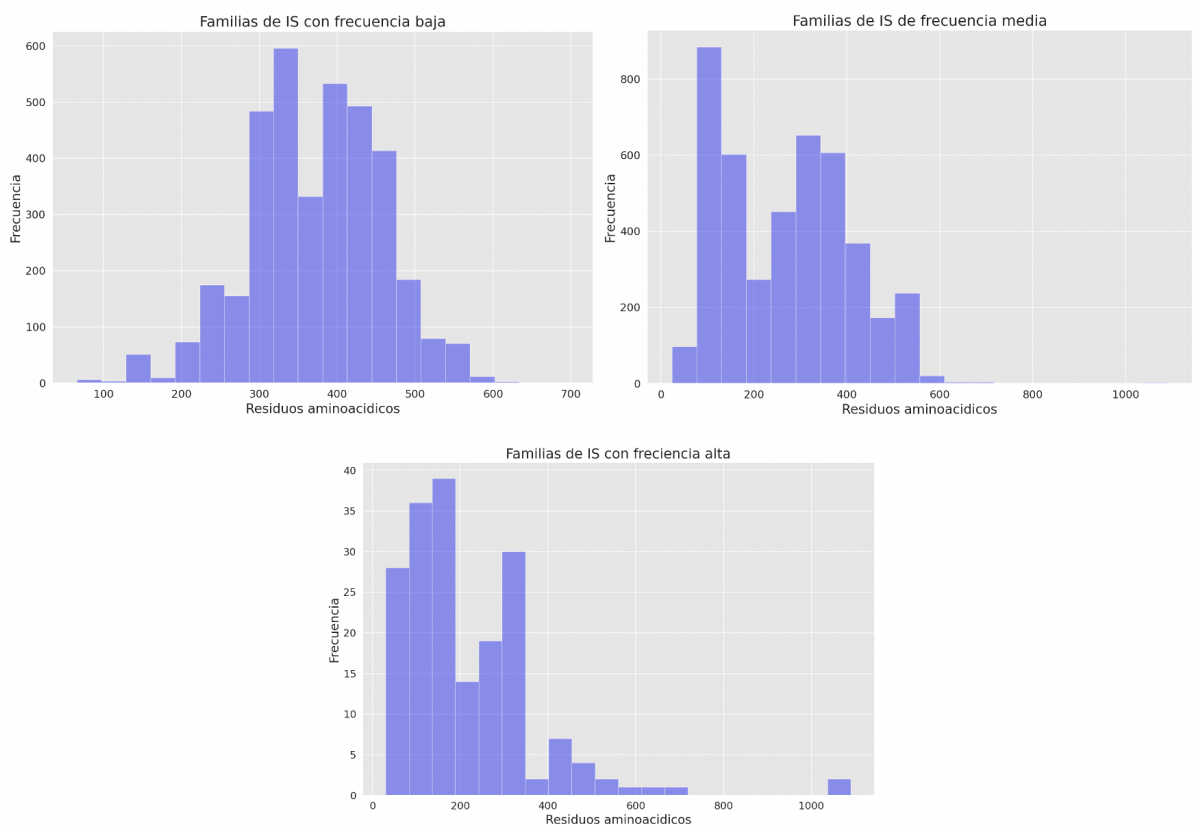

Figura 9. Longitud de secuencias en función de las categorías.

NON-IS

Para el caso de las secuencias aminoacídicas que no pertenecían a las IS (non-IS), las 8228 secuencias se distribuyeron en 901 tipos de enzimas o proteínas en base a su funcion. En este caso, y a diferencia de las secuencias de inserción, no hubo distribución en tipos de familias. Esto se debe a que dichas secuencias no corresponden a un tipo único de proteína con similar funcionalidad como en el caso de las IS.

Como se puede observar en la Figura 10, los tipos de enzima con mayor número de secuencias en el set de datos fueron Hidrolasa, Transferasa y Oxidoreductasa. En un segundo subgrupo, en función del número de secuencias, se puede incluir a proteínas cuya funcionalidad está vinculada al transporte intracelular, así como a factores de transcripción.

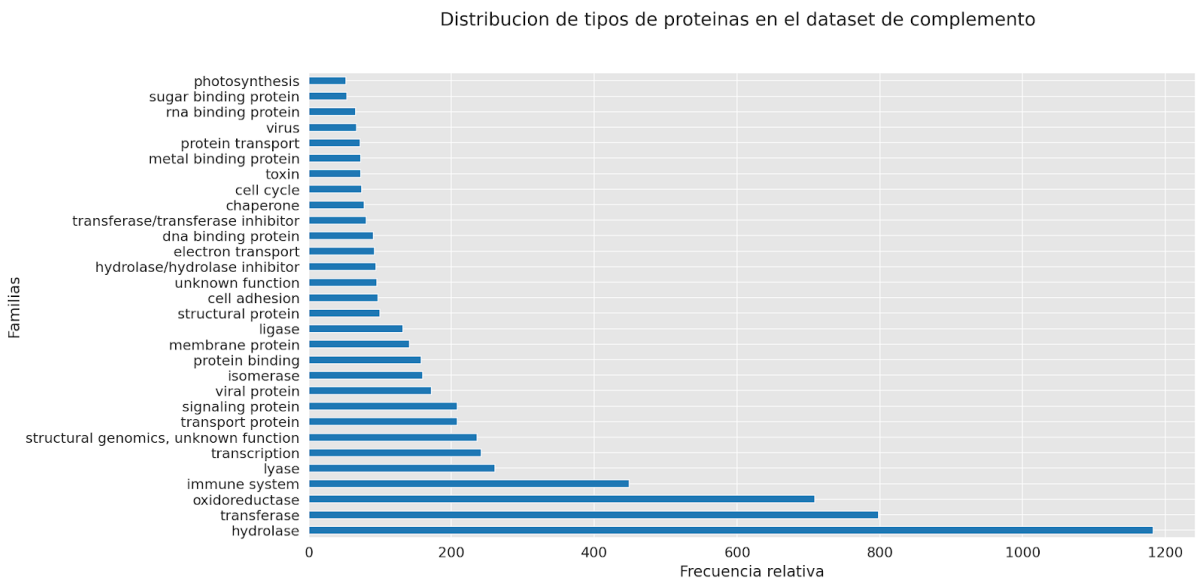

Figura 10. Distribución de tipos de proteínas y enzimas presentes en la clase complementarias (non-IS).

Del mismo modo en que se abordó el estudio de las familias en las IS, se establecieron las mismas tres categorías en base a la frecuencia: baja, media y alta (Figura 11). en función de la frecuencia relativa de los tipos de proteínas (en vez de familias como en el caso de las IS). Cabe notar que los rangos de las frecuencias, para generar los grupos, se establecieron de manera arbitraria. Los tipos de proteínas previamente se encuentran definidas en la base de datos PDB. Los puntos de corte determinados por los valores de la distribución de la frecuencia están detallados en la Tabla 8. Los niveles fueron establecidos en función de la distribución de la variable.

Tabla 8. Descripción y distribución de las categorías establecidas para las enzimas o proteínas pertenecientes al grupo non-IS.

Frecuencia	Categoría	Distribución (%)
1	baja	72,1
2-200	media	27,1
201-1200	alta	0,75

Las secuencias aminoacídicas que pertenecían a solo un tipo de proteína estuvieron representadas en un 72% en esta población. En tanto que las categorías conformadas por entre 2 y 200 tipos representaron un 27% y las de frecuencia alta (entre 201 y 1200 tipos) contribuyeron en menor medida con un 0,75%

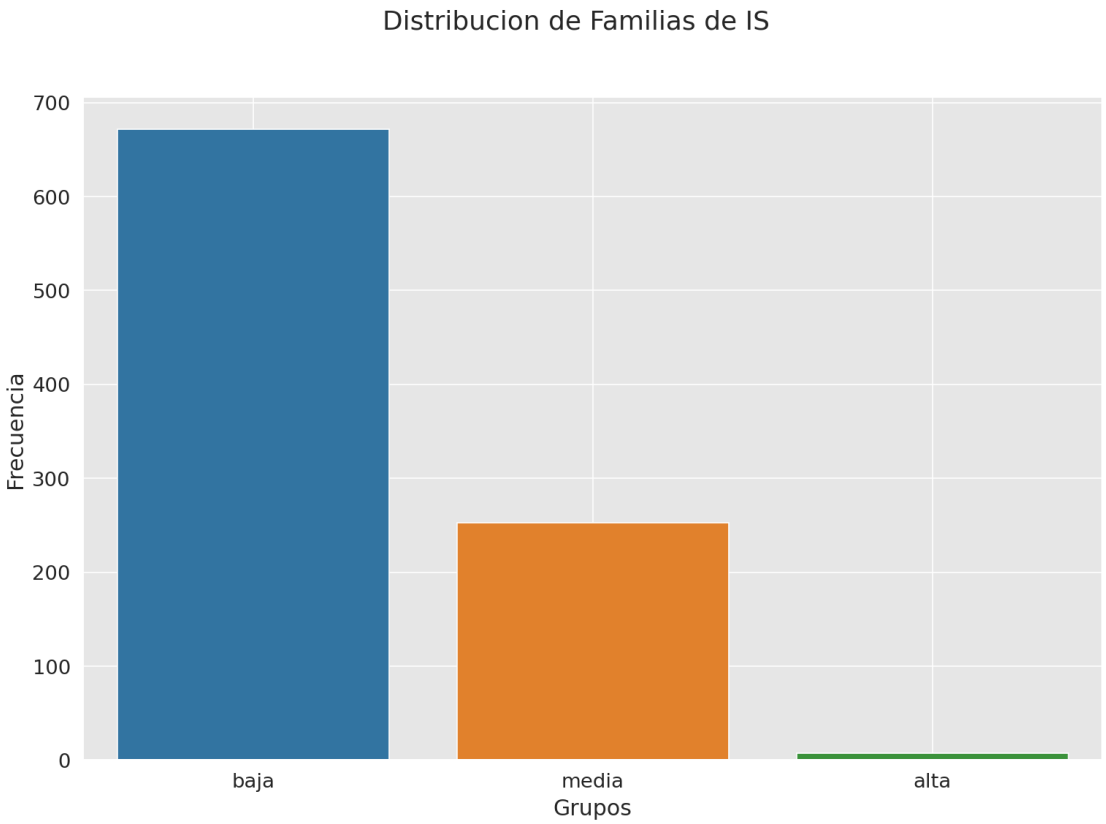

Figura 11. Distribución de categorías de IS.

Variabilidad de longitud de aminoácidos en secuencias non-IS

En términos de la longitud de cadenas de aminoácidos que contiene cada familia de non-IS, al igual que en las IS, los estadísticos descriptivos se presentan en la Tabla 9. Se agruparon en tres categorías arbitrarias según la longitud de aminoácidos. Al igual que en las IS, las longitudes de aminoácidos no seguían una distribución Gaussiana (Test de Shapiro, α = 0.05). En las secuencias non-IS, no se observó distribución de tipo gaussiana en ninguna de las tres categorías establecidas (p<0,05). En relación con la skewness, para las tres categorías alcanzaron un valor marcado mayor a 1. Dichos valores fueron 1,54, 1,94, 1,65 para las

categorías baja, media y alta respectivamente. Del mismo modo, los valores para curtosis fueron 3,75, 5,82 y 5,02 respectivamente (Tabla 10).

Tabla 9. Valores de la variable longitud de aminoácidos para los estadísticos descriptivos incluidos sobre las categorías establecidas para las enzimas-proteínas pertenecientes al grupo non-IS.

Categoría	Media	Desvío Estándar	Mediana
Baja	237,53	183,08	184,5
Media	227,06	172,77	182
Alta	297,79	175,74	264

Tabla 10. Longitud de aminoácidos y resultados de estadísticos asociados sobre los tipos de secuencia y sus categorías.

Clase	Categoría	Frecuencia	Distribución Gaussiana	Skewness	Curtosis
IS	baja	3670	No	-0,21	0,11
	media	4372	No	0,32	-0,42
	alta	186	No	2,38	9,49
non-IS	baja	656	No	1,54	3,75
	media	3752	No	1,94	5,82

	alta	3820	No	1,65	5,02

Genomas de referencia

En lo que respecta a los genomas que se utilizaron para la validación del modelo, estos correspondieron a cinco genomas altamente estudiados y utilizados en diversas publicaciones. En la Tabla 11, se detalla, para cada genoma, el nombre científico de la especie, la cepa correspondiente, el número de acceso, el número de genes y las frecuencias de las IS presentes (Se incluyeron las transposasas que no fueron previamente asignadas a algún IS conocida o anotada en base de datos ISFinder). Es decir, las secuencias objetivo a detectar por el clasificador. Es importante notar que ningún genoma tiene una proporción mayor a 1,67 % de IS.

El genoma con mayor número de genes corresponde a la especie *P. aeruginosa* (5570), la cual, a su vez, concentra el menor número de secuencias de inserción (6). Por otro lado, la especie *E. coli* presenta la mayor proporción de secuencias objetivo (IS), la cual fue de 1,67%, es decir, 71 IS sobre un total 4241 genes (Tabla 11).

Tabla 11. Genes y secuencias de inserción en genomas de validación.

Especie	Número de acceso de GenBank	Número de genes	IS presentes en genoma	Proporción de IS en genoma
***E. coli* K-12**	AN NC_000913.3	4241	71	1,67 %
***S. enterica* serovar Typhi CT18**	AN CU459141	4765	56	1,17 %
***S. aureus* Newman**	AN AL513382.1	2623	20	0,76%
***A. baumannii* AYE**	AN AP009351.1	3711	59	1,58%
P. aeruginosa PAO1	AN AE004091.2	5570	6	0,10 %

Extracción de atributos

Los datos de partida utilizados para extraer atributos incluyeron 16456 secuencias de proteínas, 8228 IS (ISFinder) y 8228 No-IS (PDB). Dichas secuencias son el resultado de la combinación de los 20 aminoácidos. CountVectorizer realizó una fragmentación de las secuencias aminoacídicas, generando *tokens* o entidades cuyo valor representa la frecuencia de esa entidad a lo largo de todo el conjunto de datos. De este modo, el producto resultante fue una matriz donde los registros o instancias representan cada una de las secuencias, y las columnas, las entidades o atributos generados. Por lo tanto, el valor asignado en cada uno de los atributos estuvo

determinado por la frecuencia de ese atributo (*token* de aminoácidos) en todo el *set* de datos de secuencias.

Las etapas posteriores a la vectorización de las secuencias aminoacídicas fueron: reducción de la dimensionalidad y la clasificación o identificación de IS mediante algoritmos de aprendizaje automatizado. En dichas etapas también se requirió el uso de valores específicos para los parámetros, inherentes al algoritmo SVD, así también como para el clasificador *per se*.

El entorno de trabajo Scikit-learn dispone de librerías específicas para llevar a cabo las tareas de armado del flujo de trabajo, ensamblando diversos métodos, creando así un único proceso secuencial denominado *pipeline,* así como, la búsqueda de valores óptimos en los diferentes parámetros de los algoritmos que conforman dicho proceso.

En el presente trabajo, el *pipeline* quedó integrado por las siguientes métodos y algoritmos:

- CountVectorizer: donde se trajeron los atributos a partir de las secuencias.
- SVD (Randomized-SVD): reducción de dimensiones.
- Algoritmo de clasificación: clasificación de IS.

Los algoritmos de clasificación utilizados fueron seis. La búsqueda de valores óptimos para los diversos parámetros presentes en el *pipeline* se puede abordar mediante dos alternativas: i) búsqueda secuencial y consecutiva, y ii) búsqueda aleatoria (randomizada). En ambas estrategias, dicha búsqueda se realiza sobre una lista de valores de parámetros definida con anterioridad.

En el presente trabajo, se optó por el segundo abordaje, denominado “Randomized Grid Search” (Bengio y cols., 2012). Es importante mencionar que en los seis algoritmos experimentados se usaron diferentes valores para los parametritos, lo cual generó diversos resultados tanto para la extracción de *atributos* como en la reducción de dimensiones. En la Tabla 12 se pueden observar los valores para los parámetros y el número de atributos extraídos sobre los *pipelines* generados para los seis clasificadores.

Tabla 12. Valores óptimos para los parámetros de la función CountVectorizer, en cada algoritmo experimentado.

Pipeline	Algoritmo	analyzer	ngram_range	max_df	Atributos extraidos
1	Logistic Regression	“char_wb”	(2,3)	1,0	9830
2	XGBoost	“char_wb”	(2,3)	1,0	431
3	SVM	“char_wb”	(2,3)	0,75	9826
4	LightGBM	“char_wb”	(2,2)	0,75	539
5	Random Forest	“char_wb”	(2,3)	0,75	9826
6	SGD	“char_wb”	(2,3)	0,25	9514

Por un lado, los valores usados en CountVectorizer dentro de los *pipelines* 2 y 4 generaron 431 y 539 *atributos,* respectivamente. En tanto que, los valores de los parámetros en el resto de los cuatro *pipelines* generaron más de 9,500 atributos.

La siguiente etapa dentro del *pipeline*, fue la reducción de dimensiones, para lo cual se implementó el algoritmo denominado Randomized-SVD.

Los valores óptimos para el parámetro K, de SVD y el porcentaje de la varianza explicada, respectivamente (Tabla 13). Como se puede ver, el mayor porcentaje de varianza se alcanza cuando el valor K se aproxima al valor de atributos. Esto se puede observar para el algoritmo LightGBM, donde extrajeron un total de 539 atributos, y el valor de K óptimo fue de 400. Como resultado obtuvimos un 99,96% de varianza explicada. Como contrapartida, en el *pipeline* donde se integró al algoritmo SVM, mediante CountVectorizer se extrajeron un total de 9826 *atributos*, el valor óptimo para K fue igual a 100 y la varianza explicada fue de 68,63% (Tabla 13).

El mejor valor fue el obtenido en el *pipeline* 2, donde se incluyó al algoritmo de clasificación XGBoost. En este caso, en la etapa del procesamiento, mediante CountVectorizer, se obtuvieron

431 atributos, el valor óptimo para K en el algoritmo SVD fue 85, siendo la varianza total explicada de 81,67%.

Tabla 13. Valor del parámetro K, en el algoritmo Randomized-SVD, y varianza explicada obtenida en los pipelines de los clasificadores incluidos.

Pipeline	Algoritmo	Atributos extraídos	SVD valor K	Varianza explicada (%)
1	Logistic Regression	9830	100	68,87
2	XGBoost	431	85	81,67
3	SVM	9826	100	68,63
4	LightGBM	539	400	99,96
5	Random Forest	9826	400	82,27
6	SGD	9514	125	70,37

Evaluación de la clasificación: etapa de *testing*.

Con el fin de evaluar el rendimiento de la clasificación y dado que la distribución de la variable dependiente, de tipo binaria, fue balanceada, la métrica seleccionada para la evaluación de la clasificación fueron Accuracy, Sensibilidad y Especificidad.

Como se mencionó en la sección “Materiales y Métodos”, con el objeto de generar robustez en los resultados generados por los clasificadores, se decidió trabajar con validación cruzada de cinco campos (*five-folds cross validation*). Dado a que se contó con una distribución balanceada, no fue necesario el uso de la estratificación de la clase de interés.

En cuanto a la métrica de *Accuracy*, en general, todos los clasificadores alcanzaron valores mayores a 90% (Tabla 14). Los clasificadores *Stochastic Gradient Descent* (SGD), *Random*

Forest y Regresión logística alcanzaron los menores valores de *Accuracy*, 91% y 92% respectivamente. Los algoritmos XGBoost y LightGBM se alcanzaron los valores de 94% y 95% respectivamente, colocándose en un rango medio. Por último, el clasificador SVM alcanzó el mayor valor para *Accuracy*, siendo 96%.

En términos de Especificidad, que se define como el cociente entre los verdaderos negativos (TN) y la sumatoria de los TN y Falsos Positivos (FP), los algoritmos con mejor desempeño fueron LightGBM y *Random Forest* alcanzando valores de 96,03% y 96,75%, respectivamente. Por su parte, SGD y XGBoost se ubicaron en una posición intermedia, con valores de 94,30% y 94.10%, respectivamente. Por último, el menor valor de Especificidad se obtuvo con la Regresión Logística, con 93,66%.

La Sensibilidad se definió como la razón entre los verdaderos positivos (TP) y la sumatoria de los falsos negativos (FN) y los TN. El clasificador que alcanzó el mayor valor fue LightGBM, 95,20%, en tanto que SVM y XGBoost estuvieron en un rango medio, 94,63% y 93,98%, respectivamente. Por último, los valores menores fueron obtenidos por SGD, Regresión Logística y Random Forest siendo métricas 91,86%, 91,73% y 90,95%, respectivamente.

Dado a que unos de los objetivos es evaluar el algoritmo una vez desplegado a gran escala sobre un pool significativo de genomas de una determinada especie, un parámetro inherente a la clasificación que debe ser considerado es el tiempo de cómputo.

Para el caso de la especie *E. coli*, que es la especie seleccionada para la etapa posterior, el número de genomas (completos) disponibles en GenBank es mayor a 1000 secuencias genómicas, el tiempo de cómputo juega un rol crucial. En nuestros experimentos, el algoritmo que logró menos tiempo de cómputo en la ejecución fue XGBoost. En total unos 15,5 segundos. En segundo lugar, la Regresión logística tomó 17,7 segundos y LightGBM demoró 55,2 segundos. Para el resto de los clasificadores, los tiempos fueron 202,8 segundos (3,38 minutos) en SGD, 18 minutos en *Random Forest* y 8,4 minutos para SVM (Tabla 14).

Tabla 14. Métricas en etapa de *testing*. Para cada algoritmo experimentado, se detallan los porcentajes de *Accuracy*, Especificidad y Sensitividad, así como el Tiempo requerido por cada clasificador.

Algoritmo	**Sensibilidad (%)**	**Especificidad (%)**	***Accuracy* (%)**	**Tiempo (segundos)**
Logistic Regression	91,73	93,66	92	17,70
XGBoost	93,98	94,10	94	15,50
SVM	94,63	93,66	96	504,00
LightGBM	95,20	96,75	95	55,20
Random Forest	90,95	96,03	92	1108,08
SGD	91,86	94,30	91	202,80

Etapa de validación

Con el objetivo de eliminar cualquier sesgo inherente al tipo de datos utilizado, en la etapa de entrenamiento, los algoritmos fueron sometidos a un paso adicional de validación. El rendimiento de los algoritmos de clasificación, cuyos valores óptimos para los parámetros fueron hallados mediante la estrategia de búsqueda en grilla y aleatorizada (*randomized grid search*), se validó mediante ejecuciones sobre genomas de referencia (descritas en la sección Materiales y Métodos). Dichos genomas corresponden a cepas bacterianas de especies modelo utilizadas en múltiples estudios experimentales.

Las funciones necesarias para transformar el formato FASTA a un objeto *dataframe* en Python, fueron incluidas dentro de un módulo específico desarrollado.

Para cuantificar el rendimiento de los diferentes algoritmos sobre los genomas de referencia, se contemplaron las mismas métricas de evaluación utilizadas en la etapa de *testing*: *Accuracy*, Especificidad y Sensibilidad (Tablas 15, 16 y 17).

Dado que, en los genomas de referencia, la distribución de la clase mostró un fuerte desbalance, se incluyó además la métrica de rendimiento F1 (Tabla 18).

La frecuencia de IS en los genomas de referencia fue diversa. Así, para el caso del genoma de *E. coli* K-12, la proporción de IS fue del 1,67% sobre el total de secuencias codificantes. En *S. enterica* serovar Typhi CT18 de 1,17%, para *S. aureus* Newman, *A. baumannii* AYE y *P. aeruginosa* PAO1, las proporciones fueron de 0,76%, 1,58% y 0,1%, respectivamente (Tabla 11).

Situándonos sobre la métrica *Accuracy*, la media observada (promedio entre todos los clasificadores) sobre el genoma *E. coli* K-12 fue de 81,11% (SD = 1,82), para *S. enterica* serovar Typhi CT18 fue 77,83% (SD=2,04), *S. aureus* Newmann 82,18% (SD = 2,38), *A. baumannii* AYE 80.74% (SD = 2,44), y por último *P. aeruginosa* PAO1 67.97 % (SD = 1,82).

Específicamente, para los genomas de E. *coli* K-12 y *S. enterica* serovar Typhi CT18, los algoritmos que mostraron valores más altos de *Accuracy* fueron *Random Forest* y XGBoost, alcanzando valores de 84,08% y 82,36% respectivamente. En el caso de *S. aureus* Newman, los clasificadores *Random Forest* y SVM fueron los que mostraron un mejor rendimiento, 84,67% y 84.29%, respectivamente. Para el genoma *A. baumannii* AYE, los algoritmos *Random Forest* y XGBoost obtuvieron 84,55% y 82,48%, respectivamente. Por último, en *P. aeruginosa* PAO1 solo el algoritmo SGD alcanzó una métrica de 71,23% (Tabla 15).

Tabla 15. *Accuracy* obtenida sobre cada uno de los genomas en la etapa de referencia.

	***Accuracy* (%)**				
Algoritmo	*E. coli*	*S. enterica*	*S. aureus*	*A. baumannii*	*P. aeruginosa*
Logistic Regression	80,42	76,60	82,80	77,68	68,63
XGBoost	82,36	78,86	82,76	82,48	66,28
SVM	81,72	78,21	84,29	81,67	68,95
LightGBM	79,39	75,21	77,54	80,11	66,46
Random Forest	84,08	81,57	84,67	84,55	66,30
SGD	78,70	76,58	81,05	77,95	71,23

En términos de la métrica Sensibilidad, en *E. coli* K-12 la media global fue de 89,41% (SD = 1,75). Para *S. enterica* serovar Typhi CT18 el valor global de Sensibilidad fue de 80,35% (SD = 19,34), en *S. aureus* Newman fue de 83 % (SD = 8,75), para *A. baumannii* AYE, el valor medio fue de 91,46% (SD = 0,13). Por último, en el caso de *P. aeruginosa* PAO1, el resultado de Sensibilidad fue del 100% (Tabla 16). No obstante, esto se debe a que el genoma de *P. aeruginosa* PAO1 solo contenía 6 secuencias de inserción, todos los clasificadores lograron "identificar" esas secuencias. No obstante, el nivel de falsos positivos fue considerable y no se puede aseverar un buen rendimiento sobre este genoma en particular *P. aeruginosa* presenta un genoma con una plasticidad y alta heterogeneidad debida a múltiples eventos de transferencia horizontal genética y recombinación homologa. El elevado número de falsos positivos puede ser debida al gran número de secuencias repetitivas cortas o fragmentos de ISs en el genoma de esta especie bacteriana.

A nivel de cada clasificador, en el genoma de E. *coli* K-12 los valores más altos obtenidos de Sensibilidad fueron por los algoritmos *Logistic Regression* y XGBoost, 91,54% y 90,14%, respectivamente. En el genoma *S. enterica* serovar Typhi CT18 los valores más elevados fueron los alcanzados por los clasificadores LightGBM y XGBoost, 98,21% y 96,42%, respectivamente. En *S. aureus* Newman los clasificadores con valor más altos fueron LightGBM y SVM, ambos con 90% para la de Sensibilidad. Por último, para el caso de *A. baumannii* AYE, el valor medio fue 91.46% y todos los clasificadores lograron valores por encima del 91%.

Tabla 16. Resultados obtenidos por cada clasificador en términos de Sensibilidad en los cinco genomas de referencia.

	Sensibilidad (%)				
Algoritmo	*E. coli*	*S. enterica*	*S. aureus*	*A. baumannii*	*P. aeruginosa*
Logistic Regression	91,54	53,57	85	91,50	100
XGBoost	90,14	96,42	80	91,52	100
SVM	90,12	94,64	90	90,21	100
LightGBM	90,10	98,21	90	91,52	100
Random Forest	85,91	85,71	65	91,52	100
SGD	89,01	53,55	88	91,15	100

La media global de la Especificidad computada sobre los diferentes genomas fue de 82,57% (SD = 3,71) para *E. coli* K-12, 73,92% (SD = 9,31) para *S. enterica* serovar Typhi CT18, 81,67% (SD = 3,45), para *S. aureus* Newman. En *A. baumannii* AYE la Especificidad media fue de 80,57% (SD = 2,47), y en *P. aeruginosa* PAO1, el rendimiento global fue más bajo que en los otros genomas de referencia, alcanzando una media de 68.05% (SD = 1,91).

Los algoritmos de clasificación con mejor rendimiento sobre el genoma *E. coli* K-12 fueron SGD y *Random Forest* con valores de 84,05% y 90,14%, respectivamente. En *S. enterica* serovar Typhi CT18 los algoritmos *Random forest* y XGBoost obtuvieron los mejores resultados, 81,52% y 78,65%, respectivamente. En el caso de *S. aureus* Newman, Random *Forest* y SVM mostraron los mayores valores, alcanzando 84,82% y 84,24%, respectivamente. En *A. baumannii* AYE, *Random Forest* mostró un valor de Especificidad de 84,44% y XGBoost 82,33%, siendo ambos clasificadores los que mostraron mejor desempeño. Por último, en *P. aeruginosa* PAO1, el algoritmo SGD fue único algoritmo que alcanzó un valor sobre 70%. (Tabla 17). Cabe notar que, como fue descrito en la sección de Sensitividad, este genoma fue particular dado a la escasa presencia de IS a detectar por los clasificadores.

Tabla 17. Especificidad obtenida por cada clasificador.

	Especificidad (%)				
Algoritmo	*E. coli*	*S. enterica*	*S. aureus*	*A. baumannii*	*P. aeruginosa*
Logistic Regression	80,23	76,87	82,76	77,54	68,60
XGBoost	82,63	78,65	82,78	82,33	66,24
SVM	81,58	78,02	84,24	81,51	68,92
LightGBM	79,20	74,94	74,44	79,92	66,42
Random	84,05	81,52	84,82	84,44	66,30

Forest					
SGD	90,14	53,57	80,98	77,73	71

En cuanto a la métrica F1, que mide la precisión de los algoritmos utilizados, contemplando la Sensibilidad y la proporción de falsos positivos, el algoritmo SVM mostró los valores más altos, mientras que el algoritmo SGD no superó el umbral de 87,5%. Se pueden observar valores similares para los diversos algoritmos para la especie *P. aeruginosa* (Tabla 18). Llamativamente, el despeño de los diferentes algoritmos considerando la métrica F1, fue similar, exceptuando la especie bacteriana *P. aeruginosa*.

Tabla 18. Valores de la medida F1-score sobre genomas de referencia.

	F1				
Algoritmo	*E. coli*	*S. enterica*	*S. aureus*	*A. baumannii*	*P. aeruginosa*
Logistic Regression	87,59	85,86	90,02	87,09	79,05
XGBoost	86,69	87,08	89,66	88,72	78,82
SVM	88,78	86,93	92,05	89,58	81,50
LightGBM	85,63	83,82	85,20	86,22	78,13
Random Forest	88,89	87,42	90,07	89,59	78,32
SGD	84,05	82,80	87,13	82,90	78,36

En relación con el tiempo requerido en las ejecuciones, el algoritmo que más demoró para realizar las clasificaciones sobre los cinco genomas de validación fue SVM, con 93,6 segundos (Figura

12). Por otro lado, el algoritmo LightGBM fue el menos costoso, requiriendo 5,42 segundos en realizar las clasificaciones. El algoritmo Random Forest demandó 11 segundos, en tanto que *Logistic Regression* y XGBoost tomaron 7,88 y 7,82 segundos.

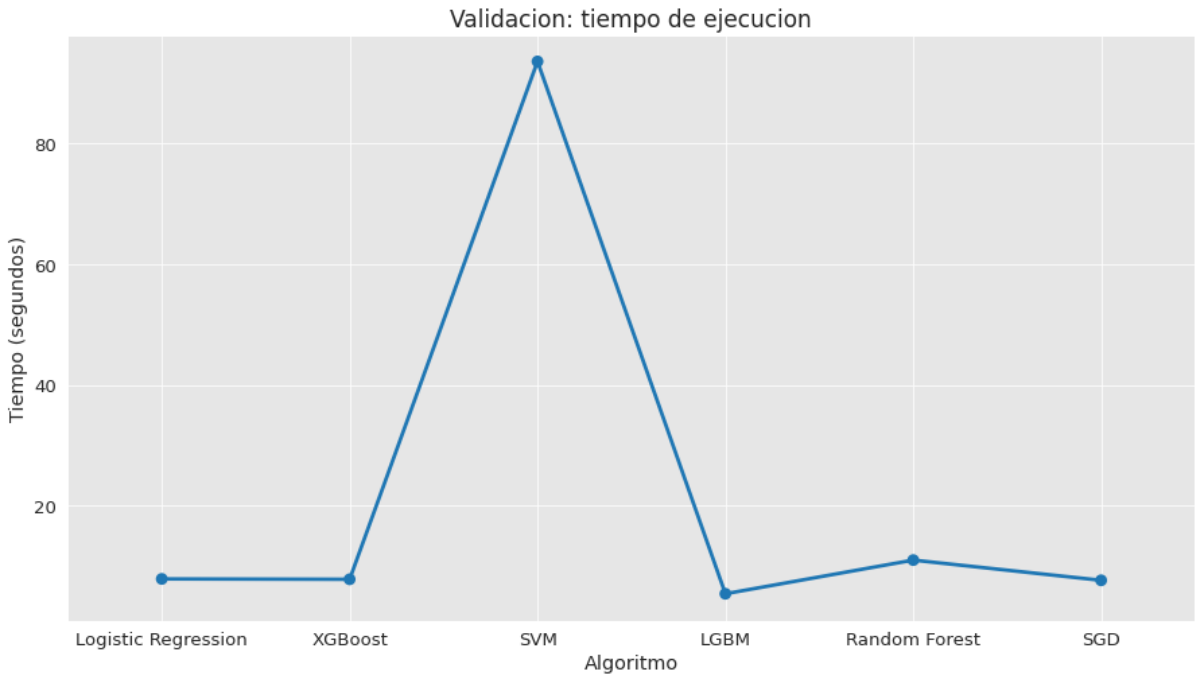

Figura 12. Tiempo de cómputo requerido por cada algoritmo de clasificación en la etapa de validación.

Selección del algoritmo de clasificación

Los criterios a tener en cuenta para la selección de algoritmo fueron dos: rendimiento en clasificación, es decir los valores obtenidos para las métricas de evaluación utilizadas y el tiempo de ejecución requerido. El primer criterio fue considerado tanto en la etapa de preparación del algoritmo, es decir sobre el conjunto de datos de IS generado a partir de la información de la base ISfinder, como en la etapa de validación, sobre los cinco genomas de referencia. Como fue detallado en la sección Materiales y Métodos, los conjuntos de datos fueron diferentes entre una y otra etapa en términos de la distribución de la variable dependiente (IS). En la fase de

preparación del algoritmo, la distribución de la variable dependiente fue balanceada, en tanto que en la etapa de validación hubo sesgo marcado de 0,1% hacia la clase positiva (valor 1)

Así, en función de los valores observados en las métricas de evaluación en clasificación en estas etapas: el algoritmo que se seleccionó fue XGBoost. Dicho clasificador, alcanzó valores para la métrica AUC (inglés: *Area under the curve,* español: Área bajo la curva) de 98,55 % (Figura 13). Además, se puede observar la distribución de probabilidades, notándose una elevada frecuencia de las probabilidades discretizadas entre 0 y 0,2, y entre 0,8 y 1 (Figura 14). La distribución de las probabilidades observadas nos estaría reforzando el buen rendimiento del clasificador seleccionado para su implementación en la predicción de ISs.

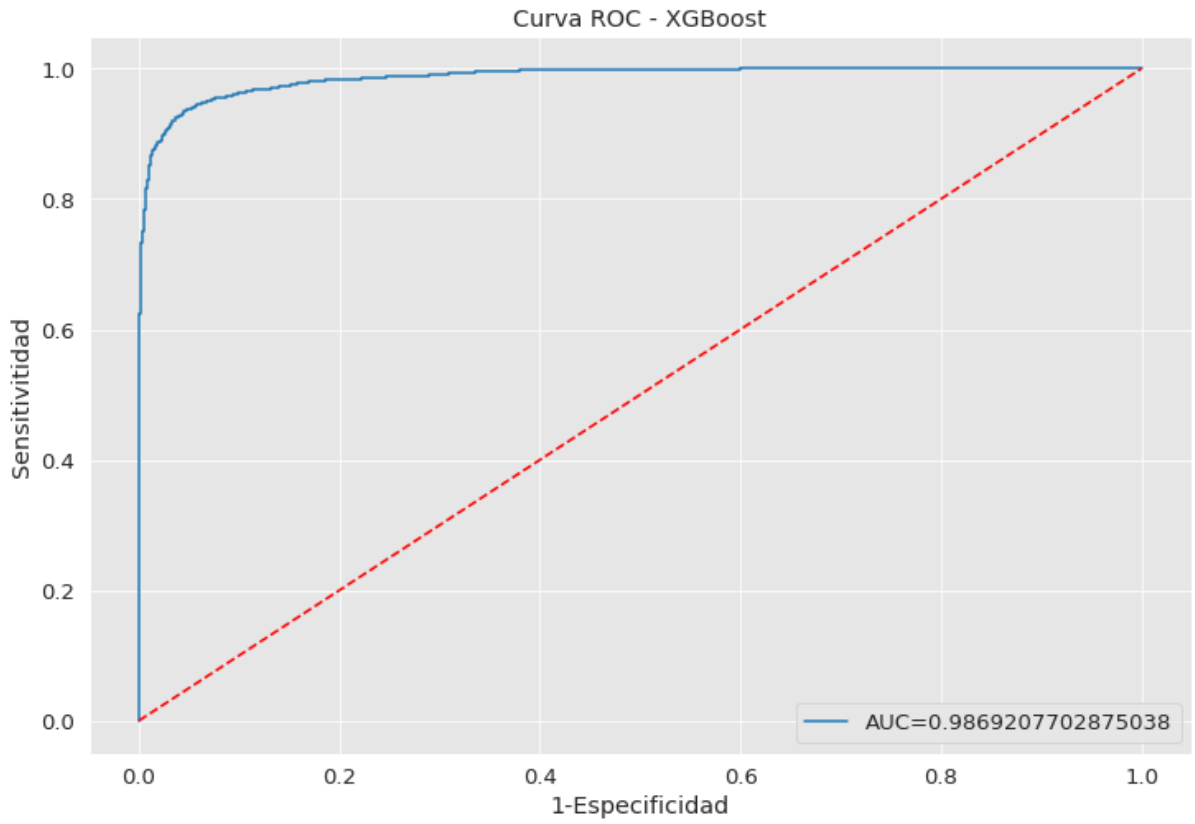

Figura 13. Curva ROC generada por el algoritmo XGBoost durante la etapa de training y testing.

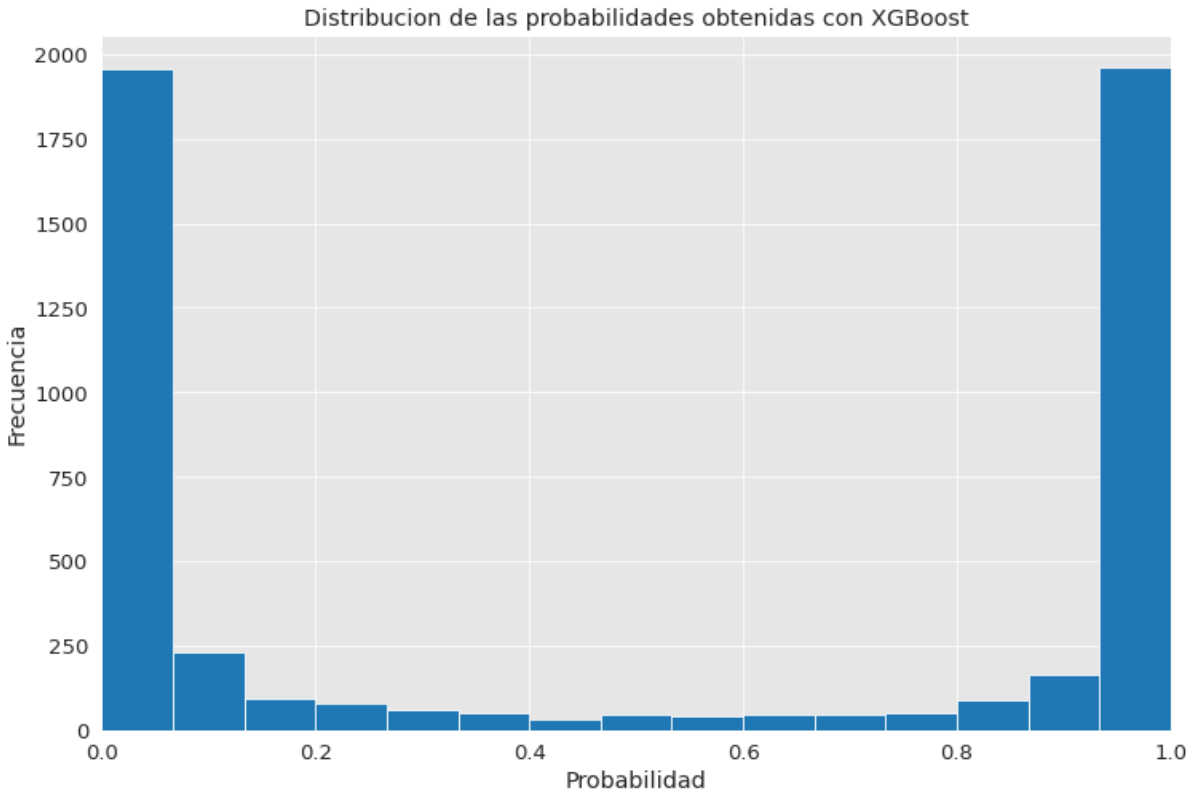

Figura 14. Histograma de distribución de probabilidades generadas en la predicción por el algoritmo XGBoost.

En términos de la métrica Especificidad, el valor medio obtenido es 78,53. Como se detalló anteriormente, dicha métrica es penalizada por el número de Falsos Positivos obtenidos en una clasificación, en este caso, XGBoost mostró valores similares entre las diferentes especies bacterianas analizadas (Tabla 16). Otro punto por considerar fue la consistencia obtenida por el algoritmo sobre los Verdadero Positivos. En relación con la métrica F1, XGBoost mostró valores promedio de 89%, similares al resto de los algoritmos utilizados.

Por último, como fue detallado previamente, el tiempo de cómputo requerido fue de 7,8 segundos en condiciones locales. El parámetro tiempo de proceso en condiciones locales, resulto clave en la selección del algoritmo debido a que el programa tiene el objetivo de ser una herramienta para la predicción de ISs sea implementado sobre un gran número de genomas bacterianos de diferentes especies, con características muy variables.

Etapa de ajuste

Con el fin de mitigar el sobreajuste u *overfitting*, es decir, la elevada varianza generada y su débil consistencia al momento de re-entrenarse sobre diferentes datos (por ejemplo, mediante *cross-validation*), se decidió incluir el parámetro: “early stopping”. Dicho parámetro surge de un abordaje denominado regularización, y la idea subyacente es incluir un valor en el cual se controla o monitorea el rendimiento del algoritmo XGBoost durante la etapa de entrenamiento. El objetivo de la regularización es prevenir el efecto de sobreajuste sobre los datos de entrenamiento.

En particular *early stopping* controla el rendimiento del modelo durante la etapa de entrenamiento (en datos de la etapa de entrenamiento o *Training*), y tiene la facultad de interrumpir dicho proceso en un punto en el cual el rendimiento (en términos de la métrica seleccionada) ha alcanzado un punto máximo y a partir del cual no se logra superar un valor determinado para dicha métrica, es decir, cuando pasado cierto número de iteraciones no se logra superar el valor obtenido. En el presente trabajo, las métricas de evaluación implementadas fueron *log-loss* (logaritmo de *loss*), una función de costo cuyo objetivo es optimizar el entrenamiento del modelo, y minimizar error en clasificación. El algoritmo XGBoost, en el cual los valores óptimos para sus parámetros fueron obtenidos mediante el uso de la estrategia de búsqueda randomizada (GridSearch), realizó la clasificación por defecto en 181 iteraciones. Luego de la inclusión del parámetro de regularización “early stopping” se logró reducir el número de iteración a 145, es decir, se redujo un 36% el número de iteraciones (Figura 15 y Figura 16).

Figura 15. Progresión de los valores de la métrica Log loss generados por el estimador XGBoost a lo largo de las epochs incluidas.

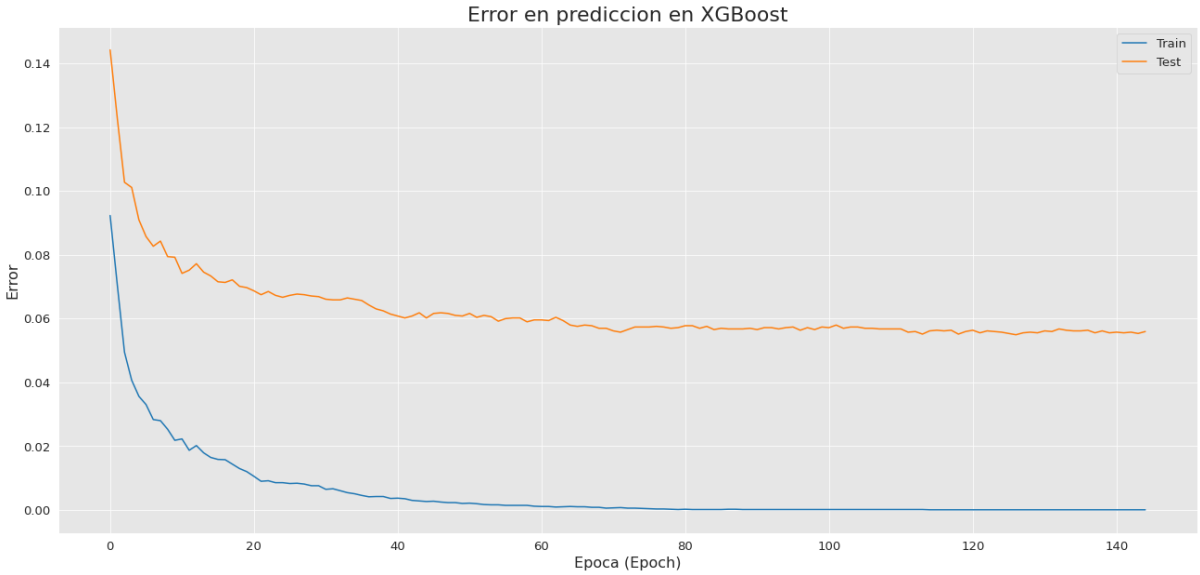

Figura 16. Métrica Error de clasificación del estimador XGBoost a lo largo de las diferentes epochs.

Calibración

Con el fin de observar y poder hacer uso de las probabilidades generadas por el algoritmo XGBoost mediante el método “predict_proba”, se llevó a cabo la calibración de tipo isotónica. Para poder cuantificar los resultados del algoritmo pre y post-calibración, se efectuaron rondas de clasificación sobre los genomas de referencia y se compararon los resultados. No se observaron grandes diferencias entre el modelo con y sin calibración. Dichos resultados reflejan la robustez del algoritmo, no siendo necesario la calibración del mismo.

Tabla 19. Comparación del rendimiento de los algoritmos clasificadores previo y posterior al proceso de calibrado. Las métricas obtenidas, son sobre los genomas de referencia.

	Calibración	**F1 (%)**	**VP**	**FP**	**Sensitividad (%)**	**Especificidad (%)**	**Accuracy (%)**
E. coli	Pre	88,69	64	755	90,14	81,89	82,36
	Post	89,30	64	713	90,14	82,90	83,02
S. enterica	Pre	87,08	54	1007	96,42	78,65	78,86
	Post	87,12	54	1004	96,42	78,67	78,90
S. aureus	Pre	89,66	455	14	80	82,78	82,76
	Post	89,29	473	16	80,22	81,82	81,81
A. baumannii	Pre	88,72	55	665	91,52	82,33	82,48
	Post	88,70	54	665	90,12	82,30	82,33
P. aeruginosa	Pre	78,82	6	1938	100	66,24	66,28
	Post	80,03	6	1845	100	66,98	66,87

Análisis sobre potenciales falsos positivos en genomas de referencia

Estudio de falsos positivos

El estudio de falsos positivos se realizó sobre los cinco genomas de referencia con el algoritmo XGBoost, seleccionado por su velocidad y eficiencia. En una prueba de tipo binaria, cómo es el caso del sistema desarrollado, se denomina Falso Positivo (FP) a la secuencia que fue predicha como IS pero, según su anotación no pertenecía a una IS. En términos de prueba de hipótesis, un FP puede también ser interpretado como un error de tipo I. Es importante señalar que la anotación genómica es un proceso operador-dependiente, que puede ser llevada a cabo mediante diversos softwares y bajo diferentes parámetros. Teniendo en cuenta lo descrito, la anotación de determinadas secuencias puede no ser exacta, con lo cual también dichas secuencias pueden ser consideradas como potenciales IS. Como se detalla en la sección Materiales y Métodos, en el apartado de análisis sobre Falsos Positivos, se implementará el alineamiento de secuencias mediante BLAST para discernir entre las secuencias que fueron realmente FP y aquellas que no fueron correctamente anotadas.

El genoma que más incidencia tuvo de FP fue el de *P. aeruginosa.* En dicho genoma, el estimador clasificó con una probabilidad $\geq$ a 0.50 a 1895 secuencias. En el genoma de *S. enterica* se encontraron 923 FP. En los genomas de las especies *E. coli* y *A. baumanii*, 742 y 622 secuencias fueron catalogadas como potenciales IS o FP, respectivamente. Por último, en el genoma de la especie *S. aureus*, fueron identificadas 436 potenciales secuencias de inserción (Figura 17).

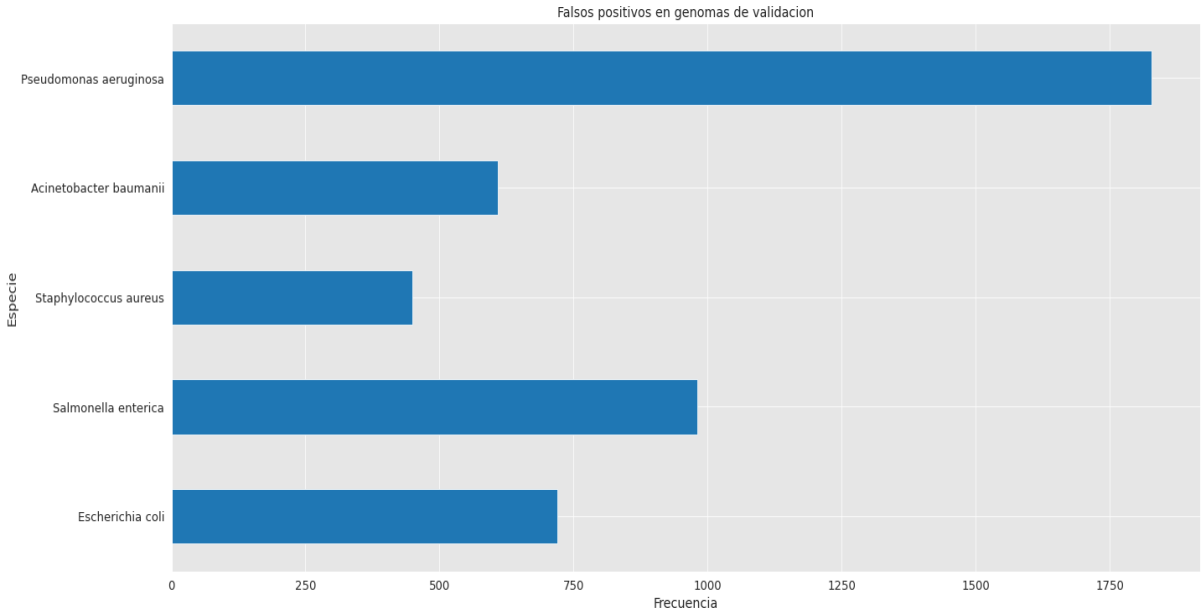

Figura 17. Distribución de secuencias consideradas como falsos positivos luego de la clasificación con el estimador XGBoost.

Análisis de secuencia sobre falsos positivos

La ejecución de los alineamientos se llevó a cabo en un módulo desarrollado como parte del software propuesto. En el mismo, se trabajó con la librería BioPython (Cock y cols., 2009) que permite ejecutar el algoritmo BLAST dentro del entorno del lenguaje Python.

El análisis de secuencia se realizó considerando las secuencias aminoacídicas pertenecientes a cada genoma. De este modo, para el genoma de la especie *P. aeruginosa*, el análisis de secuencia confirmó un total de 73 secuencias de inserción, lo cual representó un 3.9% de las secuencias previamente categorizadas como FP según la anotación. Sobre el genoma de la especie *S. enterica* luego del alineamiento se obtuvieron 36 IS sobre las 923 FP, representando un 3.9% de los FP generados luego de la clasificación. En el caso de la especie *E. coli*, se lograron detectar 25 nuevas IS sobre un total de 742, representando un 3.36% de los FP. En *A. baumannii*, se encontraron luego de la búsqueda por similitud 20 IS sobre 622 secuencias previamente categorizadas en base a su anotación como FP, lo que representa un 3.21%. Por último, en la especie *S. aureus* previamente se habían categorizado 436 secuencias como FP, no obstante, se encontraron 6 mediante búsqueda por similitud, lo que representa un 1.37%.

Cabe destacar, que los FP presentaron una identidad aminoacídica menor al 95%, por lo que se podrían considerar posibles nuevas ISs. Para confirmar dicha funcionalidad, se deberían hacer ensayos experimentales.

Los resultados de las secuencias de inserción halladas en cada especie y sus respectivas frecuencias se encuentran detalladas en la Tabla 20.

Tabla 20. Frecuencias de secuencias de inserción identificadas en cada especie.

Genoma	Familia	Frecuencia	%
P. aeruginosa	IS3	28	37.83
	IS1595	13	17.56
	ISL3	7	9.45
	IS66	4	5.40
	ISNCY	4	5.40
	ISKra4	3	4.05
	IS481	3	4.05
	IS200/IS605	2	2.70
	IS1	2	2.70
	IS91	2	2.70
	IS5	2	2.70
	IS1182	1	1.35
	IS630	1	1.35
	IS6	1	1.35
	IS4	1	1.35
S. enterica	IS3	10	26.31
	ISNCY	6	15.78

	IS1595	5	13.15
	ISL3	4	10.52
	IS200/IS605	4	10.52
	IS66	3	7.89
	ISKra4	3	7.89
	IS91	2	5.26
	IS1	1	2.63
E. coli	IS3	8	32.0
	ISKra4	4	16.0
	IS1595	4	16.0
	ISNCY	3	12.0
	IS200/IS605	2	8.0
	ISL3	1	4.0
	IS1	1	4.0
	IS481	1	4.0
	IS91	1	4.0
A. baumannii	ISL3	5	25.0
	IS91	4	20.0
	IS3	3	15.0
	IS200/IS605	2	10.0
	IS1	1	5.0
	ISKra4	1	5.0
	IS481	1	5.0
	ISNCY	1	5.0

	IS1595	1	5.0
S. aureus	IS91	2	33.33
	IS30	1	16.66
	IS3	1	16.66
	ISNCY	1	16.66
	ISKra4	1	16.66

Otro aspecto que se tuvo en cuenta en el presente análisis fue la frecuencia de las familias a lo largo de todos los genomas de validación. Es decir, se hizo foco sobre las familias de IS presentes en todas las especies. El análisis sobre las familias de IS presentes en todas las especies se realiza debido a que son IS que pudieron saltar de una especie bacteriana a otra, es decir, presentan un amplio rango de huésped, mientras que la familia de ISs únicas fueron excluidas debido a que probablemente estén asociadas intrínsecamente a la especie.

Las cuatro familias que se observaron compartidas en los cinco genomas de validación fueron IS3, ISKra4, ISNCY, e IS91. Dentro de las cuales, la frecuencia mostró cierta variabilidad a lo largo de las especies. La frecuencia relativa de la familia IS3 fue significativamente mayor en la la especie *P. aeruginosa*. La familia ISKra4, mostró una frecuencia relativa mayor en la especie *E. coli*, en tanto que la familia ISNYC se mostró con mayor frecuencia en la especie *S. enterica*. Por último, en la especie *A. baumannii* la familia IS91 mostró una mayor frecuencia. (Tablas 21 y 22).

Tabla 21. Frecuencias absolutas de las familias de secuencias de inserción sobre los genomas de validación.

	P. aeruginosa	***S. enterica***	***E coli***	***A. baumannii***	***S. aureus***
IS3	28	10	8	3	1
ISKra4	3	3	4	1	1
ISNYC	4	6	3	1	1
IS91	2	2	1	4	2

Tabla 22. Frecuencias relativas de las familias de las secuencias de inserción sobre los genomas de las especies incluidas en la etapa de validación.

	P. aeruginosa	***S. enterica***	***E. coli***	***A. baumannii***	***S. aureus***
IS3	0.56	0.20	0.16	0.06	0.02
ISKra4	0.25	0.25	0.33	0.08	0.08
ISNYC	0.26	0.40	0.20	0.06	0.06
IS91	0.18	0.18	0.09	0.36	0.18

Escalado y despliegue

Para reforzar el potencial de la herramienta generada, se decidió probar el algoritmo de clasificación seleccionado sobre un volumen amplio de genomas correspondiente a una especie determinada. Se desarrolló un software compuesto de módulos específicos con el fin de automatizar las tareas de descarga de genomas, clasificación e identificación de potenciales IS sobre cada genoma, generación de reporte (en términos de clasificación) y validación de resultados mediante búsqueda por similitud utilizando la herramienta BLAST.

De este modo, en función de los módulos diseñados, la arquitectura del programa desarrollado quedó definida como:

- Descarga automatizada de genomas.
- Procesamiento de genomas
- Clasificación y reporte
- Alineamiento local de genomas (BLAST)

Cabe notar que en la etapa de validación se incluyó una búsqueda por similitud sobre los falsos positivos. Como falso positivo (FP) se definió como aquella secuencia aminoacídica que fue clasificada por el algoritmo como IS, pero en función de su anotación no pertenecía a una secuencia de inserción. No obstante, la anotación tiene un sesgo inherente a la metodología empleada, y la razón es la falta de estandarización y reproducibilidad. Por este motivo, fue necesario incluir un análisis sobre estas secuencias FP en particular

Descarga automatizada de genomas

Para la obtención de la información de los genomas se procedió a la descarga, mediante el protocolo FTP sobre el servidor de GenBank, de las secuencias codificantes, el genoma completo, proteínas anotadas, etc. Esta tarea se realiza en forma masiva y automatizada en un módulo específico.

Así, en primera instancia, se realizó la búsqueda en una especie bacteriana de interés en la base de datos de genomas del GenBank (https://www.ncbi.nlm.nih.gov/genome/). Posteriormente, se seleccionó la especie que se desea analizar, en el caso del presente trabajo, el estudio se focalizó en el genoma de *E. coli*. Esta especie fue seleccionada por múltiples motivos. En primer lugar, *E. coli* es un organismo modelo que ha sido estudiado desde distintos puntos de vista desde hace más de un siglo (Escherich 1895). Su anotación es confiable y basada en información experimental abundante. En segundo lugar, y ligado a lo anterior, existe una basta cantidad de genomas disponibles en repositorios y bases de datos públicas. Siendo la especie bacteriana con más genomas disponibles a la fecha.

Es importante notar que los genomas que se tuvieron en cuenta en esta etapa fueron completos o también llamados cerrados, es decir con un nivel de ensamblaje ("assembly level") entero. En estos genomas se tiene la información de todas bases de la o las moléculas que componen el genoma del organismo (es decir de cada cepa, en este caso). Luego de seleccionar este requerimiento, se descargó un archivo denominado "prokaryotes" de extensión CSV. Sobre dicho archivo se seleccionaron las columnas *Strain* y "GenBankFTP". *Strain* es un identificador de la secuencia localizada en el repositorio y hace referencia a un nombre propio en tanto que *GenBankFTP* contiene la ruta FTP hacia donde se alojan los diversos archivos (comprimidos en formato .gz) que contienen al genoma.

Para un mejor entendimiento, un ejemplo del esquema del archivo se detalla en la Tabla 23. Por lo tanto, el *scrapper* desarrollado utiliza el archivo "prokaryotes.csv", luego itera sobre cada registro de la columna *GenBankFTP*, lee el registro FTP y procede a la descarga del archivo con extensión.faa.gz. La columna *Strain*, es utilizada por el programa para generar un directorio específico con el nombre del registro, donde finalmente se descargará el genoma correspondiente a dicha cepa.

Tabla 23. Ejemplo del esquema que contiene los datos para cada gen.

Strain	**GenBankFTP**
K-12 substr. MG1655	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/005/845/GCA_000005845.2_ASM584v2

Una vez descargados todas las secuencias proteicas de ese genoma, el programa descomprime el archivo, crea un directorio con el nombre de la cepa y dentro de ese directorio convierte los archivos desde formato FASTA a una matriz de tipo *dataframe* el cual contiene, la descripción de la secuencia, anotación, la secuencia aminoacídica. Todos los campos en formato *string*.

Procesamiento de secuencias y clasificación

Luego de generado el *dataframe* correspondiente a cada genoma como resultado del paso previo y tomando como referencia la anotación de la secuencia, se determinó la variable dependiente (Secuencia de Inserción). De este modo el módulo asignó el valor de 1, si la anotación corresponde a una secuencia de inserción (IS) y 0, si no lo es (non-IS). La variable *target*, fue de importancia relevante ya que se utilizó para determinar las métricas de rendimiento Especificidad, Sensibilidad y A*ccuracy*.

El clasificador XGBoost fue el algoritmo que mostró resultados consistentes a sobre las diferentes especies en testing y validación y por lo tanto, fue implementado en la etapa de despliegue. Así, XGBoost recorre de una manera iterativa cada directorio correspondiente a cada cepa descargada, generando un reporte (Figura 18) específico de ese genoma, explicitando las siguientes métricas:

- Matriz de confusión
- Accuracy
- Sensitividad
- Especificidad
- Potenciales falsos positivos
- Falsos negativos

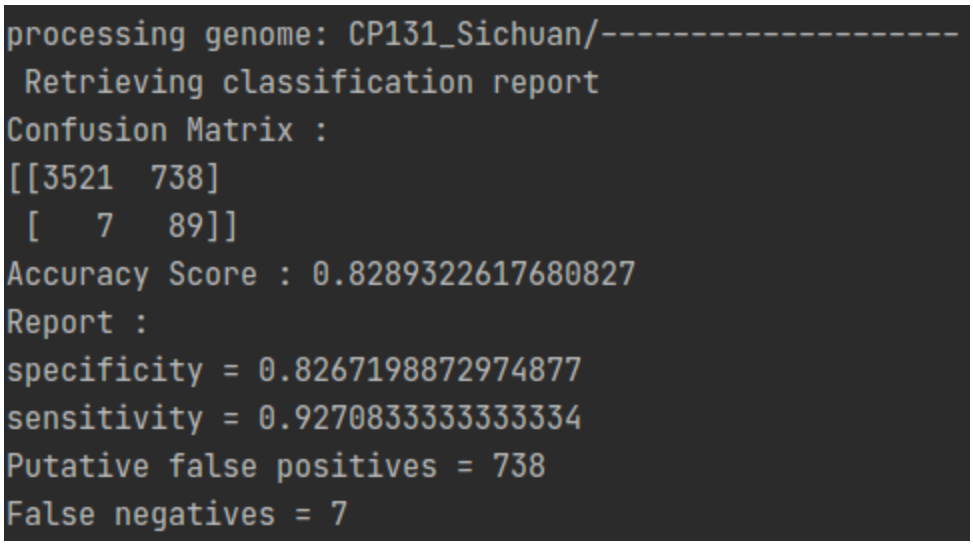

Figura 18. Reporte específico para cada cepa/genoma, de *E. coli*. Se detalla el nombre de la cepa (*Strain*), matriz de confusión y métricas de evaluación de rendimiento. En este ejemplo, se detallan los resultados para el genoma CP131_Sichuan.

Reporte global

En total se analizaron 1196 genomas completos de *E. coli*, el tiempo tomado por el algoritmo de clasificación fue de aproximadamente 30 minutos a nivel local utilizando la maquina completa (Intel(R) Core (TM) i7-6700HQ CPU, 2.60GHz, 16gb RAM), aproximadamente el tiempo requerido por genoma fue de 0.66 segundos. Las medidas globales para el conjunto de cepas analizadas fueron: mediana de *Accuracy* = 82,11%, mediana de Especificidad = 82,04% y mediana de Sensibilidad = 87,50% (Figura 19)

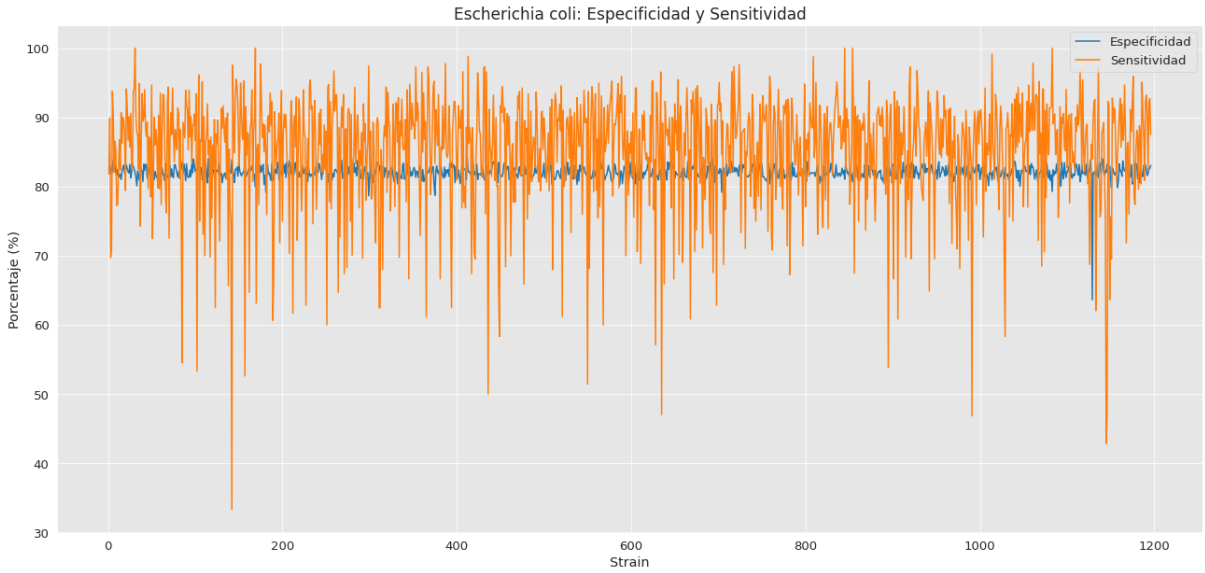

Figura 19. Métricas de Especificidad y Sensibilidad expresada en porcentaje sobre los 1196 genomas completos de *E. coli* donde se ejecutó el clasificador XGBoost.

En total se identificaron 73154 secuencias de inserción. Es decir, catalogadas como verdaderos positivos (VP), lo que significa una frecuencia de 6.7% sobre el total de secuencias codificantes para proteínas. En términos de la probabilidad asignada por XGBoost a dichas secuencias, la media fue de 95,85% (SD = 10,08) y una la mediana de 99,71%. En términos de la distribución de familias de IS presentes, las familias más frecuentes fueron IS1 con 214 secuencias, IS66 con 152 secuencias, IS629 con 128 secuencias y por último IS621, con 101 secuencias identificadas en la familia (Figura 20).

Por otro lado, 1.024.386 secuencias, con una probabilidad media asignada de 78,93% (SD = 15,53) y una mediana de 80,81%, fueron determinadas como potenciales falsos positivos (FP). Los detalles de todas las medidas de tendencia central incorporadas en el análisis se muestran en la Tabla 24.

Tabla 24. Medidas de tendencia central en la probabilidad sobre las secuencias marcadas como falsos positivos (FP) y verdaderos positivos (VP) a lo largo de los genomas analizados en la validación.

Especie		Medidas			
		Frecuencia	**Mediana**	**Media**	**Desvío Standard**
E. coli	VP	73.154	99,7	95,8	10,0
	FP	1.024.386	80,8	78,9	15,5

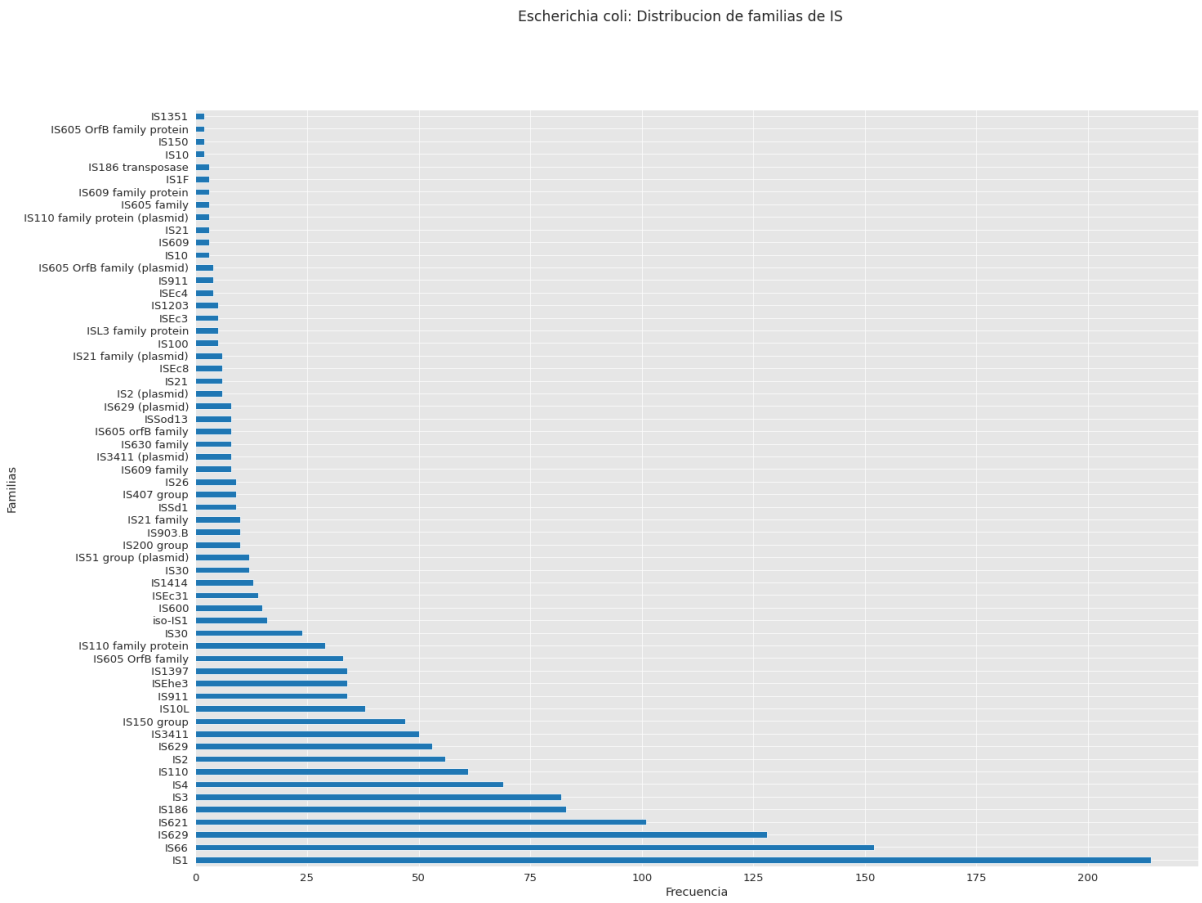

Figura 20. Distribución de las familias de IS detectadas como VP en los genomas de la especie *E. coli* analizados en la validación.

Identificación de IS sobre secuencias catalogadas como falsos positivos

Al igual que en el análisis de FP sobre genomas de validación, en esta etapa se procedió a recolectar los FP producidos luego de la clasificación con el algoritmo XGBoost y en base a la anotación por defecto.

El número de FP fue de 1.024.386 con una media para la probabilidad de 78,93% (SD =15,53), en tanto que la mediana fue de 80,81%

El análisis de secuencia mediante el algoritmo BLAST requirió once horas, treinta y siete minutos (11:37) en ordenador local (Intel(R) Core (TM) i7-6700HQ CPU, 2.60GHz, 16gb RAM). Cabe notar que dicho análisis se llevó a cabo sobre todas las secuencias catalogadas como FP. Como

resultado se hallaron 41.128 IS nuevas. Esto representó un 4,01% sobre el total de FP identificados inicialmente por el estimador.

En términos de la distribución de familias halladas, la IS3 tuvo una frecuencia relativa mayor al resto de las familias, representando un 28,75% (12322 secuencias) del total. La familia IS1595 estaba representada por 5,937 secuencias, lo cual significó un 13,85% del total y la familia ISKra representó 4.219 siendo un 9,84% del total. (Figura 21).

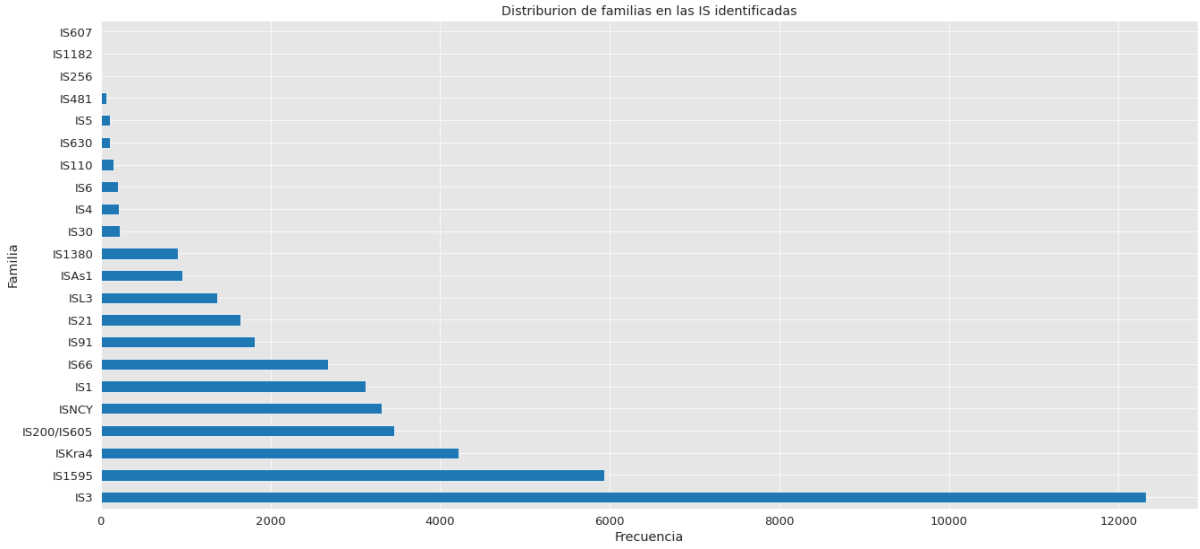

Figura 21. Distribución de familias de IS sobre el genoma a gran escala de la especie *E. coli.*

Dado que, en la etapa de validación, también se trabajó con el genoma de referencia de *E. coli* K-12, se realizó una comparación de los resultados obtenidos, entre las etapas de validación y escalado. De este modo, fue posible comparar las métricas de rendimiento en ambos casos. No se observaron grandes diferencias, para las métricas de Especificidad, Sensibilidad y *Accuracy*, lo cual sugiere una distribución similar de falsos positivos y verdaderos positivos entre el genoma utilizado en validación y los genomas en el escalado (Tabla 25), sustentando la consistencia del software desarrollado tanto en el procesamiento de las secuencias de entrada como en la clasificación de dichos datos.

Tabla 25. Comparación de resultados obtenidos en clasificación, entre las etapas de validación y escalado sobre el mismo genoma la especia *E. coli*.

Genoma	Etapa	Especificidad (%)	Sensibilidad (%)	Accuracy
E. coli K-12	Validación	82,63	90,14	80,42
E. coli	Escalado	82,07	85,15	82,14

Repositorio

El código de la aplicación desarrollada se encuentra alojado en GitHub, y se puede acceder mediante el siguiente vinculo https://github.com/Mangel1782/IS_detector_app/

Discusión

Por definición, las secuencias de inserción (IS) son un tipo de elementos transponibles (TE) que tienen la capacidad de desplazarse de un segmento de ADN a otro a través de reacciones de escisión e integración que son independientes de la recombinación homóloga. Pueden movilizarse desde el cromosoma a un plásmido o viceversa, o dentro de la misma molécula de ADN. Los transposones han sido encontrados en todos los géneros bacterianos en los cuales han sido buscados. Estos elementos juegan un rol central en la evolución proporcionando mecanismos para la generación de variabilidad, y, en conjunción con los sistemas de transferencia de ADN, participan de la rápida diseminación de material genético entre bacterias (Hall y cols., 1989, Syvanen y cols., 1984, Grinsted y cols., 1990).

Hasta el momento, los trabajos publicados que se han enfocado en la detección e identificación de IS lo han hecho mediante algoritmos de alineamiento de secuencias. Específicamente, implementando el algoritmo BLAST, el cual, como ha sido descrito en secciones previas, es hasta ahora el método *gold standard* para realizar búsquedas por similitud. A grandes rasgos, a partir de una secuencia incógnita de partida o *query*, BLAST hará una búsqueda de secuencias similares referenciando una base de datos predefinida por el usuario. Luego de ejecutarse dicha comparación mediante alineamiento local, el programa genera un reporte que especifica los hits, es decir, las secuencias halladas *subject* en la base de datos y los alineamientos a la secuencia *query*. Trabajos como el de Ochiai (Ochiai y cols., 2005) se han focalizado en la identificación de IS que están presentes en la bacteria *Xanthomonas oryzae*. Dicho patógeno es responsable de una de las enfermedades más importantes en el arroz. La identificación de IS tuvo lugar mediante el uso de BLAST, y como secuencias de referencia se utilizaron las reportadas en la base ISFinder (Siguier y cols., 2016). Trabajos más recientes como el Huda (Huda y cols., 2014) han reportado la identificación de IS en la especie *P. aeruginosa*, también mediante el uso de un flujo de trabajo o *pipeline* que incluye; la búsqueda de secuencias mediante BLAST y la comparación de los resultados generados con ISFinder.

Si bien hasta el momento no se han publicado trabajos implementando abordajes mediante aprendizaje automatizado (*machine learning*) para identificar y predecir IS bacterianos, existe literatura disponible donde se describe el empleo de este tipo de algoritmos para clasificar otros tipos de elementos transponibles (TE). Nakano y colaboradores (Nakano y cols., 2018) han

reportado el uso de algoritmos de aprendizaje supervisado con el objetivo de clasificar los TE. Cabe mencionar que esta identificación es de tipo jerárquica, es decir, basándose en la taxonomía sobre los TE propuesta por Wicker y cols. (2007), la cual se fundamenta en características filogenéticas y estructurales de dichos elementos genómicos. Del mismo modo, más recientemente, Panta y cols., (2019) han implementado un algoritmo SVM con el mismo objetivo.

Uno de los principales desafíos que se pudo resolver en el presente trabajo fue alcanzar una adecuada representación de las secuencias aminoacídicas. Esta etapa resulta crítica, dado a que el objetivo es extraer atributos de las secuencias que luego serán interpretadas por el algoritmo de clasificación como variables independientes.

Un clasificador de aprendizaje automatizado puede interpretarse como una función cuyo objetivo es mapear un grupo de variables de entrada (por convención, representado X) correspondientes a una instancia en el conjunto de datos, interpretar su distribución y asignar dicha instancia a una de las clases o categorías, previamente definidas en la etapa de entrenamiento, y que corresponden a la variable dependiente o clase.

En el problema abordado en el presente trabajo, las instancias estuvieron representadas por los atributos extraídos de las secuencias proteicas provenientes de la base de datos ISFinder y PDB. La variable dependiente o clase, de tipo binaria, estuvo compuesta por los niveles IS o Non-IS representando a una secuencia de inserción y a una secuencia diferente a una secuencia de inserción, respectivamente.

Los trabajos que han implementado algoritmos de aprendizaje automatizado sobre secuencias aminoacídicas han hecho uso de técnicas “simples”, con el fin de extraer atributos de las secuencias, como por ejemplo el uso de k-meros (Panta y cols., 2019), basándose en atributos específicos como dominios funcionales conservados (Steinbiss y cols., 2009), o bien mediante heurísticas, como la implementación de sistemas de codificación de aminoácidos (Wu 1992), e incluso abordajes más elementales como la longitud de secuencias u otras características estructurales (Hoede y cols 2014). A esto, además, hay que sumarle el elevado costo computacional requerido por estas estrategias.

En cuanto a las técnicas de reducción de dimensiones de las secuencias previamente procesadas, la mayoría de los trabajos publicados, cuyo objetivo es clasificar TE, hacen uso de del análisis de componentes principales o PCA (Principal Component Analysis, por sus siglas en

inglés). No obstante, este abordaje presenta algunas falencias cuando se aplica sobre datos en los cuales no hay una relación lineal entre las variables independientes y la variable objetivo. La razón de ésta performance débil al momento de ser aplicado a un problema como la clasificación de secuencias, está dada por la naturaleza de la técnica. El PCA, que no tiene supuestos, proporciona una rotación del sistema de ejes que logra optimizar la varianza de los datos (la cual se cuantifica mediante el porcentaje de varianza explicada) en un menor número de dimensiones (componentes) (Jolliffe y Calima., 2016). Esta es la razón por la cual, además, el método es sensible a las variaciones de los datos. Razón por la cual es necesario escalar o estandarizar los datos previamente, agregando un paso más al procesamiento de las secuencias. Wu y colaboradores (1995), con el objetivo de realizar clasificación de familias y superfamilias de proteínas, diseñaron de un sistema de codificación de las secuencias residuales utilizando n-gramas y el algoritmo SVD (Descomposición en Vectores Singulares, en inglés, “Singular Value Decomposition”). Lograron así una representación de una matriz comprimida de los datos, y para llevar a cabo la clasificación, la implementación de un algoritmo basado en redes neuronales. No obstante, para realizar dicha tarea fue necesario el uso de una supercomputadora, demorando hasta 10 horas de CPU para la etapa del entrenamiento del modelo.

En el presente trabajo, el procesamiento de secuencias aminoacídicas demostró ser efectivo, escalable y poco costoso en términos de recursos computacionales. Por medio de la combinación de métodos de desarrollados en el área de la minería de texto (text mining), como la función CountVectorizer para la representación de las secuencias en forma vectorial y la técnica de reducción de dimensiones SVD, se obtuvieron porcentajes de varianza explicada mayores al 68% con una Especificidad y Sensibilidad mayor a 93% y 90%, respectivamente. Específicamente, el algoritmo de reducción de dimensiones que se implementó para procesar los datos de secuencias fue el desarrollado por Halko (Halko y cols., 2011) denominado *Randomized SVD*, el cual logra reducir significativamente los tiempos de cómputo y alcanza buena representación de las secuencias de entrada, generando una matriz comprimida con un máximo porcentaje de varianza explicada.

Otro aspecto relevante fueron los algoritmos de clasificación. Diversos trabajos hacen uso de clasificadores de tipo supervisado basados en la generación de hiperplanos, como es el caso de SVM. Este algoritmo, muestra resultados aceptables cuando es implementado en clasificación de secuencias (Barman y cols 2017; Lu y cols 2016). Este buen rendimiento radica en dos aspectos: la naturaleza de los datos, los cuales son no linealmente separables, y la posibilidad

de SVM de mitigar esta complejidad mediante el uso de funciones matemáticas o *kernels*. De este modo, mediante el uso de un *kernel*, como el de base radial o RBF (de las Ingles, Radial Basis Function), el algoritmo transforma los datos originales de entrada hacia un espacio de atributos altamente dimensional haciendo posible crear un hiperplano para la separación de las clases. En el presente trabajo se incluyó al clasificador SVM dentro de los *pipelines* experimentados. No obstante, se pudo observar una alta demanda en términos de recursos de CPU al momento de buscar los valores óptimos de parámetros C y Gamma cuando se utilizó el *kernel* RBF, llegando a consumir 10 horas en dicha etapa (en un ordenador portátil de CPU 8 núcleos y 16 GB RAM).

En la etapa de *testing*, si bien SVM alcanzó resultados aceptables en términos de *Accuracy*, fue el clasificador que obtuvo el valor más bajo en la métrica de Especificidad, mostrando una cierta pérdida de robustez debido a los falsos positivos generados. Además, fue uno de los algoritmos que más tiempo demandó en la etapa de *testing*, junto al clasificador *Random Forest*. Una tendencia similar para el rendimiento fue observada en la etapa de validación, en la cual los clasificadores (ya entrenados) fueron expuestos a genomas de validación de cepas bacterianas de referencia y con la distribución de la variable dependiente conocida. Es importante notar que SVM fue el clasificador que más tiempo de CPU requirió para llevar a cabo las estimaciones, específicamente 93 segundos sobre los cinco genomas. Este último punto es crítico, ya que, en el presente trabajo se buscó un algoritmo que no demande demasiado tiempo al momento del escalamiento. Por último, SVM no genera un resultado probabilístico en las clasificaciones, es decir no se pudo obtener una distribución de las probabilidades, lo cual fue dificultoso al momento de la interpretación de los resultados, tanto en el caso de las secuencias correctamente clasificadas (verdaderos positivos o VP), como en el caso de los falsos positivos (FP).

Como ha sido descrito previamente, en el presente trabajo se experimentó con cinco *pipelines*, cuyas tres etapas incluyeron: extracción de atributos de las secuencias, reducción de dimensiones y clasificación. Las cinco rutinas se diferenciaron por el tipo de algoritmo de clasificación incluido, lo cual implicó la estimación de diversos parámetros correspondientes a los clasificadores. Los valores óptimos de los parámetros de las funciones y algoritmos fueron buscados en subespacios de valores predefinidos, y de manera aleatoria implementando la estrategia *Randomized Grid Search* (Bergstra y Bengio, 2012). Existen dos estrategias usualmente adoptadas para la optimización de parámetros: exhaustiva y randomizada (aleatoria). En la primera, una grilla o *grid* es generada explicitando todos los valores de

parámetros de las funciones y algoritmos incluidos en el *pipeline*. Posteriormente, se realizan ejecuciones iterando sobre cada una de las combinaciones generadas por la grilla. Esta es la razón por la cual este abordaje es también denominado búsqueda por fuerza bruta. Por otro lado, la estrategia denominada *Randomized Grid Search* es similar, dado a que parte de una grilla de valores para los parámetros, pero, la búsqueda es aleatoria (en vez que secuencial). Los autores Begrstra y Bengio (2012) realizaron la comparación entre ambas estrategias. En dicho trabajo se implementó un clasificador complejo, en términos del número de parámetros a optimizar, como una red neuronal, sobre el mismo conjunto de datos e iguales recursos computacionales. Los autores demostraron que, mediante la optimización aleatoria (Randomized Grid Search), es posible hallar los valores óptimos para los parámetros, alcanzando igual o mejor rendimiento en la clasificación que su contrapartida de búsqueda exhaustiva. Todo esto asignando una cantidad aceptable de recursos y reduciendo de manera significativa el tiempo de cómputo. Aún más interesante, sobre los diversos conjuntos de datos incluidos en los experimentos, los autores han demostrado que, sobre el total de parámetros empleados, sólo algunos son relevantes para lograr una mejora en la clasificación.

Trabajos como el de Panta y colaboradores (2019), hacen uso de clasificador SVM, implementando la búsqueda exhaustiva de parámetros. Por otro lado, los trabajos de Nakano y colaboradores (2018) y Zhang y colaboradores (2018), que también han implementado algoritmos de ML para clasificar TE, si bien se describen los valores de los parámetros, no se explicita el abordaje utilizado para la optimización de dichos valores.

En el presente trabajo, basándonos en los resultados obtenidos en los estudios de Begrstra y Bengio (2012), se hizo uso de la estrategia randomizada para la optimización de los valores de parámetros de las funciones y algoritmos. Esto permitió hallar valores óptimos en tiempos razonables.

En la etapa de *training*, es decir donde se entrenó a los diversos clasificadores incluidos en el presente trabajo, el espectro de valores obtenidos varió en función del algoritmo de clasificación incluido en el *pipeline*. Si bien la naturaleza de los algoritmos experimentados fue diversa, en la etapa de *testing*, en base a las métricas seleccionadas, se observó que cada clasificador logró un buen desempeño. Estos resultados, nos demostraron la importancia de trabajar con valores óptimos tanto para funciones como para los algoritmos de clasificación.

En relación con las métricas de evaluación implementadas con el fin de mensurar el rendimiento de los diferentes algoritmos de clasificación, se utilizó: Sensibilidad, Especificidad, y *Accuracy*. La distribución de la variable dependiente en el conjunto de datos de partida, donde fueron entrenados todos los modelos, fue de tipo balanceada. No obstante, dado a que el objetivo fue identificar y predecir IS sobre los genomas, se focalizó en el uso de la medida de Sensibilidad. Dicha métrica es también denominada "*Recall*", y por definición, se centraliza en aquellos ejemplos que son identificados por el clasificador como positivos, o en el caso del presente problema, como IS. En otros términos, la Sensibilidad mide la efectividad del clasificador ante los verdaderos positivos (VP). Otro aspecto que fue tenido en cuenta al momento de seleccionar la Sensibilidad como la principal métrica, estuvo basado en diseño experimental. En el cual incluyó una etapa de análisis sobre los falsos positivos (FP) generados por la clasificación mediante un algoritmo de alineamiento (BLAST)

Específicamente, el *pipeline* que incluyó al algoritmo XGBoost fue considerado como el mejor en función de los resultados obtenidos. En base a los valores de parámetros devueltos por *GridSearch*, fue el *pipeline* que menos atributos extrajo (431) y que logró uno de los mayores porcentajes de varianza explicada (82%) con un valor para el parámetro K de 85. Es decir, una vez ejecutado el pipeline con XGBoost, la matriz de datos generada por SVD (de tipo *mxn*) estuvo definida por 16456 registros y 85 variables. Es importante notar que, un elevado porcentaje de varianza explicada y el hecho de trabajar con un número aceptable de atributos, para un algoritmo de clasificación basado en ensamble bajo una estrategia de *boosting*, permite mantener una baja complejidad en el modelo y así evitar, potencialmente el *overfitting* o sobreajuste. Es por ello que, el clasificador XGBoost tuvo también uno de los mejores valores para Accuracy (94%).

Para Especificidad y Sensibilidad, dos métricas que para el objetivo del presente trabajo fueron críticas y donde se hizo énfasis en obtener resultados consistentes, XGBoost generó resultados de 94,1% y 93,9% respectivamente. Posteriormente, en términos de costo computacional y tiempo, dicho algoritmo tomó 15.5 segundos en efectuar la clasificación en etapa de *testing* siendo el clasificador más rápido dentro de todos los experimentados. El tiempo de ejecución requerido fue un factor determinante dentro del presente trabajo y se debe a que dicho modelo fue incluido dentro de un *software* cuyo objetivo es ejecutarse sobre un pool de genomas.

Otro punto para resaltar es que además de la puesta a punto de un algoritmo de clasificación sobre un conjunto de datos de IS, se incluyó una fase adicional de validación. Esto consistió en ejecutar el clasificador seleccionado sobre cinco genomas bacterianos correspondientes a cepas

de referencia utilizadas en múltiples estudios experimentales: *E. coli* K-12, *S. enterica* serovar Typhi CT18, *A. baumannii* AYE, *S. aureus* Newman y *P. aeruginosa* PAO1. Esta fase nos permitió reducir la incertidumbre acerca del rendimiento del modelo al momento de clasificar datos nuevos, a los cuales nunca fue expuesto previamente. Además, evidenciar la presencia, o no, del fenómeno de *overfitting* o sobreajuste inherente a la naturaleza de los datos del problema y, por otro lado, a la complejidad de los algoritmos de clasificación incluidos en el estudio. Es importante resaltar que la proporción de secuencias de inserción en estos genomas de referencia fue significativamente baja, no superó el 2% en ningún genoma, incluso el 0.1% en el caso de *P. aeruginosa.*

Dado este marcado desbalance en la variable dependiente, se incluyó la métrica F1 score, la cual se define como la media armónica ponderada de Precisión y Sensibilidad lo que, al ser un producto de dichas métricas, representa de una manera global el rendimiento de la clasificación mitigando así la sobrerrepresentación de uno de los dos niveles de la variable dependiente. El promedio de la medida F1 generado por XGBoost sobre los genomas de referencia fue 85.14, en tanto que para Especificidad y Sensibilidad los valores fueron de 81.5% y 89.5%, respectivamente. De este modo, teniendo en cuenta los resultados tanto en entrenamiento del modelo como en la etapa de validación, el algoritmo XGBoost mostró consistencia en términos de métricas de rendimiento y estabilidad sobre los diversos *sets* de datos donde fue desplegado.

Los resultados satisfactorios del clasificador XGBoost se atribuyen a múltiples razones. En primer lugar, el tipo de datos que se busca clasificar, secuencias de caracteres (o *strings*). Como se explicó anteriormente, las secuencias residuales son datos de tipo no linealmente separables, es decir que no se puede implementar un hiper-plano sobre los datos para discernir las clases, en este caso IS *vs*. non-IS.

Un árbol de decisión es un algoritmo que realiza particiones recursivas sobre el espacio de atributos en los datos, generando sub-regiones paralelas al eje cartesiano X, “etiquetando” o asignando los puntos a la variable dependiente en función de un determinado criterio. Es decir, en base de una regla inicial, que gobierna el nodo “raíz”, los datos se dividen en dos subconjuntos no superponibles, denominados también nodos “hijos”. La calidad de esta división es cuantificada mediante una métrica de impureza como, por ejemplo, el índice de Gini, Entropía, o bien, la ganancia de información (*information gain*). Esta rutina iterativa converge cuando se llega a los denominados nodos terminales donde, dicha impureza alcanza un mínimo. Este proceso sobre el conjunto de atributos hace posible la separación y la asignación de cada uno de los puntos a

una clase, sin la necesidad de establecer un hiperplano sobre el conjunto de datos. Esta propiedad hace que los árboles de decisión sean una alternativa viable para implementar en problemas como el abordado en el presente trabajo. Un ejemplo de ello son los algoritmos XGBoost y *Random Forest*.

XGboost está comprendido dentro de los denominados algoritmos basados en ensamble. Esto se fundamenta en que es un meta-estimador compuesto de múltiples estimadores simples o también denominados estimadores "débiles". La lógica subyacente es que este meta-estimador alcanzará una mejor capacidad de generalización que los estimadores aplicados individualmente. El concepto de *boosting* fue expuesto por primera vez en 1990 (Freund y Schapire, 1990). Este tipo de abordaje se centraliza en aquellas instancias que son más "difíciles" para clasificar. Esto representa, dejar que los estimadores débiles aprendan de las muestras clasificadas erróneamente para luego contribuir con un buen rendimiento en el ensamble de estimadores.

Bajo este marco conceptual, el algoritmo XGBoost incluye por defecto ciertos parámetros y funcionalidades que le permiten así, llevar a cabo la tarea de aprendizaje y clasificación. Algunos de ellos son: algoritmo de optimización de descenso en gradiente, parámetros de regularización, la capacidad de implementar ejecuciones en paralelo, validar resultados de ejecuciones mediante cross-validation y aspectos que dotan de más flexibilidad al algoritmo, como son la posibilidad de seleccionar la función objetivo, así como métricas de evaluación de rendimiento.

Brevemente, el descenso en gradiente es un algoritmo de optimización que se utiliza para minimizar una función objetivo determinada. Realiza esta tarea recorriendo iterativamente en la dirección del descenso más pronunciado según lo definido por el negativo del gradiente. En el caso de XGBoost, lo que se busca optimizar es la función de costo o *loss function*. Es decir, se busca minimizar la diferencia entre el valor predicho y el observado (o actual), en el caso de una clasificación como la que se busca en el presente trabajo. El algoritmo realiza movimientos en forma de "pasos" cuyo valor es determinado por el parámetro denominado "velocidad o tasa de aprendizaje" o "*learning rate*". El valor implementado en el presente trabajo fue de 0,1. Al igual que cualquier parámetro, el valor varía en función de: el tipo de datos con el que se entrenará al algoritmo, al problema que estemos abordando y del costo computacional que se esté dispuesto a afrontar. El algoritmo de descenso en gradiente logra alcanzar la convergencia cuando se encuentra un mínimo local óptimo.

Una característica del estimador XGBoost, conservada desde su predecesor Random Forest (Breiman, 2001), es el parámetro denominado sub-muestreo de columna o “colsample_bytree” como se denomina en el algoritmo (cuyo valor utilizado en el presente trabajo fue 0,6). Este parámetro hace referencia al número (fracción) de columnas incorporadas al momento de la construcción de los diferentes estimadores o arboles individuales. Dicho parámetro, tiene un impacto directo previniendo el sobre ajuste u *overfitting*, dado a que permite romper con el fenómeno de multicolinealidad entre variables.

Otro punto relevante es que la función de costo en XGBoost incluye el término de regularización, por lo cual, los autores la denominan función de aprendizaje regularizada (Chen y Gestrin 2016). Este, es un algoritmo basado en árboles de decisión, y esta función resulta en una ventaja significativa como se explica a continuación. Uno de los parámetros mediante los cuales se puede controlar la regularización es gamma. El objetivo de la regularización es intervenir en la complejidad del modelo. En XGBoost, el parámetro gamma es un multiplicador de Lagrange, que controla la complejidad de un árbol de decisión determinado, especificando una reducción de pérdida mínima requerida para efectuar una partición adicional en un nodo-hoja de dicho árbol (Chen y Gestrin 2016). Cuando gamma es especificado, como en el presente trabajo, XGBoost genera el árbol hasta una profundidad máxima especificada, pero luego efectúa una poda (o *pruning*) con el fin de encontrar y eventualmente remover divisiones que no cumplan con el valor de gamma especificado (Chen y Gestrin 2016). En este trabajo, el valor de gamma especificado fue de 1.

Como se mencionó anteriormente, mediante el parámetro de regularización es posible controlar la complejidad del modelo buscando alcanzar un balance óptimo entre bias (sesgo) y varianza. En el caso del presente trabajo, el objetivo fue entrenar un identificador de secuencias de inserción basado en algoritmos de aprendizaje automatizado (*machine learning)*. Se buscó entrenar un estimador donde el error de generalización sea el mínimo, buscando mitigar fenómenos como como bias y varianza. Cuando un modelo se torna cada vez más complejo, comienza a utilizar en gran manera los registros presentes en los datos de entrenamiento y comienza a adaptarse a estructuras más complejas, promoviendo el fenómeno de *overfittin*g o sobre-ajuste. Es por eso por lo que el término complejidad está relacionado con el sobreajuste. Asimismo, tiene lugar un incremento significativo de la varianza, la cual puede definirse como variabilidad en las predicciones generadas por el algoritmo. Esto se debe a que hay una pérdida de consistencia cuando el estimador es ejecutado sobre instancias a las que nunca fue expuesto,

ya sea del mismo conjunto de datos, como es en el caso de *cross-validation*, o bien ante un nuevo *set* como en la etapa de *testing*. Debido a ello, se dice que el algoritmo es susceptible (y poco consistente) frente a la aleatoriedad de los datos.

Como contrapartida, el *bias* o sesgo, tiene lugar cuando la varianza es baja, y describe cuán lejos están los valores observados de los esperados. Tiene su máximo valor cuando el número de registros a los que fue expuesto el estimador, es mínimo. En otras palabras, el bias se incrementa cuando el número de instancias presentes en la etapa de entrenamiento del estimador no fue la suficiente.

Siendo estos dos fenómenos los tipos de error presentes en cualquier estimador, en el presente trabajo fue posible mitigar el sobre ajuste y alcanzar un equilibrio *bias* - varianza, no solo haciendo uso de un clasificador dotado de parámetros específicos, sino también mediante las estrategias de selección y evaluación de modelos incorporando diferentes conjuntos de datos inherentes al diseño experimental. Es decir, se utilizó un conjunto de datos inicial con el fin de entrenar al estimador. Posteriormente se evaluó el error de predicción durante la etapa de *testing*, y, por último, sobre los cinco genomas de referencia, se determinó el error de generalización.

Cabe notar que los cinco genomas de referencia pertenecían a especies bacterianas diferentes y relativamente lejanas en términos evolutivos, con lo cual las instancias que fueron presentadas al algoritmo también mostraron una variación. De este modo, el estimador diseñado fue expuesto a niveles de aleatoriedad significativos en los datos. No obstante, y positivamente, presentó valores estables para las métricas de Sensibilidad y Especificidad, indicando claramente que el clasificador implementado no fue susceptible al efecto de aleatoriedad, siendo consistente frente al fenómeno sobreajuste u *overfitting*. Además, tampoco fue impactado negativamente por un sesgo elevado, dado a que las predicciones de secuencias de inserción alcanzaron valores satisfactorios en las métricas de rendimiento seleccionadas.

Adicionalmente, se decidió incluir una etapa de calibrado, en la cual se buscó el punto de corte, para establecer las clases, en función de las probabilidades predichas. En otras palabras, se buscó conocer el valor de la probabilidad a partir del cual se puede establecer que, efectivamente se trata de una secuencia de inserción o no. En los resultados de dicha etapa, no se observaron cambios significativos en relación con las métricas de performance. Con lo cual se puede establecer que el algoritmo fue estable y que, aun forzando los puntos de corte, mantuvo la consistencia. Con lo cual, observando dichos resultados en esta etapa, se decidió trabajar con el

punto de corte obtenido por defecto por XGBoost luego de la búsqueda de hiperparámetros, y sin el punto de corte (en las probabilidades) establecido por el investigador mediante calibrado.

Sobre los genomas de validación incluidos en el presente estudio XGBoost mostró resultados sólidos. El primer aspecto que fundamenta dicha conclusión son los valores obtenidos tanto en validación como en la etapa de escalado en la especie *E.coli* (Tabla 25)

Las anotaciones de los genomas no son certeras para diversos grupos de genes, particularmente para IS, debido a la variabilidad genética intrínseca de estos genes. Algunos de estas características pueden ser: la variabilidad a nivel de la tasa de mutación más alta que el resto de los genes, como también, variabilidad en la estructura génica de dichos elementos móviles. Esto se pudo evidenciar por la elevada frecuencia de falsos positivos observados, tanto en la etapa de validación sobre los cinco genomas de referencia, como en el escalado en la especie *E. coli*.

Aún más, en la etapa de validación el genoma de la especie *P. aeruginosa*, en función de la anotación por defecto y luego de la clasificación, una proporción significativa de sus secuencias (32%) fueron catalogadas como falsos positivos. Esta proporción se redujo luego de ejecutar un análisis de secuencia mediante BLAST, donde se lograron identificar 73 secuencias de inserción, lo cual representó un 3.9% del total de falsos positivos hallados en la especie según la anotación por defecto. Particularmente, los falsos positivos presentaban niveles de homología menor al 95% frente a las secuencias de ISs de referencia, por los que podrían ser consideradas posibles nuevas ISs. El estimador entrenado fue capaz de reconocer dichas IS. La misma tendencia se observó en el resto de las especies y con las secuencias inicialmente marcadas como falsos positivos, pero que en realidad fueron correctamente identificadas por el clasificador XGBoost. En la etapa de escalamiento, donde sólo se utilizó un gran número de genomas de la especie *E. coli* (n = 1,196), el resultado fue similar (Tabla 25). Del mismo modo que en validación, el clasificador fue capaz de reconocer secuencias de inserción con un elevado grado de certeza, y dichos resultados fueron fundamentados por las ejecuciones posteriores con el método de búsqueda por similitud (BLAST) que consideramos como el "*gold standard"*.

Otro punto que vale la pena notar son los resultados en las métricas de rendimiento, tanto en la etapa de validación como en el escalado, que fueron similares. Es decir, las frecuencias de falsos positivos y verdaderos positivos detectados por el *software* fueron similares en ambas poblaciones, las cuales no mostraron diferencias. Basándose en estos resultados, es posible fundamentar la estabilidad del clasificador desarrollado.

En función de las secuencias de inserción identificadas, en la etapa de validación han sido detectadas cinco familias: IS3, ISKra4, ISNCY, e IS91, distribuidas en todos genomas de las especies incluidas en esta fase: *E. coli, S. enterica*, *A. baumannii*, *S. aureus* y *P. aeruginosa.* Si bien la frecuencia relativa fue diversa, en algunas familias de IS se observó una tendencia hacia una especie en particular. Dicha tendencia, podría sugerir la aparición temprana de alguna familia de IS en particular, como también así, una evolución huésped especifico de las ciertas familias de ISs. Mahillon et al., explica la posibilidad que ciertas secuencias de inserción, debido a la especificidad de sitio de inserción, sean específicas de especies bacterianas, mientras que otras familias de ISs presentan una mayor promiscuidad (Mahillon et al 1998).

Por otro lado, en el caso de la familia ISKra4 no está claro el origen, pero se ha encontrado con mayor frecuencia en *Leptolyngbya sp.* y *Nostoc punctiforme*. En nuestro estudio dicha familia fue hallada con mayor frecuencia en la especie *E. coli* (0.33) y luego distribuida de igual manera en *S. enterica* (0.25) y *P. aeruginosa* (0.25). Respecto a la familia ISNCY, su origen fue observado en mayor proporción en las especies *P. aeruginosa y Burkholderia cepacia*. Sin embargo, en el presente estudio se observó una elevada frecuencia de dicha familia (0.40) en la especie *S. enterica*. Del mismo modo, en el presente trabajo la frecuencia relativa observada en la familia IS91 fue significativamente mayor sobre la especie *A. baumanii* (0.36). No obstante, el origen de la familia IS91 fue descrito en la especie *E. coli*.

En esta tesis se estudió la presencia de familias de IS identificadas en 1196 genomas de *E. coli* en la etapa de escalado. La familia IS3 fue la que estuvo presente en mayor proporción, con un 29%. Según la base de datos ISFinder, esta familia no solo tiene como origen la especie *E. coli* y también puede observarse en la especie *P. aeruginosa*. Otro resultado interesante hallado en el escalado fue el caso de la familia IS1595, presente en un 14%. En función de ISFinder, dicha secuencia ha sido reportada en otras especies como *Natrinema pellirubrum, Haemophilus influenzae* o *Bacteroidetes bacterium*. Del mismo modo, la familia IS200/IS605 presentando una proporción de un 8%, su origen es *Natronomonas pharaonis, Haloquadratum walsbyi* o *Natrialba magadii.*

Como se mencionó anteriormente, la plasticidad genética, principalmente asociada a la transferencia horizontal y en particular a los elementos genéticos transponibles, en bacterias les confiere ventajas en términos funcionales impactando directamente en su evolución. Dichas características tienen una influencia positiva en su capacidad adaptación a medios hostiles.

Además, este tipo de fenómenos les otorga a las bacterias capacidades especiales como resistencia a determinados antibióticos lo cual hace que tenga relevancia clínica.

En función de la estabilidad que mostró el estimador XGBoost, no solo por las características intrínsecas del algoritmo sino por la diversidad de conjunto de datos donde fue ejecutado, se puede remarcar que el clasificador diseñado es una alternativa para la identificación de IS.

Como se explicó anteriormente, el análisis de secuencia mediante BLAST puede llegar a demorar varias horas, incluso días, en generar resultados sobre un conjunto relativamente grande de secuencias. En el presente estudio, cuando se ejecutó el análisis de alineamiento sobre las 5382000 secuencias aminoacídicas (provenientes de 1196 genomas completos de *E. coli*) el tiempo total de análisis fue de sesenta horas. En contraposición con los treinta minutos demandados por el sistema aquí propuesto.

Hasta el momento no se han reportado trabajos similares cuyo objetivo sea identificar y predecir IS mediante el uso de algoritmos de aprendizaje automatizado *(machine learning)*. En función de los resultados observados se puede resaltar que dicho algoritmo fue estable y consistente. El tiempo de cómputo fue razonable, no superando los cuarenta minutos en un ordenador portátil, cuando fue ejecutado sobre los 1196 genomas durante la etapa de escalado. Por todo lo antes mencionado concluimos que el software desarrollado en el presente trabajo es una alternativa viable y robusta al momento de identificar y predecir secuencias de inserción.

Conclusión

En el presente trabajo, se demostró que mediante el desarrollo de un software basado en algoritmos de *machine learning* es posible identificar secuencias de inserción bacterianas. El software desarrollado fue dotado de cierta flexibilidad, permitiendo acoplar clasificadores de ma*chine learning* con algoritmos de alineamiento de secuencias como BLAST. De este modo, fue posible realizar ejecuciones de detección de IS y en una etapa posterior, su validación mediante alineamiento de secuencias.

Una de las ventajas de hacer uso de esta herramienta, fue poder efectuar procesos integrados en la misma rutina de trabajo sin necesidad de hacer use de aplicaciones *online*, o mediante el uso de una terminal de Linux. Además, la ejecución para detección de IS, se pueden llevar a cabo de manera “local” y la instalación del software no requiere de un entrenamiento avanzado por parte del usuario.

El algoritmo XGBoost mostró resultados contundentes en todas las etapas experimentales. No solo en términos de la efectividad al momento de detectar IS, sino también por el tiempo de cómputo demandado. Así, el sistema desarrollado permitió clasificar secuencias aminoacídicas a una razón de 180.000 secuencias por minuto (aproximadamente) en un ordenador portátil. De este modo. El software presentado consta de un poder de computo considerable.

Basándonos en los resultados obtenidos, es posible proponer el método como alternativa al método de BLAST para detectar secuencias de inserción bacterianas.

Referencias

Alter, O., Brown P and D. Botstein. (2000). Singular value decomposition for genome-wide expression data processing and modeling. Proc. Natl. Acad. Sci. USA. 97:10101–10106.

Altschul S, Gish W. ,Miller W. , Myers E, and Lipman D.(1990). “Basic local alignment search tool,” Journal of Molecular Biology, vol. 215, no. 3, pp. 403 – 410.

Amidi A, Amidi S, Vlachakis D, Paragios N, and Zacharaki E.(2016) "A machine learning methodology for enzyme functional classification combining structural and protein sequence descriptors." In International Conference on Bioinformatics and Biomedical Engineering, pp. 728-738. Springer, Cham.

Badrul S, Karypis G, Konstan J, and Riedl J. (2002). "Incremental singular value decomposition algorithms for highly scalable recommender systems." In Fifth International Conference on Computer and Information Science: 27-28.

Baldi P, Brunak S. (2001). Bioinformatics: The Machine Learning Approach, The MIT Press, Cambridge, Mass, USA.

Barman RK, Mukhopadhyay A, Das S. An improved method for identification of small non-coding RNAs in bacteria using support vector machine. Sci Rep. 2017 Apr 6;7:46070. doi: 10.1038/srep46070. PMID: 28383059; PMCID: PMC5382675.

Barve A, Ghaskadbi S, Ghaskadbi. (2013). “Structural and sequence similarities of *hydra xeroderma pigmentosum* a protein to human homolog, suggest early evolution and conservation. BioMed Research International, vol. 2013, Article ID 854745, 9 pages.

Betancor L , Gadea M and Flores K. ed., (2006). Genética bacteriana. In: Temas de Bacteriología y Virología Médica, 1st ed. [online] Uruguay: Instituto de higiene, departamento de bacteriología y virología. Available at: http://www.higiene.edu.uy/cefa/cefaed2006.htm

Bergstra J and Bengio Y. (2012). Random search for hyper-parameter optimization. J. Mach. Learn. Res. 13, null (3/1/2012), 281–305.

Berman H, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN and Bourne PE. (2000) The Protein Data Bank. Nucleic Acids Res. Jan 1;28(1):235-42. doi: 10.1093/nar/28.1.235. PMID: 10592235; PMCID: PMC102472.

Bonomo R, Van Zile P, Li Q, Shermock K, McCormick W and Kohut B. (2007). Topical triple-antibiotic ointment as a novel therapeutic choice in wound management and infection prevention: a practical perspective. Expert Rev Anti Infect Ther. 2007 Oct;5(5):773-82. doi: 10.1586/14787210.5.5.773. PMID: 17914912.

Borro L., Stanley R, Yamagishi M, Mancini A, Jardine J, Mazoni I, Dos Santos E, Higa R, Kuser P, and Neshich G.(2006) "Predicting enzyme class from protein structure using Bayesian classification." Genet. Mol. Res 5, no. 1s: 193-202.

Boström, H. (2008). Calibrating random forests. Proceedings - 7th International Conference on Machine Learning and Applications, ICMLA 2008, 121–126. https://doi.org/10.1109/ICMLA.2008.107

Bottou L.(1991) Stochastic Gradient Learning in Neural Networks, Proceedings of Neuro-Nîmes 91, EC2, Nimes, France.

Breiman, L. (2001) Random Forests. Machine Learning, 45, 1, 5-32.

Brigulla, M., and Wackernagel, W. (2010). Molecular aspects of gene transfer and foreign DNA acquisition in prokaryotes with regard to safety issues. Applied microbiology and biotechnology, 86(4), 1027–1041. https://doi.org/10.1007/s00253-010-2489-3.

Bryant JA, Sellars LE, Busby SJW, Lee DJ. 2015. Chromosome position effects on gene expression in Escherichia coli K-12. Nucleic Acids Res 42:11383–11392. https://doi.org/10.1093/nar/gku828.

Burkhard R, Twilight zone of protein sequence alignments, *Protein Engineering, Design and Selection*, Volume 12, Issue 2, February 1999, Pages 85-94, https://doi.org/10.1093/protein/12.2.85

Calamante F, Gadian D, and Connelly A. (2000). "Delay and dispersion effects in dynamic susceptibility contrast MRI: simulations using singular value decomposition." Magnetic resonance in medicine 44, no. 3: 466-473.

Cambray G, Guerout A, and Didier Mazel (2010). Integrons Annual Review of Genetics. 44:1, 141-166

Caragea C, Silvescu A, and Mitra P.(2012) “Protein sequence classification using feature hashing,” Proteome Science, vol. 10, no. 1, pp. 1–8.

Carroll, Marion & Nguyen, Son & Batzer, Mark. (2001). Genome Databases. 10.1038/npg.els.0003027.

Carregari V, Floriano R, Rodrigues-Simioni L, Vischi F.(2013).“Bio-chemical, pharmacological, and structural characterization of new basic PLA2 bbil-tx from Bothriopsis bilineata snake venom,” BioMed Research International, vol. 2013, Article ID 612649, 12 pages.

Cao, D.S, Xu Q, Liang Y,. (2013). propy: a tool to generate various modes of Chou’s PseAAC. Bioinformatics, 29, 960–962.

Camacho, C., Coulouris, G., Avagyan, V, Ma N, Papadopoulus J, Bealer K and Madden T. (2009) BLAST+: architecture and applications. BMC Bioinformatics10, 421. https://doi.org/10.1186/1471-2105-10-421

Chen I, Christie PJ, Dubnau D. (2005). The ins and outs of DNA transfer in bacteria. Science. 2005 Dec 2;310(5753):1456-60. doi: 10.1126/science.1114021. PMID: 16322448; PMCID: PMC3919525.

Chen C, McGarvey P, Huang H, Wu C.(2010) “Protein bioinformatics infrastructure for the integration and analysis of multiple high-throughput omics data,” Advances in Bioinformatics”, vol. 2010, Article ID 423589.

Chen, T. and Guestrin, C. (2016) XGBoost: A Scalable Tree Boosting System. In Proceedings of the Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (San Francisco, California, USA, 2016). ACM.

Cock P, Antao T, Chang J, Chapman B, Cox C, Dalke A, Friedberg I, Hamelryck T, Kauff F, Wilczynski B, de Hoon M.(2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics, Bioinformatics, Volume 25, Issue 11.Pages 1422–1423, https://doi.org/10.1093/bioinformatics/btp163.

Colangeli R, Haq A, Arcus VL, Summers E, Magliozzo RS, McBride A, Mitra AK, Radjainia M, Khajo A, Jacobs WR Jr et al (2009) The multifunctional histone-like protein Lsr2 protects mycobacteria against reactive oxygen intermediates. Proc Natl Acad Sci USA 106:4414–4418

Cong H, Zhang M, Zhang Q, Gong J, Cong H, Xin Q, He S.(2013). “Analysis of structures and epitopes of surface antigen glycoproteins expressed in brady- zoites of Toxoplasma gondii,” BioMed Research International. Article ID 165342, 9 pages.

Cortes, C. and Vapnik, V.(1995) Support-Vector Networks. Machine Learning, 20, 3, 273-297.

Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., ... & De Hoon, M. J. (2009). Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, *25*(11), 1422.

Craig NL. V(D)J (1996) recombination and transposition: closer than expected. Science.;271(5255):1512. doi: 10.1126/science.271.5255.1512. PMID: 8599105.

Dantas G, Sommer MO, Oluwasegun RD, Church GM.(2008) Bacteria subsisting on antibiotics. Science. 320(5872):100-3. doi: 10.1126/science.1155157. PMID: 18388292.

Deerwester, S., Dumais,ST., Furnas, Landaur, T.K., & Harshman, R. (1990). Indexingby latent semantic analysis. Journal of American Society for Information Science, 41:391-407.

De Castro Nunes, R.; Orozco-Arias, S.; Crouzillat, D.; Mueller, L.A.; Strickler, S.R.; Descombes, P.; Fournier, C.; Moine, D.; de Kochko, A.; Yuyama, P.M.;(2018). Structure and distribution of centromeric retrotransposons at diploid and allotetraploid Coffea centromeric and pericentromeric regions. Front. Plant Sci. 9, 175.

Dryselius R, Izutsu K, Honda T, Iida T. 2008. Differential replication dynamics for large and small Vibrio chromosomes affect gene dosage, expression and location. BMC Genomics 9:559. https://doi.org/10.1186/ 1471-2164-9-559.

diCenzo GC, Finan TM. 2017. The divided bacterial genome: structure, function, and evolution. Microbiol Mol Biol Rev 81:e00019-17. https://doi.org/10.1128/ MMBR.00019-17.

Dorman CJ, Deighan P (2003) Regulation of gene expression by his- tone-like proteins in bacteria. Curr Opin Genet Dev 13:179–184

Duff I , Grimes R , and Leiws J (1989), Sparse matrix test problems, ACM Trans. Math. Soft., 15 (1989), pp. 1-14.

Escherich T. Die darmbakterium des neugebornen und sauglangs. Fortschrite der Medizin. 1885;3:515-22.

Ewing, A.D. Transposable element detection from whole genome sequence data. Mobile DNA 6, 24 (2015). https://doi.org/10.1186/s13100-015-0055-3

Freedman, D. A.(2009) Statistical models: theory and practice. Cambridge university press.

Freund, Y., & Schapire, R. (1999). A Short Introduction to Boosting. Journal of Japanese Society for Artificial Intelligence, 14(5):771-780.

Grinsted J, de la Cruz F and Schmitt R. (1990). The Tn21 subgroup of bacterial transposable elements. Plasmid. 24(3):163-89. doi: 10.1016/0147-619x(90)90001-s. PMID: 1963947.

Gomide J, Melo-Minardi R, Dos Santos MA, Neshich G, Meira W Jr, Lopes JC, Santoro M.(2009). Using linear algebra for protein structural comparison and classification. Genet Mol Biol.;32(3):645-51. doi: 10.1590/S1415-47572009000300032. Epub 2009 Sep 1. PMID: 21637532; PMCID: PMC3036040.

Gupta, Chhote Lal & Bihari, Anand and Tripathi, Sudhakar. (2019). Human Protein Sequence Classification using Machine Learning and Statistical Classification Techniques. International Journal of Recent Technology and Engineering. 8. 3591-3599. 10.35940/ijrte.B3224.078219. H

Halko N , Martinsson P. G. , and Tropp J. A. (2011). Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. SIAM Rev. 53, 2, 217–288. DOI:https://doi.org/10.1137/090771806

Hartl DL, Ajioka JW, Cai H, Lohe AR, Lozovskaya ER, Smoller DA, Duncan IW. (1992) Towards a Drosophila genome map. Trends Genet. 8(2):70-5. doi: 10.1016/0168-9525(92)90353-6. PMID: 1566375.

Harrison PW, Lower RPJ, Kim NKD, Young JPW. 2010. Introducing the bacterial “chromid”: not a chromosome, not a plasmid. Trends Microbiol 18:141–148. https://doi.org/10.1016/j.tim.2009.12.010.

Hall BG, Parker LL, Betts PW, DuBose RF, Sawyer SA, Hartl DL (1989). IS103, a new insertion element in Escherichia coli: characterization and distribution in natural populations. Genetics. 121(3):423-31. PMID: 2541046; PMCID: PMC1203630.

Hide, Winston. (2010) The South African EMBnet Node: AGM 2010 report. EMBnet.journal, [S.l.], v. 16, n. 1, p. pp. 25-27, july 2010. ISSN 2226-6089. Available at: <https://journal.embnet.org/index.php/embnetjournal/article/view/191/393>. Date accessed: 18 jan. 2021. doi:https://doi.org/10.14806/ej.16.1.191.

Hoecker A and Vakhtang K.(1996). "SVD approach to data unfolding." Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 372, no. 3: 469-481.

Hoede C, Arnoux S, Moisset M, Chaumier T, Inizan O, Jamilloux O, and Quesneville H. (2014) “PASTEC: an automatic transposable element classification tool.” PloS one, vol. 9, no. 5, p. e91929.

Hyatt, D., Chen, GL., LoCascio, P.F. *et al.* Prodigal: prokaryotic gene recognition and translation initiation site identification. *BMC Bioinformatics* **11,** 119 (2010). https://doi.org/10.1186/1471-2105-11-119

Huda al-nayyef, Guyeux C & Bahi J, Jacques. (2014). A Pipeline for Insertion Sequence Detection and Study for Bacterial Genome. Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI).

Inoue, Yasuhiro, Takeya, Masaru, Sasaki, Aeni, Kaku, Hisatoshi. (2005). Genome Sequence of Xanthomonas oryzae pv. oryzae Suggests Contribution of Large Numbers of Effector Genes and Insertion Sequences to Its Race Diversity. JARQ. 39. 275-287. 10.6090/jarq.39.275.

Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: a review and recent developments. Philosophical transactions. Series A, Mathematical, physical, and engineering sciences, 374(2065), 20150202. https://doi.org/10.1098/rsta.2015.0202

Jones, P., Binns, D., Chang, H. Y., Fraser, M., Li, W., McAnulla, C., McWilliam, H., Maslen, J., Mitchell, A., Nuka, G., Pesseat, S., Quinn, A. F., Sangrador-Vegas, A., Scheremetjew, M., Yong, S. Y., Lopez, R., & Hunter, S. (2014). InterProScan 5: genome-scale protein function classification. Bioinformatics (Oxford, England), 30(9), 1236–1240.

Jurka J, Klonowski P, Dagman V, Pelton P. Censor—a program for identification and elimination of repetitive elements from DNA sequences. Computers & Chemistry. 1996; 20(1):119—121. http://dx. doi.org/10.1016/S0097-8485(96)80013-1.

Junier I. 2014. Conserved patterns in bacterial genomes: a conundrum physically tailored by evolutionary tinkering. Comp Biol Chem 53:125–133. https://doi.org/10.1016/j.compbiolchem.2014.08.017.

Ke G., Meng Q., Finley T., Wang T., Chen W., Ma W., Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. Advances in Neural Information Processing Systems, 30, 3146–3154.

Lawrence JG. 2003. Genome organization: selection, selfishness, and serendipity. Annu Rev Microbiol 57:419 – 440. https://doi.org/10.1146/ annurev.micro.57.030502.090816.

Lee, B.J., Lee, H.G., Ryu, K.H.(2008) Design of a novel protein feature, enzyme function classification. In: CIT Workshops. IEEE 8th International Conference on Computer and Information Technology Workshops, pp. 450–455. IEEE (2008)

Liu L, Cui J, Zhang, Wei T, Jiang P, Wang Z. (2013). “Analysis of structures, functions, and epitopes of cysteine protease from Spirometra erinaceieuropaei spargana,” BioMed Research International, vol. 2013, Article ID 198250, 7 pages.

Loureiro, T., Camacho, R., Vieira, J., Fonseca, N.A.(2013) Boosting the Detection of Transposable Elements Using Machine Learning. In 7th International Conference on Practical Applications of Computational Biology & Bioinformatics; Springer: Heidelberg, Germany, pp. 85–91.

Lu B, Leong HW. GI-SVM: A sensitive method for predicting genomic islands based on unannotated sequence of a single genome. J Bioinform Comput Biol. 2016 Feb;14(1):1640003. doi: 10.1142/S0219720016400035. PMID: 26907990.

Machado J, Costa A, Quelhas M. (2013). “Can power laws help us understand genes and proteome information?” Advances in Mathematical Physics, vol. 2013, Article ID 917153, 10 pages.

Mahillon J, Chandler M (1998). Insertion sequences. Microbiol Mol Biol Rev. 62(3):725-74. doi: 10.1128/MMBR.62.3.725-774.1998. PMID: 9729608; PMCID: PMC98933.

Mahillon J, Kirkpatrick HA, Kijenski HL, Bloch CA, Rode CK, Mayhew GF, Rose DJ, Plunkett G 3rd, Burland V, Blattner FR.(1998) Subdivision of the Escherichia coli K-12 genome for sequencing: manipulation and DNA sequence of transposable elements introducing unique restriction sites. Gene. 223(1-2):47-54. doi: 10.1016/s0378-1119(98)00365-5. PMID: 9858680.

Mahillon J, Léonard C, Chandler M.(1999) IS elements as constituents of bacterial genomes. Res Microbiol. 150(9-10):675-87. doi: 10.1016/s0923-2508(99)00124-2. PMID: 10673006.

Mazel D, Davies J.(1999) Antibiotic resistance in microbes. Cell Mol Life Sci. 30;56(9-10):742-54. doi: 10.1007/s000180050021. PMID: 11212334.

Machado J, Costa A, Quelhas M. (2013). “Can power laws help us understand genes and proteome information?” Advances in Mathematical Physics, vol. 2013, Article ID 917153, 10 pages.

McClintock B. Controlling elements and the gene.(1956) Cold Spring Harb Symp Quant Biol. 21:197-216. doi: 10.1101/sqb.1956.021.01.017. PMID: 13433592.

Mukherjee A, Sokunbi AO, Grove A (2008) DNA protection by his- tone-like protein HU from the hyperthermophilic eubacterium Thermotoga maritima. Nucleic Acids Res 36:3956–3968

Nakano, F.K.; Martiello Mastelini, S.; Barbon, S.; Cerri, R.(2017) Stacking methods for hierarchical classification. In Proceedings of the 16th IEEE International Conference on Machine Learning and Applications, Cancun, Mexico, 18–21.Volume 2018–Janua, pp. 289–296.

Rocha EPC. 2004. The replication-related organization of bacterial genomes. Microbiology 150:1609 –1627. https://doi.org/10.1099/mic.0.26974-0.

Orozco-Arias S, Isaza G, Guyot R.(2019) Retrotransposons in Plant Genomes: Structure, Identification, and Classification through Bioinformatics and Machine Learning. Int J Mol Sci. 20(15):3837. doi: 10.3390/ijms20153837. PMID: 31390781; PMCID: PMC6696364.

Rao JE, Miller PS, Craig NL.(2000) Recognition of triple-helical DNA structures by transposon Tn7. Proc Natl Acad Sci U S A. 97(8):3936-41. doi: 10.1073/pnas.080061497. PMID: 10737770; PMCID: PMC18120.

Rocha EPC. 2008. The organization of the bacterial genome. Annu Rev Genet 42:211–233. https://doi.org/10.1146/annurev.genet.42.110807 .091653.

Panta, M., Mishra, A., Tamjidul Hoque, M., and Atallah, J.,(2019) “Machine Learning based Prediction of Hierarchical Classification of Transposable Elements”, BIOKDD.

Pedregosa, F., Varoquaux, Gael , G, *et al*. (2012). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 12.

Pettijohn DE (1988) Histone-like proteins and bacterial chromosome structure. J Biol Chem 263:12793–12796

Platt R, Vandewege M, Ray D. (2017).Mammalian transposable elements and their impacts on genome evolution.Chromosome Res (2018) 26:25–43.

Santos, B.; Cerri, R.; Lu, R.W. (2016). A New Machine Learning Dataset for Hierarchical Classification of Transposable Elements. In Proceedings of the XIII Encontro Nacional de Inteligência Artificial-ENIAC, Sao Paulo, Brazil, 9–12.

Seemann T. Prokka: rapid prokaryotic genome annotation. Bioinformatics. 2014 Jul 15;30(14):2068-9. doi: 10.1093/bioinformatics/btu153. Epub 2014 Mar 18. PMID: 24642063.

Siguier P, Perochon J, Lestrade L, Mahillon J, Chandler M. ISfinder: the reference centre for bacterial insertion sequences. (2006) Nucleic Acids Res. 34(Database issue):D32-6. doi: 10.1093/nar/gkj014. PMID: 16381877; PMCID: PMC1347377.

Sommer MO, Church GM, Dantas G.(2010) The human microbiome harbors a diverse reservoir of antibiotic resistance genes. Virulence. 1(4):299-303. doi: 10.4161/viru.1.4.12010. PMID: 21178459.

Schietgat, L.; Vens, C.; Cerri, R.; Fischer, C.N.; Costa, E.; Ramon, J.; Carareto, C.M.A.; Blockeel, H. (2018). A machine learning based framework to identify and classify long terminal repeat retrotransposons. PLoS Comput. Biol., 14, e1006097.

Smit AFA, Hubley R, Green P. RepeatMasker Open-3.0; 2010.

Steinbiss S, Willhoeft U, Gremme G, and Kurtz S.(2009) “Fine-grained annotation and classification of de novo predicted ltr retrotransposons,” Nucleic Acids Research.

Stuart, G, Berry, M.(2004) An SVD-based comparison of nine whole eukaryotic genomes supports a coelomate rather than ecdysozoan lineage. BMC Bioinformatics 5, 204.https://doi.org/10.1186/1471-2105-5-204

Syvanen M.(1984) The evolutionary implications of mobile genetic elements. Annu Rev Genet. 18:271-93. doi: 10.1146/annurev.ge.18.120184.001415. PMID: 6099089.

Traglia, G., Chiem, K., Quinn, B., Fernandez J., Montaña S., Almurzara M., Mussi M., Tolmalsky M., Iriarte A., Centron D., and Ramirez S. (2018)Genome sequence analysis of an extensively drug-resistant Acinetobacter baumannii indigo-pigmented strain depicts evidence of increase genome plasticity. Sci Rep 8, 16961.

UniProt Consortium. UniProt: a hub for protein information. Nucleic Acids Res. 2015 Jan;43(Database issue):D204-12. doi: 10.1093/nar/gku989. Epub 2014 Oct 27. PMID: 25348405; PMCID: PMC4384041.

Van Domselaar GH, Stothard P, Shrivastava S, et al. BASys: a web server for automated bacterial genome annotation. *Nucleic Acids Res*. 2005;33(Web Server issue):W455-W459. doi:10.1093/nar/gki593

Vendrell-Mir, P., Barteri, F., Merenciano, M. et al. A benchmark of transposon insertion detection tools using real data. Mobile DNA 10, 53 (2019). https://doi.org/10.1186/s13100-019-0197-9

Wallace, J., Smith C, and Bretherton C. (1992). "Singular value decomposition of wintertime sea surface temperature and 500-mb height anomalies." Journal of climate 5, no. 6: 561- 576.

Wicker T ,Sabot F, Hua-Van A, Bennetzen JL, Capy P, Chalhoub B, Flavell A., Leroy P, Morgante M, Panaud O, Paux E, SanMiguel P, Schulman A.(2007). “A unified classification system for eukaryotic transposable elements,” Nat Rev Genet.

Wu C, Whitson G, McLarty J, Ermongkonchai A, Chang TC. (1992). Protein classification artificial neural system. Protein Sci.;1(5):667-77. doi: 10.1002/pro.5560010512. PMID: 1304365; PMCID: PMC2142223.

Wu C, Berry M, Shivakumar S, McLarty J.(1995) Neural Networks for Full-Scale Protein Sequence Classification: Sequence Encoding with Singular Value Decomposition. Machine Learning, 21,177-193.

Yang Y, Lu B.-L. Yang W.-Y. (2008) “Classification of protein sequences based on word segmentation methods,” in Proceed- ings of the 6th Asia-Pacific Bioinformatics Conference (APBC ’08), vol. 6, pp. 177–186, Imperial College Press.

Zadrozny, B & Elkan, C. (2001). Obtaining Calibrated Probability Estimates from Decision Trees and Naive Bayesian Classifiers. ICML. 1.

Zadrozny, B., & Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD ’02, 694. https://doi.org/10.1145/775047.775151.

Zhang X, Wang J, Li J, Chen W, Liu C. (2018). CRlncRC: a machine learning-based method for cancer-related long noncoding RNA identification using integrated features. BMC Medical. Genomics 11(S6):120 DOI 10.1186/s12920-018-0436-9.