
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y EN SISTEMAS

# PROGRAMACIÓN GENÉTICA PARA PREDICCIÓN DE CASOS DE INFLUENZA A(H1N1) Y SARS-CoV-2 EN MÉXICO

T E S I S

QUE PARA OPTAR POR EL GRADO DE:
MAESTRA EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:
ERIKA LILIAN CASTILLO GUTIÉRREZ

DIRECTOR DE TESIS:
DRA. KATYA RODRÍGUEZ VÁZQUEZ
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

CIUDAD UNIVERSITARIA, CD. MX.
ABRIL, 2022

# Agradecimientos

Agradezco a la *Universidad Nacional Autónoma de México* y al *Posgrado en Ciencia e Ingeniería de la Computación* por permitirme continuar con mis estudios. Al *Consejo Nacional de Ciencia y Tecnología, (CONACYT)* por el apoyo económico durante la realización del posgrado.

Agradezco a la Dra. Katya Rodríguez Vázquez por su gran apoyo en todo momento, y a mis profesores y sinodales del posgrado, por compartir su conocimiento.

Agradezco a mis padres y hermanos, por su gran amor.

# Índice general

# Índice de figuras

# Índice de tablas

Capítulo 1

# INTRODUCCIÓN

A lo largo del tiempo, han surgido diversas enfermedades que afectan al ser humano, algunas causadas por hongos, bacterias o virus. Aunque la mayoría de ellas son tratables gracias a los avances científicos, la humanidad ha sufrido algunas que resultan mortales e incluso algunas, que por su alto nivel de contagio, se han expandido a nivel mundial, como es el caso de la influenza A(H1N1) que surgió en el año 2009 y la COVID-19 actual, en los cuales se centra el desarrollo de este trabajo.

Una de las características de estos virus, es que se transmiten de persona a persona, como lo hacen los virus gripales estacionales, y con la misma facilidad pues el contagio puede ocurrir por la exposición a las gotículas que las personas enfermas expulsan al toser, estornudar o hablar [12].

Los primeros brotes de la influenza A(H1N1) sucedieron en América del Norte en marzo y abril de 2009 y debido al desconocimiento del virus y al alto nivel de contagio, este se propagó rápidamente por todo el mundo. Cuando la OMS declaró la pandemia en junio de 2009, un total de 74 países y territorios ya habían notificado infecciones confirmadas mediante pruebas de laboratorio [12].

En México la ola epidémica de influenza inició el 11 de marzo y cuatro meses después no habían dejado de aparecer casos. Las principales zonas afectadas fueron las del centro del país, pues el 47 % de los casos confirmados ocurrieron en las entidades Ciudad de México, Estado de México, Hidalgo, Querétaro, Morelos, Tlaxcala y Puebla; con mayor número de defunciones registradas en la Ciudad de México y Estado de México. [4].

En el año 2019 surgió el coronavirus SARS-CoV-2, causante de la enfermedad COVID-19; apareció originalmente en China, propagándose después hacia todos los continentes provocando una pandemia a nivel mundial, declarada oficialmente por la *Organización Mundial de la Salud (OMS)* el 11 de marzo de 2020. Desde su aparición, la gran mayoría de los países han sido afectados y con mucho mayor fuerza que en la pandemia de 2009. Entre los países con mayor número de casos reportados destacan Estados Unidos, India, Brasil, Francia, Italia y Turquía. En México, hasta la fecha existen aproximadamente 3.86 millones de casos oficiales confirmados y 292 mil defunciones, a partir del 8 de Enero de 2020.

Cabe mencionar que las tasas contagio de COVID-19 cada vez son más cercanas a las pandemias del siglo XX, como el caso de la Influenza Española de 1918, en la que se estimó una tasa de morbilidad del 50 % de la población mundial y de 40 a 50 millones de muertes [14]. La facilidad de transmisión de los virus es uno de los factores principales que determinan el número de contagios

en un determinado momento y también debe tenerse en cuenta que estos cambian periódicamente, por lo cual la OMS vigila la existencia de nuevos virus con potencial pandémico.

A pesar de que México cuenta con el Plan Nacional de Preparación y Respuesta ante una pandemia de influenza, debe haber un esfuerzo mayor para comprender y combatir estas enfermedades, pues los impactos negativos que producen además de los altos índices de mortalidad y morbilidad, también son emocionales, psicológicos, sociales y económicos.

Uno de los aspectos principales al estudiar las enfermedades, es la detección de las personas más vulnerables ante los virus. El virus de la influenza A(H1N1) produce mayor mortalidad en personas jóvenes, al contrario a lo que sucede con la influenza estacional, la cual afecta en mayor proporción a personas de la tercera edad y niños pequeños, ya que 65 % de los fallecimientos correspondió a personas hasta los 39 años y 45.1 % se presentó en adultos jóvenes, entre los 20 y 39 años [4]. En el caso de la COVID-19, la mayoría de los casos se presenta en personas jóvenes entre los 20 y los 59 años, con un mayor número de hospitalizaciones de personas entre 50 y 70 años y afectando principalmente a personas con hipertensión, obesidad y diabetes [3].

Por lo anterior, se requiere obtener el mayor conocimiento posible de los virus y su dinámica de transmisión, para poder actuar y tomar mejores decisiones en caso de otra posible pandemia de influenza A(H1N1), COVID-19 u otra que pudiera surgir en el futuro. Una de las tareas más importantes y estudiadas, es la detección temprana de los pacientes infectados, para así detener la propagación del virus entre el resto de la población; esta detección puede realizarse a partir de los síntomas que presentan las personas infectadas.

## 1.1. Planteamiento del problema

Actualmente las pruebas clínicas o de laboratorio, aunque son altamente eficaces en la detección de los virus, pueden requerir de bastante tiempo de análisis o bien de un alto costo económico. Programas computacionales, se diseñan actualmente como apoyo en el sector salud para la detección temprana de diversas enfermedades, clasificando si un paciente está o no enfermo; esto no quiere decir que sustituyan las pruebas médicas, sino que se utilizan como apoyo ya que pueden obtener un resultado de manera más rápida y a un costo bajo, con la ventaja de incluir una mayor probabilidad de certeza a una prueba clínica o bien como ayuda para identificar individuos con mayor riesgo de poseer una enfermedad y aplicar sobre ellos las pruebas clínicas y/o tomar decisiones para evitar el mayor número de contagios posibles.

La programación genética es un método computacional que ha sido aplicado en la detección de enfermedades debido a que tiene mayor grado de expresividad respecto a otros métodos y un alto poder de búsqueda en el espacio de soluciones. De manera general, se busca utilizar programación genética para crear un modelo de clasificación que pueda ser usado para predecir el estado de un paciente (infectado/no-infectado) dado el conjunto de síntomas que presenta.

Aunque ya se están creando modelos de clasificación en otros países, debe tenerse en cuenta que cada región y zona geográfica tiene diversas características (climáticas, de alimentación,

educación, etc) por lo cual un solo modelo puede no ser funcional para todas las regiones del mundo. En este trabajo se pretende encontrar modelos de clasificación de Influenza A(H1N1) y SARS-CoV-2 exclusivamente para algunas regiones de México.

Se creará el modelo a partir del conjunto de datos de pacientes mexicanos que fueron sometidos a pruebas clínicas, de los cuales, se cuenta con los síntomas que presentaron y el resultado de la prueba que indica si poseen o no la enfermedad. Cada modelo, será una expresión en forma de reglas proposicionales, por ejemplo:

**Si (Neumonía=Sí y Tabaquismo=Sí) o (Edad=19-60 o Cefalea=Sí) $\rightarrow$ Enfermo=Sí**

Para los modelos encontrados, debe ser prioridad la detección de los pacientes infectados en comparación con los no infectados. Además, debe tenerse en cuenta que las expresiones encontradas pueden llegar a ser muy grandes, por lo que se busca que las expresiones sean pequeñas, sin perder la efectividad.

Los datos que serán utilizados en el desarrollo del trabajo son los siguientes:

- Influenza A(H1N1): Datos de pacientes, provenientes del Hospital General de México. Consiste de un conjunto de datos binarios en su mayoría, con información de síntomas, sexo, edad y un atributo de clasificacion positivo/negativo de la prueba clínica al virus de influenza A(H1N1).
- SARS-CoV-2: Datos de pacientes, publicados diariamente por la Dirección General de Epidemiología en el portal de datos abiertos del Gobierno de México. Datos categóricos de síntomas, entidad federativa, fecha de incidencia y un atributo de clase resultado de la prueba clínica al virus SARS-CoV-2 . Disponibles en: https://www.gob.mx/salud/documentos/datos-abiertos-152127

## 1.2. Objetivos

Se tiene como objetivo general y principal, encontrar un modelo de clasificación de casos de Influenza A (H1N1) y un modelo de clasificación de casos de SARS-CoV-2, ambos con buenas métricas de desempeño en la evaluación sobre registros de pacientes mexicanos. Dado que el conjunto de datos de COVID-19 es demasiado grande y se trata de un primer acercamiento con este conjunto de datos, se limitó el trabajo a las entidades Baja California Sur, Estado de México y Veracruz.

Como objetivos particulares se tienen:

- Detectar los principales síntomas que poseen las personas infectadas con el virus de influenza A (H1N1).
- Detectar los principales síntomas que poseen las personas infectadas con el virus de SARS-CoV-2.

- Comparar el modelo generado con respecto a otro clasificador, C4.5 (J48) de weka. Las métricas de evaluación utilizadas son, para la detección de casos positivos: la precisión, medida-F ($F_1$) y sensibilidad; para los casos negativos: la especificidad; y de manera general la exactitud.

## 1.3. Estructura de la Tesis

I. **INTRODUCCIÓN**: Se presenta la justificación, el planteamiento del problema, los objetivos generales y particulares, así como el contenido general de la tesis.

II. **CLASIFICACIÓN**: Se describen los conceptos teóricos de minería de datos, como la comprensión del dominio del problema, limpieza y transformación de datos, etc. Además, presentación de la clasificación y sus métricas de evaluación.

III. **PROGRAMACIÓN GENÉTICA**: Descripción de la teoría de Programación Genética y su aplicación para tareas de clasificación con un enfoque basado en gramáticas.

IV. **DESCRIPCIÓN DEL MODELO**: Descripción de los conceptos particulares del modelo en específico para los datos de influenza A(H1N1) y SARS-CoV-2, tales como la gramática específica utilizada, enfoques de programación genética y operadores genéticos utilizados para selección, cruza y mutación.

V. **RESULTADOS**: Presentación y análisis de los modelos obtenidos para cada conjunto de datos. Comparación de resultados de efectividad con respecto a otro método de clasificación: C4.5 (J48 en weka).

VI. **CONCLUSIONES Y TRABAJO FUTURO**: Conclusiones finales, respondiendo los objetivos planteados y trabajo futuro.

Capítulo 2

# CLASIFICACIÓN

Analizar los datos clínicos de los pacientes que se sometieron a pruebas de laboratorio y crear un modelo de predicción a partir de ellos para detectar enfermedades no es una labor sencilla, ya que obtener la información estratégica y correcta de los datos involucra diversas tareas de procesamiento y análisis.

La búsqueda de esta información estratégica generada a partir de los datos, es lo que se llama "Descubrimiento del conocimiento en bases de datos" *(Knowledge Discovery in Databases, KDD)* y se involucran diversos factores para garantizar el éxito de este proceso de búsqueda, el cual debe asegurar que la información obtenida sea integrada, accesible, creíble y a tiempo, con el fin de apoyar la toma de decisiones para un ser humano o un sistema más grande.

Dentro del proceso de búsqueda existe una fase llamada "Minería de datos", en la cual se definen y aplican las técnicas de exploración y análisis para detectar las relaciones que existen entre los datos. Estas técnicas se eligen de acuerdo a los objetivos que se deben alcanzar, pues pueden requerir trabajo de clasificación, predicción, clustering, detección de anomalías, etc. Entre estas técnicas destacan las redes neuronales, la regresión lineal, el cómputo evolutivo, los árboles de decisión, entre otras.

## 2.1. Minería de datos

La minería de datos es el proceso de descubrir patrones válidos, interesantes y novedosos, así como modelos descriptivos y comprensibles, a partir de datos a gran escala [10]. Sin embargo, para alcanzar el éxito en esta fase y poder alcanzar los objetivos, llegar al "Descubrimiento del conocimiento", se debe pasar por una serie de tareas de preparación e interpretación de los datos.

Formalmente, se define el descubrimiento del conocimiento como el proceso general de convertir datos de bajo nivel en conocimiento de alto nivel [13], dicho proceso abarca tareas de preprocesamiento, minado y postprocesamiento de los datos, las cuales se pueden estructurar en 9 etapas [17] que no son necesariamente secuenciales, sino que siguen un orden en el que puede haber recurrencia [11], para realizar retroalimentación y/o ajustes de parámetros y así mejorar los resultados de las tareas posteriores, figura 2.1 :

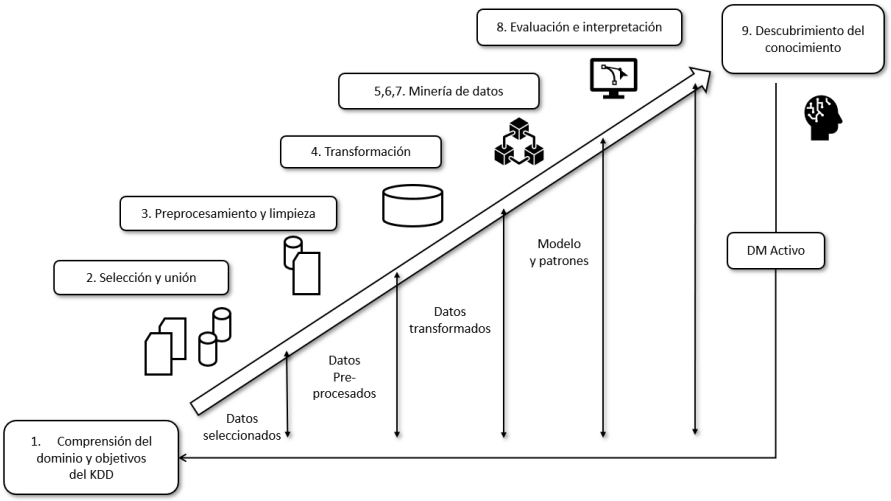



Figura 2.1: Proceso de búsqueda del conocimiento.

1. **Comprensión del dominio y objetivos del KDD:** Se deben comprender y definir las metas y el entorno en que se realizará el proceso, así como incluir el conocimiento previo relevante. Además conforme se desarrolla el proceso puede haber una revisión y ajustes de este paso.

2. **Selección y unión:** Se determinan los datos que serán utilizados, esto incluye encontrar los datos disponibles, la obtención de datos adicionales necesarios y la integración de todos ellos en un conjunto de datos donde se incluyen las características que serán considerados para el proceso.

   Esta etapa es la base para la construcción del modelo, pues se aprende y descubre a partir de los datos disponibles, entonces si los datos con los que se cuenta no son suficientes para alcanzar los objetivos, todo el proceso de búsqueda del conocimiento puede fallar. Por esta razón, es bueno tener en cuenta el mayor número posible de datos en esta etapa, sin embargo, recoger, organizar y operar los repositorios de datos es caro en tiempo y espacio, por lo que es bueno encontrar un equilibrio y hacer una buena selección de datos fuente.

3. **Preprocesamiento y limpieza:** Debido a que los datos fuente normalmente no son de buena calidad, se requiere que se apliquen técnicas para eliminar ruido, manejo de datos perdidos, duplicados, erróneos, incompletos, inconsistentes, irrelevantes para el objetivo y atípicos. Además, se definen aspectos de almacenamiento en el sistema gestor de base de datos (DBMS Data Base Management System), como los metadatos y esquemas utilizados.

   Existen diversas acciones que se toman con respecto a la limpieza de datos, por ejemplo, para el manejo de datos anómalos o también llamados *outliers*, pueden ignorarse, eliminarse, reemplazar por un valor como el promedio o la moda o discretizar el conjunto al que pertenece.

En el caso de los datos incompletos, inconsistentes, erróneos o perdidos, también puede optarse por la eliminación, el reemplazo por un valor común o esperar hasta que los datos estén disponibles, dependiendo del tipo de dato y del significado que tenga dentro del problema, pues no es lo mismo reemplazar la altura de una persona por un valor de estatura promedio, que el número telefónico de un paciente.

4. **Transformación:** Se realiza la generación de las mejores características, o también llamados atributos, que representan al conjunto total de datos. Se utilizan métodos de reducción de dimensiones y transformación de atributos (cambio de tipo, rango y creación de nuevos atributos si es necesario) con el fin de reducir el número efectivo de variables a estudiar o para encontrar representaciones invariantes de los datos.

   También debe tenerse en cuenta que más adelante existe una fase de evaluación, por lo cual es importante seleccionar o dividir el conjunto de datos en dos grupos distintos, uno para realizar la minería de datos y otro para evaluación, generalmente se utilizan los porcentajes 70-30, es decir, 70 % para construcción de un modelo o minería de datos y 30 % para evaluación.

5. **Elección de la tarea de minería adecuada:** Se decide la tarea de minería de datos con base en el propósito del negocio, por ejemplo, clasificación, regresión, reglas de asociación, clustering, etc.

   De manera general existen dos objetivos principales en minería de datos: la predicción y la descripción. La predicción se refiere a la creación, análisis y empleo de datos históricos con el fin de predecir eventos futuros; mientras que la descripción incluye la extracción y análisis de las características más representativas sin algún tipo de hipótesis previa, aplicando técnicas estadísticas, de visualización y agrupación.

   La mayoría de las técnicas de minado se basan en el aprendizaje inductivo, donde se construye el modelo de manera explícita o implícitamente, generalizando un número suficiente de datos de entrenamiento. El supuesto básico del enfoque inductivo es que el modelo entrenado es aplicable a los casos futuros. Esta estrategia también toma en cuenta el nivel de meta-aprendizaje para el conjunto particular de datos disponibles [11].

6. **Elección de algoritmos y técnicas de minado:** Se seleccionan y definen los métodos y sus parámetros para realizar la búsqueda de patrones, se deben tener presentes y comprender las condiciones bajo las cuales funciona cada uno, pues cada uno identifica distintos patrones y trabaja con diferentes métricas.

   La forma de aprendizaje de los algoritmos de minería puede dividirse en 3 categorías [7]:

- Supervisado: El algoritmo trabaja con una serie de ejemplos cuyo valor objetivo se conoce, a partir de ellos crea una función capaz de predecir el valor correspondiente a un objeto de entrada.

- No supervisado: Los valores objetivo de los ejemplos en el conjunto de datos son desconocidos, por lo cual el algoritmo agrupa los ejemplos de acuerdo con la similitud de los valores de sus atributos.

- Semi-supervisado: El algoritmo trabaja con ejemplos donde el valor objetivo es conocido y también con otros donde el valor es desconocido.

Las técnicas más populares son redes neuronales, árboles de decisión, modelos de inferencia bayesiana, algoritmos genéticos, programación genética, entre otros. Debe tenerse en cuenta, que la elección depende de la tarea o tareas finales que se tienen como objetivos, y a su vez, de los tipos de datos que se manejen. Por otro lado, cuando no se sabe exactamente qué es lo que se quiere encontrar o cómo, pueden aplicarse métodos de exploración de datos como k-means, visualización de histogramas, análisis de series de tiempo, entre otros.

7. **Minería de datos:** Se implementa el algoritmo o algoritmos de minado seleccionados, variando los parámetros del mismo y ejecutándolo hasta que se obtengan resultados satisfactorios. En caso de no tener los resultados esperados, es posible redefinir la elección de los algoritmos de minado o incluso, volver a etapas anteriores, pues con cada resultado obtenido, se conoce una perspectiva diferente del conjunto de datos, es imposible conocer o encontrar todos los patrones o información importante con la aplicación de un solo algoritmo, sin embargo, cualquier información que se obtenga del conjunto de datos, es valiosa.

8. **Evaluación e interpretación:** Interpretación y evaluación de los resultados con respecto a las metas. Aquí se puede comprobar la calidad del modelo con el conjunto de datos independiente al que fue utilizado para construir el modelo y comprender el modelo inducido y su utilidad.

   Posiblemente se tenga que reingresar a etapas anteriores o remover patrones irrelevantes, redundantes o que no van de acuerdo a las metas del proyecto. Después, si se requiere, se deben traducir los que fueron seleccionados en términos entendibles para el usuario final del modelo.

9. **Descubrimiento del conocimiento:** El éxito de este paso determina la efectividad del proceso, y consta en la incorporación del conocimiento obtenido en otro sistema, documentarlo o enviarlo a otras secciones de la organización donde corresponda.

   La información estratégica que se obtiene al final del proceso debe tener las siguientes características:

- Integrada: Obtenida a partir de los datos fuente proporcionados.
- Accesible: Podemos ver y comprender fácilmente la información.
- Creíble: La información obtenida no es inconsistente.
- A tiempo: Se tiene accceso a ella cuando se necesita, no después.

## 2.2. Clasificación

Como se había mencionado anteriormente, la minería de datos ayuda a descubrir patrones interesantes, pero debe tenerse en cuenta el tipo de tarea que se va a realizar, una de ellas es la Clasificación, la cual extrae las características de los datos u objetos que cumplan algún tipo de relación para agruparlos en clases, es decir, extrae modelos que describen las clases. Tales modelos, llamados clasificadores, predicen etiquetas de clase categóricas [5].

Los clasificadores, se pueden dividir en 2 categorías con respecto al tipo de conocimiento que se utiliza: los comprensibles por humanos y los de caja negra, los primeros utilizan expresiones que pueden ser fácilmente entendibles por humanos, entre ellos están los árboles de decisión, programación genética y modelos bayesianos. Los de caja negra no son tan fáciles de ser interpretables, sino que requieren de un análisis o conocimiento experto para ser entendibles, entre ellos se encuentran las redes neuronales artificiales.

La mayoría de las técnicas de clasificación utilizan un aprendizaje inductivo, cuyo objetivo principal es la predicción. La clasificación consiste en dos etapas, la primera es la de aprendizaje, el cual generalmente es supervisado y es donde se construye un modelo de clasificación. En esta fase se crea un algoritmo que trata de descubrir una relación predictiva entre los atributos predictores y las clases, produciendo como salida un modelo de clasificación que expresa la relación de predicción descubierta.

La segunda fase utiliza el modelo creado para predecir etiquetas de clase para el conjunto de datos de prueba, con los cuales el clasificador es evaluado, asignando a cada ejemplo del conjunto una clase basada en los valores de los atributos predictores. De manera general, se puede ver el proceso en la figura 2.2.

Formalmente, el problema de clasificación puede definirse de la siguiente forma:

Sea una base de datos con $n$ atributos $A_1, A_2, , A_n$ y una tupla $X$, representada por un vector n-dimensional de atributos: $X = (x_1, x_2, , x_n)$, que representan los valores correspondientes a un registro de la base de datos, se busca encontrar una función $f$ tal que $y = f(X)$ que pueda predecir el valor asociado $y$ para la tupla $X$ ,$y$ llamado atributo etiqueta de clase, el cual es de valores discretos y sin orden [2].

La exactitud de un clasificador en un conjunto de prueba dado, es el porcentaje de tuplas que son clasificadas correctamente por el clasificador, donde la etiqueta de la clase asociada de cada tupla en el conjunto de prueba se compara con la clase resultante del clasificador.

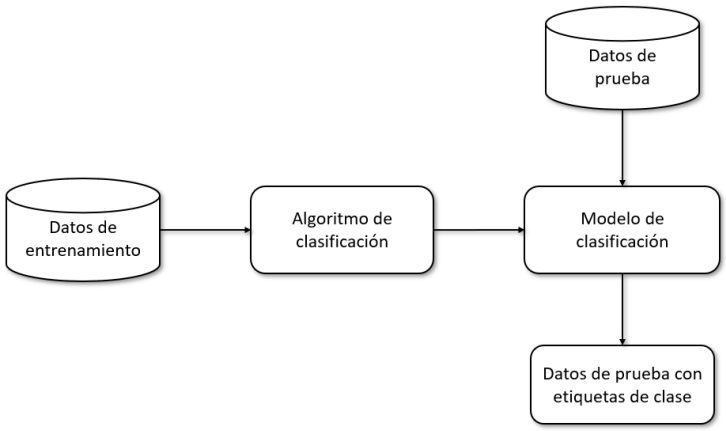



Figura 2.2: Fases de la clasificación.

Hay una enorme variedad de métricas que pueden ser utilizadas para estimar qué tan bueno es un clasificador, una herramienta muy utilizada para medir el desempeño de un algoritmo de aprendizaje supervisado es la matriz de confusión, en la cual se ordenan las clases estimadas por el modelo vs las clases reales de las tuplas $X$.

En el caso más simple donde sólo hay dos clases A y B, la matriz de confusión se ilustra en la figura 2.3, dentro de ella se incluyen los conteos del número de registros por clase que fueron obtenidos por el clasificador tomando en cuenta la clase real a la que pertenecen. Los cuatro posibles casos en los que puede contabilizarse un registro, son los siguientes:

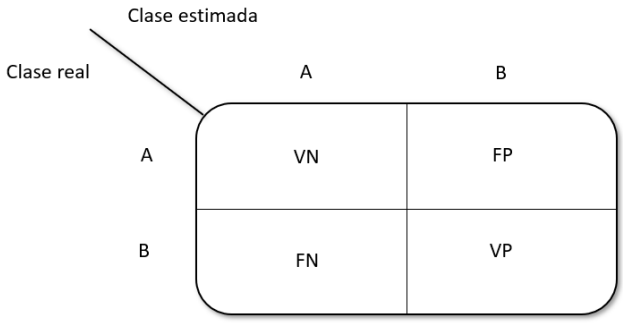



Figura 2.3: Matriz de confusión de dos clases

- VP Verdaderos positivos = Número de instancias de la clase B que el clasificador asignó de clase B.
- FP Falsos positivos = Número de instancias de la clase A que el clasificador asignó de clase

B.

- FN Falsos negativos = Número de instancias de la clase B que el clasificador asignó de clase A.
- VN Verdaderos negativos = Número de instancias de la clase A que el clasificador asignó de clase A.

Dados estos valores, algunas medidas de desempeño son:

- Precisión (*Precision*): De las instancias estimadas como positivas, mide el porcentaje de las que realmente pertenecen a esa clase:

$$Precision = \frac{VP}{VP+FP} \tag{2.1}$$

- Sensibilidad (*Sensitivity*): Mide la capacidad del clasificador para detectar como casos positivos, los casos realmente positivos; pues de las instancias positivas, mide el porcentaje de las que fueron clasificadas como positivas. A esta métrica también se le conoce como exhaustividad:

$$Sensibilidad = \frac{VP}{VP+FN} \tag{2.2}$$

- Especificidad (*Specificity*): Mide la capacidad del clasificador para detectar como casos negativos, los casos realmente negativos:

$$Especificidad = \frac{VN}{VN+FP} \tag{2.3}$$

- Exactitud (*Accuracy*): Se refiere a cuán cerca del valor real se encuentra el valor medido, es decir, qué tan bien el clasificador identifica correctamente o excluye una condición, en otras palabras, qué tan bien detecta un caso como positivo o negativo:

$$Exactitud = \frac{VP+VN}{VP+FP+FN+VN} \tag{2.4}$$

- Medida-F (*F-score*): Es un promedio entre precisión y sensibilidad, el cual es útil y comúnmente utilizado cuando en el conjunto de entrenamiento las clases no están balanceadas, es decir, cuando el número de registros por clase es muy diferente entre clases:

$$F_1 = \frac{2VP}{2VP+FP+FN} \tag{2.5}$$

La elección de las métricas depende del proyecto en que se usen y de su significado dentro de él, pues debe tenerse en cuenta que no todas las clases son de la misma importancia, por ejemplo, en la detección de fraudes por tarjeta de crédito, es mucho más importante detectar los movimientos anormales o fraudulentos que los que no lo son. Para los clasificadores de esta Tesis, es de mucho mayor importancia clasificar correctamente los casos positivos, pues desde el punto de vista de la

pandemia, se busca brindarles ayuda a los pacientes enfermos y aislarlos para prevenir el contagio hacia otras personas no infectadas.

Una característica muy importante a tomar en cuenta al momento de realizar clasificaciones, es el número de registros por clase que existen, pues generalmente los datos vienen con muestras desbalanceadas, o más registros con clase negativa que positiva o viceversa. En estos casos resulta conveniente, a parte de seleccionar las métricas adecuadas, incluir muestreos al momento de generar el algoritmo de clasificación. Dos de las más comunes y simples son:

- Submuestreo (Undersampling): Se selecciona un subconjunto de la clase mayoritaria y todos los registros de la clase minoritaria, donde el subconjunto de la clase mayoritaria se genera de manera aleatoria. En este enfoque, se recomienda realizar el submuestreo varias veces durante el algoritmo de clasificación, para que el modelo sea creado con la mayoría de los datos y no perder información.
- Sobremuestreo (Oversampling): En lugar de tomar un subconjunto de la clase mayoritaria, se duplican los datos de la clase minoritaria hasta que se tiene el mismo número de registros en ambas clases. En algoritmos de aprendizaje supervisado, esta técnica puede crear un sobreajuste en el modelo generado, pues se entrena con datos duplicados.

## 2.3. Inducción de reglas

El conocimiento encontrado por un algoritmo de clasificación, puede ser expresado por un conjunto de reglas, donde el valor que toman los atributos predictores constituyen la primer parte de la regla o lo que se le conoce como el antecedente. La segunda parte de la regla que se define como el consecuente, es el valor del atributo clase. Existen diversos algoritmos que obtienen reglas sobre un conjunto de datos, los más conocidos son los árboles de decisión, como ID3 y C4.5; por otro lado la Programación Genética, que no ha sido tan utilizada como los anteriores.

La principal razón para centrarse en los algoritmos de inducción de reglas es que este tipo de algoritmo tiene la ventaja de descubrir el conocimiento expresado en forma de reglas *if-then* que son intuitivamente comprensibles para los humanos. Más precisamente, los algoritmos de inducción de reglas descubren reglas en la forma *if (condiciones) then (clase predicha)*, donde la regla consecuente predice una clase para cualquier ejemplo, registro o instancia de datos que satisface las condiciones incluidas en el antecedente de la regla [7]. En algunos casos, las reglas crecen a expresiones tipo *if-then-else*, donde el valor de la cláusula *else* corresponde un valor de clase distinto al de la cláusula *then*.

Las reglas pueden ser representadas como expresiones de la lógica proposicional o de la lógica de primer orden. En la lógica proposicional, las reglas pueden ser pares de expresiones modulares de tipo *atributo-valor*, por ejemplo: $(Edad < 20 \bigwedge Cefalea = No) \rightarrow Enfermo = No$ , que es la conjunción de dos pares *atributo-valor* y otro que representa la clase predicha. Se observa que también se involucran operadores, y estos dependen del tipo de dato del atributo. En el ejemplo, el atributo $Edad$ es numérico, por eso se puede usar el operador "$<$", mientras que $Cefalea$ es un

tipo categórico, donde usar el operador ”=“ tiene sentido.

Las reglas lógicas de primer orden son más sofisticadas y tienen mayor poder de expresividad, ya que pueden expresar relaciones entre dos atributos, generando reglas con condiciones tales como $Ingresos > Gastos$ [7].

Las ventajas de utilizar el formalismo de las reglas son:

- Claridad: Son altamente interpretables por humanos.
- Expresividad: Pueden representar cualquier conjunto de instancias y relaciones entre atributos.
- Modularidad: Los segmentos de la regla son distinguibles e independientes.

Los algoritmos de inducción de reglas utilizan diversas métricas para medir su calidad, entre ellas están la precisión, la exactitud, la sensibilidad, especificidad y combinaciones de ellas, aunque la métrica más utilizada es la precisión, llamada confianza si se habla en términos de reglas y no de clasificadores. Se define como sigue:

$$confianza(R) = \frac{p}{p+n}$$

donde $p$ es el número de instancias positivas que cubre la regla $R$, y $n$ el número de instancias negativas que cubre $R$.

## 2.4. Isolation Forest

Una anomalía dentro de un conjunto de datos, se define como un dato que contiene características diferentes al del resto, que son considerados puntos normales. Detectar anomalías o datos atípicos dentro de un conjunto es esencial para mejorar la calidad de los datos, esto no significa que las anomalías sean malas para los datos, pues existen enfoques que se dirigen exclusivamente en su detección para resolver problemas específicos, por ejemplo, detección fraudulenta en el uso de tarjetas bancarias, detección de enfermedades, accesos no autorizados en sistemas computacionales, etc.

Para crear un modelo de clasificación como el que se propone en el presente trabajo, la detección de anomalías juega un papel importante, pues se detectan los registros atípicos que pudieran disminuir la calidad del modelo de clasificación, por ello es importante eliminar estos registros de la muestra y así construir el modelo. Debido a la alta dimensionalidad de los datos y la amplia variedad de tipos que pudiesen tomar los diversos atributos, los algoritmos deben adecuarse para trabajar con las métricas que mejor se adapten al problema y que no presenten un alto costo computacional.

Isolation Forest es un algoritmo de detección de anomalías, propuesto por Fei Tony Liu, Kai Ming Ting y Zhi-Hua Zhou en el año 2008, su enfoque a diferencia de otros, es aislar las anomalías en lugar de buscar patrones que describan las instancias normales. El método, aprovecha las propiedades cuantitativas de las anomalías [8]:

- Son la minoría que consta de menos instancias.
- Los valores de sus atributos son diferentes a los de las instancias normales.

Lo que las hace más suceptible a ser puntos que puedan ser aislados del resto. Además es capaz de trabajar con grandes conjuntos de datos en varias dimensiones y utiliza un enfoque no supervisado.

La manera en que trabaja Isolation Forest es aislar los puntos anómalos a través de particiones recursivas, seleccionando de manera aleatoria un atributo y un valor dentro del dominio del atributo seleccionado con el cual creará la partición. La figura 2.4 obtenida de la referencia [8], muestra de manera gráfica que el punto $x_i$ es más difícil de aislar del resto que $x_0$, es decir, que se requieren más particiones en el conjunto para aislar $x_i$ que $x_0$.

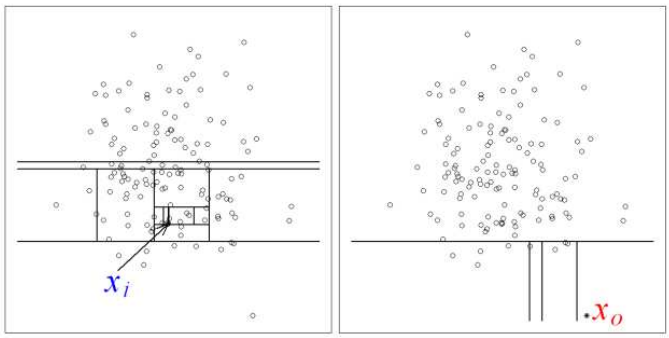



Figura 2.4: Aislamiento de puntos

Cada partición recursiva sobre el conjunto se interpreta como un árbol binario al que se le llama iTree, donde el número de particiones para aislar un punto es equivalente a la longitud del árbol desde la raíz al nodo terminal al que corresponde el punto. De esta manera, las anomalías tenderán a tener menor longitud de camino desde la raíz que los datos difíciles de separar.

Isolation Forest crea múltiples iTrees de manera aleatoria, posteriormente calcula la longitud de camino promedio entre todos los iTrees de la raíz a los nodos terminales, esta longitud de camino, promediada sobre el bosque de iTrees, es una medida de normalidad y la función de decisión de anomalía.

Capítulo 3

# PROGRAMACIÓN GENÉTICA

El objetivo de crear inteligencia artificial procede desde los inicios de la era informática. Científicos de la computación como Alan Turing y John Von Neumann, se motivaron por su visión de crear programas de computadora con inteligencia, con la capacidad realista para auto-replicarse y con la capacidad de adaptación para aprender y controlar sus entornos [9].

En las décadas de los 50s y 60s, algunos científicos de la computación comenzaron a estudiar los sistemas evolutivos, pensando en ellos como una buena solución a problemas de optimización y aprendizaje de máquina, principalmente enfocados a la ingeniería; aunque posteriormente algunos biólogos utilizaron estas técnicas computacionales para simular la evolución de sus sistemas.

Los algoritmos evolutivos, EA(Evolutionary Algorithms), están basados en el proceso de evolución Darwiniana, donde los individuos más aptos son los seleccionados para la supervivencia, lo que aumenta la probabilidad de reproducción y la transmisión de sus rasgos a sus descendientes. Existen diversos paradigmas en EA, pero esta Tesis se enfoca en Programación Genética.

## 3.1. Algoritmos Evolutivos

Los algoritmos evolutivos son algoritmos de búsqueda estocástica, es decir, que el siguiente estado del sistema está determinado tanto por las acciones predecibles del proceso como por elementos aleatorios. El uso de elecciones aleatorias es una herramienta para guiar la búsqueda con probabilidad de mejorar los óptimos encontrados al momento, no se trata simplemente de tomar decisiones aleatorias, sino de guiarlas mediante el azar [2].

Los diversos paradigmas de los algoritmos evolutivos son Estrategias Evolutivas (*Evolutionary Strategies, ES*), Programación Genética (*Genetic Programming, GP*), Evolutionary Programming (*Evolutionary Programming, EP*) y Algoritmos genéticos (*Genetic Algorithms, GA*). La programación genética, cuyo método general se comparte con los algoritmos genéticos, inicia con una población de individuos, cada uno de ellos representados como una posible solución a un problema de optimización. Posteriormente cada individuo es evaluado respecto a una función de "aptitud", la cual mide la calidad de la solución; o visto desde el punto de vista darwiniano, la capacidad de supervivencia de los genes del individuo para sobrevivir y poder ser transmitidos a la siguiente generación.

A lo largo del algoritmo los individuos evolucionan mediante un proceso de selección y combinación para generar mejores individuos, o lo que es lo mismo, mejores soluciones al problema. El algoritmo 1 muestra el pseudocódigo para GA y GP.

**Algoritmo 1** Pseudocódigo para GA y GP

```
1: Crear una población inicial.
2: Evaluar la función de aptitud de cada individuo.
3: repeat
4:    Seleccionar individuos con base en su nivel de aptitud.
5:    Aplicar operadores genéticos a los individuos seleccionados para crear hijos.
6:    Evaluar la función de aptitud de los hijos.
7:    Actualizar la población actual.
8: until (Alcanzar un criterio de paro)
```

Algunas de las ventajas de los algoritmos evolutivos son [6]:

1. Pueden trabajar con valores discretos o continuos.
2. Aptos para trabajar con múltiples variables.
3. Proveen una lista de soluciones al problema planteado.
4. Buenos para ser paralelizados en computadoras paralelas.
5. Se pueden codificar las variables.
6. Realizan la búsqueda de soluciones en espacios muy grandes.

Una vez definido el proceso general, se procede a detallar las características antes mencionadas, como la función de evaluación y los métodos de selección más populares. Los operadores genéticos de cruza y mutación, serán descritos en la sección de programación genética de este capítulo.

### 3.1.1. Función de evaluación

Evaluar la capacidad de supervivencia un individuo depende directamente de su material genético, también llamado genotipo, el cual es la información contenida en su genoma (genes). En algoritmos genéticos, se representa generalmente por una cadena binaria o de números reales. No debe confundirse con el fenotipo, el cual es una característica observable del individuo, en GA representa el significado del genotipo dentro del dominio del problema [6].

Una función de evaluación genera una salida a partir del cromosoma $s$, el cual es un arrelgo de parámetros (genes del individuo). Si éste tiene $N$ variables, es decir, representa una solución a un problema de optimización $N-dimensional$, se puede tomar el cromosoma como un vector renglón:

$$s = [p_1, p_2, \cdots, p_N] \tag{3.1}$$

el cual tiene asociado un costo que se evalúa mediante una función $f$:

$$costo = f(s) = f(p_1, p_2, \cdots, p_N) \tag{3.2}$$

dicho costo, también llamado *aptitud* del individuo, indica qué tan buena es la solución.

Para el problema de clasificación $s$ puede representar los valores de la matriz de confusión obtenidos de la evaluación sobre el conjunto de datos y $f$ alguna o algunas métricas de evaluación de desempeño mencionadas en la sección 2.2. Por ejemplo, si únicamente se toma en cuenta el valor del f-score para medir el desempeño de un clasificador entonces:

$$costo = f(s) = f(VN, FP, FN, VP) = \frac{2VP}{2VP + FP + FN} \tag{3.3}$$

Como se vio en el capítulo anterior, la calidad de un clasificador no depende únicamente de una sola métrica de desempeño, sino que a veces es necesario considerar una combinación de varias de ellas. La forma más sencilla de realizar esto es utilizar una función de agregación lineal, la cual es una suma ponderada de los $k$ objetivos de un problema de optimización, en este caso, las $k$ métricas a considerar, por ejemplo:

$$costo = f(s) = w_1 \times f_1(s) + \cdots + w_k \times f_k(s) \tag{3.4}$$

donde $w_i \in \{1, ..., k\}$ son no negativos, $\Sigma_{i=1}^{k} w_i = 1$ y cada $f_i$ representa una métrica de desempeño del algoritmo de clasificación.

### 3.1.2. Métodos de selección

El objetivo de la selección es hacer hincapié en los individuos más aptos de la población, esperando que sus descendientes tengan mayor aptitud. Algunos de los métodos más utilizados son:

- **Selección de la Ruleta, (Roulette Wheel Selection, RWS):**

  Supone una ruleta circular particionada de tal manera que a cada individuo se le asocia una rebanada, cuyo tamaño es proporcional a la aptitud del individuo [9]. La probabilidad de que un individuo sea seleccionado puede verse de la siguiente manera:

  $$p(i) = \frac{f(i)}{\sum_{j=1}^{N} f(j)} \tag{3.5}$$

  donde $f(i)$ es la función de aptitud del individuo $i$. La ruleta gira $N$ veces, donde $N$ es el número de individuos en la población. Para cada giro, el individuo seleccionado se coloca en la piscina de padres para la siguiente generación. Este procedimiento requiere de $O(N)$ operaciones.

  Este método tiene el riesgo de converger prematuramente a un óptimo local debido a la presencia de un individuo dominante.

- **Muestreo estocástico universal, (Stochastic Universal Sampling, SUS):**

  Variación del RWS propuesta por James Baker (1987), la cual trata de reducir el riesgo de convergencia prematura.

  Selecciona todos los individuos en una sola ejecución, ya que a diferencia de RWS utiliza $N$ apuntadores igualmente separados. Este procedimiento requiere de $O(N)$ operaciones.

- **Torneo:**

  Inicia tomando aleatoriamente un conjunto de $k$ individuos de la población actual, eligiendo entre ellos al más apto para pasar a la siguiente generación. Posteriormente realiza lo mismo $N$ veces para así completar los padres de la siguiente generación. La complejidad de este método es de $O(N)$ y la probabilidad de que cada elemento sea seleccionado está dada por:

$$p(i) = \begin{cases} \frac{C_{N-1}^{k-1}}{C_N^k} & \text{si } i \in [1, N-k-1] \\ 0 & \text{si } i \in [N-k, N] \end{cases} \tag{3.6}$$

- **Elitismo:**

  Complementario a los métodos antes mencionados, propuesto por Kenneth De Jong (1975), forza al algoritmo genético a mantener en la siguiente generación a un número dado de individuos, los cuales deberán ser los mejores, esto para asegurar que la información de los mejores elementos no se pierda en las fases de cruza y mutación.

  Este método presenta una complejidad $O(1)$.

### 3.1.3. VEGA

Una de las partes más difíciles al diseñar un algoritmo genético para clasificación, es definir la función objetivo. En la sección 3.1.1 se mencionó la función de agregación, en la cual se toman en cuenta diversos objetivos, es decir, que requieren la optimización simultánea de más de un objetivo, a este tipo de problemas se les conoce como problemas multiobjetivo. Generalmente, no existe un elemento o un individuo que optimice cada uno de los $k$ objetivos simultáneamente, ya que los objetivos pueden entrar en conflicto, de tal manera que mejorar el desempeño en alguno, empeorará el desempeño en otro y viceversa.

Existen diversos enfoques que ayudan a resolver problemas de optimización multiobjetivo, como son los basados en el óptimo de Pareto y agregación. Sin embargo, aunque los primeros son efectivos, toman mayor uso de recursos computacionales y tiempo para encontrar soluciones. Los métodos de agregación son simples, aunque podrían no encontrar una solución global a un problema.

VEGA, un método propuesto en 1985 por David Schaffer, se enfoca en la resolución a problemas multiobjetivo. Es un algoritmo genético simple, donde se cambia el método de selección de los individuos de la siguiente manera:

Sean $k$ objetivos, se generan $k$ subpoblaciones de tamaño $m$ tal que $P = k \times m$ si $P$ es el número de individuos de la población total. Se eligen los miembros de la subpoblación $k$ dependiendo del valor de aptitud del individuo en $f_k$, tal que los mejores individuos para el objetivo $k$ conforman la subpoblación. Este conjunto de subpoblaciones serán los individuos padres a los cuales se les aplicarán los operadores de cruza y mutación de la manera convencional. La figura 3.1 muestra este comportamiento [16].

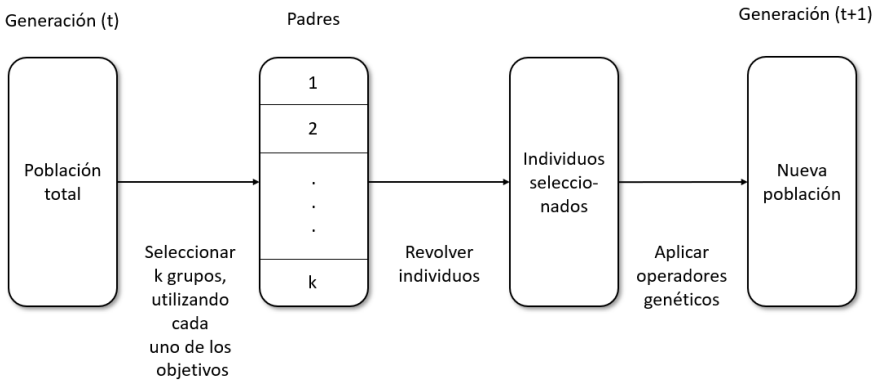



Figura 3.1: VEGA.

Este método, proteje la supervivencia de los mejores individuos por cada objetivo, y simultáneamente proporciona las probabilidades apropiadas para la selección múltiple de individuos que son mejores que el promedio en más de una dimensión [16]. Cabe señalar que generalmente la elección de los individuos utilizados para la cruza y mutación son elegidos de forma aleatoria, aunque se pueden aplicar otras heurísticas para realizar esta selección.

## 3.2. Programación Genética

La programación genética es un método basado en algoritmos evolutivos, cuya meta es evolucionar programas de cómputo. Como se vio en la sección anterior, se manejan poblaciones de individuos que son evaluados y seleccionados para crear una nueva población de individuos más aptos.

Los individuos en la programación genética, son soluciones candidatas que tienen mayor poder de expresividad comparado a otros métodos evolutivos. En este caso, los individuos, o bien llamados soluciones, no solo poseen datos y variables, sino también funciones y operadores, los cuales ayudan a encontrar soluciones genéricas a funciones, en contraste a otros métodos que buscan una solución para una instancia específica de un problema.

Generalmente, los individuos pueden verse como árboles, donde los nodos internos representan funciones y los nodos terminales representan valores fijos o de variables; este tipo de representación es más fácilmente interpretable por los humanos. Otra representación utilizada es la lineal, donde cada individuo se puede ver como un arreglo donde la posición de las variables y funciones depende del tipo de ordenamiento que se maneje: in-orden, pos-orden o pre-orden. La imagen 3.2

muestra la representación en árbol y la representación lineal en ordenamiento in-orden de la misma función $f(x) = x \times x + 1$.

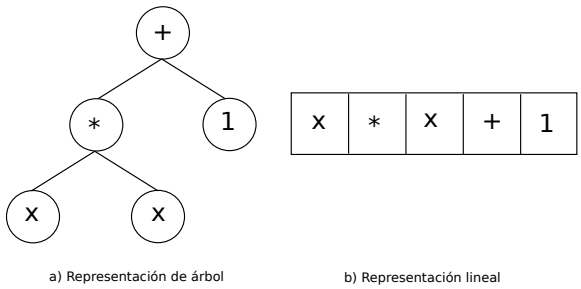



Figura 3.2: Representaciones de individuos más usadas.

Otro aspecto importante a considerar es la forma en que se crea la población inicial, así como en algoritmos genéticos, esta es creada de manera aleatoria. Dos de los métodos más simples son el grow y el full; el full crea árboles completos, donde los nodos terminales están todos en el mismo nivel de profundidad/altura. En el método grow se permiten árboles de estructuras más variadas, donde no importa que se haya alcanzado o no una profundidad/altura máxima para seleccionar nodos terminales.

Los métodos de cruza que se utilizan en la programación genética son muy variados, el que se utiliza en las implementaciones que siguen es la siguiente: dados dos árboles seleccionados para cruza, se eligen dos nodos de cruza aleatorios en los árboles (uno en cada árbol), posterior a esto, cada sub-árbol con raíz en el nodo seleccionado es intercambiado por el sub-árbol del otro padre en la posición del nodo de cruza seleccionado del otro árbol. De manera general, los padres intercambian sub-árboles.

Para la mutación también existen diversas variantes, una de las más utilizadas es la mutación por nodo, en la que dado un nodo no terminal en el árbol, se cambia por otra función, cuyo número de argumentos sea igual a la función en el nodo seleccionado, con una probabilidad de mutación dada. Otro método de mutación ampliamente utilizado es sustitución por rama, donde se elige un nodo con cierta probabilidad y se reemplaza el sub-árbol con raíz en ese nodo por otro árbol generado aleatoriamente a partir de las funciones y terminales permitidas.

De manera general, un algoritmo de programación genética involucra especificar:

1. Un conjunto de funciones y un conjunto de terminales, usados para crear los individuos.
2. Una representación de los individuos.
3. Un método de inicialización de la población.
4. Una función de aptitud, usada para medir la calidad de los individuos.
5. Un método de selección que selecciona individuos basados en su valor de aptitud.
6. Operadores de cruza y mutación, que se usan para seleccionar individuos y generar hijos.

## 3.3. Programación genética gramatical

La programación genética ha sido aplicada satisfactoriamente a distintas áreas de investigación, sin embargo, existen varios problemas que pueden surgir y que deben tenerse en cuenta. Cuando se trabaja la programación genética para crear modelos de clasificación, el tipo de dato de los conjuntos de terminales y funciones generalmente no es uno solo, pues pueden existir datos categóricos y numéricos.

El conjunto de funciones depende del tipo de atributo que se tiene en los datos, involucra operadores aritméticos (e.g. +,-,*, /), funciones trigonométricas (e.g. seno, coseno, tangente) o cualquier función que pueda manejar valores numéricos (e.g $x^2, e^x, \sqrt{x}$), también se pueden utilizar funciones de comparación y de conjunto (==, !=, ¡, ¿, ¡=, ¿=, IN, NOT IN, BETWEEN), al igual que operadores lógicos (and, or, xor) y hasta expresiones condicionales (if-then-else).

Este conjunto de funciones, debe satisfacer al menos dos condiciones, llamadas suficiencia y cerradura. La suficiencia significa que el poder expresivo del conjunto de funciones es lo suficientemente bueno como para poder representar una solución al problema objetivo. La cerradura significa que una función debe poder aceptar, como entrada, cualquier salida producida por cualquier función en el conjunto de funciones, [1].

Para satisfacer la condición de cerradura existen diversos enfoques:

- Convertir todos los tipos de datos de todas las terminales y funciones a un tipo de dato estándar y seguir trabajando con GP convencional.
- Usar Programación Genética Fuertemente Tipada (*Strongly Typed Genetic Programming*).
- Usar Programación Genética Basada en Gramática (*Grammar-Based Genetic Programming*).

El enfoque más utilizado es el basado en gramática, pues además de garantizar la propiedad de cerradura a través de la definición de reglas de producción gramatical, permite al usuario incorporar conocimiento del dominio del problema para guiar la búsqueda. Se divide en diferentes clases de acuerdo a dos criterios [7]:

1. La representación usada por los individuos.
2. El tipo de gramática sobre la cual se basa el sistema.

En el primer caso, se decide entre hacer o no un mapeo entre el fenotipo y el genotipo del individuo. En el segundo caso, se utilizan gramáticas como Gramáticas Lógicas, Gramáticas de Atributos o una de las más utilizadas, las Gramáticas Libres de Contexto, [7].

Una de las dificultades que surgen al trabajar con Programación Genética, es el crecimiento incontrolado de los individuos a lo largo de las generaciones sin mejora en el valor de aptitud, a esto se le conoce como *Bloat*. Este fenómeno está asociado con el incremento en el número de intrones en los individuos, los cuales son segmentos de una cadena cromosómica que no

contribuyen al valor de aptitud, por ejemplo, las cadenas $x = x + 0$ o $y = y * 1$.

El bloat puede ser causado por dos razones, la primera es el valor de aptitud de los individuos y la segunda es la neutralidad del código. El primer caso se da porque para una misma solución, existen más representaciones largas que cortas, lo que genera un sesgo al momento de aplicar la selección de individuos, donde se favorece a las representaciones largas, puesto que hay más. En el segundo caso, se dice que el bloat ofrece protección contra los efectos destructores de los operadores genéticos cuando los individuos crean hijos con aptitud menor a la de sus padres, [7].

Para mitigar el *bloat*, se aplican diversas técnicas:

- Especificar un número máximo en el tamaño de los individuos, limitando el número de nodos y/o altura máxima de los árboles.
- Que la función de aptitud no mida únicamente la calidad, sino también el tamaño, favoreciendo a los individuos más pequeños.
- Diseñar operadores genéticos para mitigar el crecimiento de los individuos, por ejemplo, reemplazar los nodos seleccionados por nodos terminales, de esta manera, el número total de nodos y la altura son menores o iguales a los del árbol original.

Existen diversos enfoques que trabajan con estos dos problemas, como la Programación Genética Basada en Gramática (*Grammar Based Genetic Programming*) y la Programación Genética Gramatical Multiobjetivo (*MGGP Multiobjective Grammatical Genetic Programming*).

### 3.3.1. Programación genética basada en gramática

Las gramáticas pueden ser usadas para crear estructuras que pertenecen a un lenguaje específico y verse como un conjunto de reglas que son capaces de generar una expresión. De manera formal, una gramática es una 4-tupla $G = (\Sigma_N, \Sigma_T, P, S)$ donde:

- $\Sigma_N$ es un conjunto finito de símbolos no terminales.
- $\Sigma_T$ es un conjunto de símbolos terminales ($\Sigma_N \cap \Sigma_T = \emptyset$)
- $P$ es un conjunto finito de producciones.
- $S$ es el símbolo inicial tal que $S \notin (\Sigma_N \cup \Sigma_T$

Cada producción $P$ es una pareja ordenada $(\alpha, \beta)$ con $\alpha = \gamma A \delta$ en la cual $\beta, \gamma$ y $\delta$ son posiblemente vacías en $(\Sigma_N \cup \Sigma_T)^*$ y $A \in \Sigma_N$ o bien $A = S$. Se denota a la pareja $(\alpha, \beta)$ como $\alpha \to \beta$.

La programación genética basada en gramáticas, se divide en diversas clases de acuerdo a dos criterios:

1. La representación de los individuos.
2. Tipo de gramática utilizada.

Típicamente, para programación genética se utilizan gramáticas libres de contexto GLC (Context Free Grammar, CFG), o también llamadas gramáticas tipo 2; las cuales también son utilizadas para establecer las reglas sintácticas de los lenguajes de programación. Estas gramáticas generalmente son utilizadas ya que permiten definir de manera formal la lengua natural por medio de sus reglas, las cuales pueden ser fácilmente interpretadas por los humanos, además de que tienen restricciones necesarias para desarrollar algoritmos eficaces.

En estas gramáticas todas las producciones son de la forma:

$$A \rightarrow \alpha, \alpha \in (\Sigma_N \cup \Sigma_T)^+, A \in \Sigma_N \cup S \tag{3.7}$$

excepto por la posible producción $S \rightarrow \varepsilon$.

Las GLC permiten incorporar conocimiento previo sobre el dominio del problema para guiar la búsqueda del algoritmo, y además, se utilizan como herramienta para garantizar la propiedad de cerradura de las soluciones candidatas [7].

La representación de los individuos puede ser directa o puede existir un mapeo entre el genotipo y el fenotipo. En las representaciones directas, el genotipo del individuo es el árbol que representa la función y no necesita una transformación para poder interpretarse. En el otro caso, el genotipo está codificado generalmente por un arreglo binario o de números enteros, que son generados independientemente de la gramática, en este caso, el mapeo con la gramática se realiza hasta evaluar el valor de aptitud de cada individuo.

En el caso de la representación directa, que es la utilizada en esta Tesis, se requiere un procesamiento específico al generar los individuos de la población inicial y al aplicar los operadores de cruza y mutación. Al generar cada individuo de la población inicial, se siguen las reglas gramaticales dadas por la GLC definida para el problema, hasta que cada nodo terminal del árbol generado contenga un símbolo terminal de la gramática. Por ejemplo, para la gramática siguiente:

$S = \{< expr >\}$,
$\Sigma_N = \{< log >, < bool >, < compBin >, < varBinaria >, < claseBinaria >\}$,
$\Sigma_T = \{!=, ==, and, or, 0, 1, fiebre, tos, escalofrios, cefalea\}$
$P =$

$$\begin{aligned}
< expr > &:= < log >< bool >< bool > \\
< bool > &:= < log >< bool >< bool > | \\
&:= < compBin >< varBinaria >< claseBinaria > | \\
< compBin > &:= != | == \\
< log > &:= and | or \\
< claseBinaria > &:= 0 | 1 \\
< varBinaria > &:= fiebre | tos | escalofrios | cefalea
\end{aligned}$$

donde $\alpha := \beta | \iota$ representa las producciones $\alpha \to \beta$ y $\alpha \to \iota$. Siguiendo las reglas desde el símbolo inicial $S$, se puede generar el individuo de la figura 3.3.

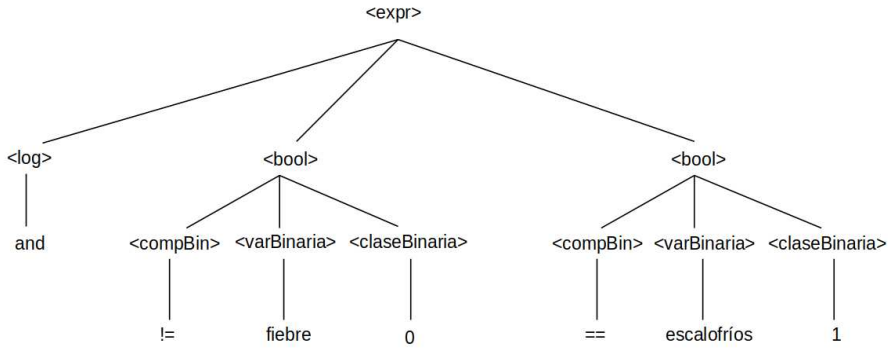



Figura 3.3: individuo generado con gramática.

De manera similar, los operadores de cruza y mutación, deben satisfacer las reglas gramaticales y generar individuos correctos. Para la cruza debe verificarse que el nodo en que se realiza el punto de corte en un árbol padre, satisfaga el tipo de dato que se requiere para ser reemplazado en el otro árbol padre y viceversa. Para la mutación, ya sea por nodo o por rama, se debe cumplir que el árbol o nodo sean del mismo tipo de dato que recibe la función padre del nodo o rama que será reemplazado.

Capítulo 4

# DESCRIPCIÓN DEL MODELO

El descubrimiento del conocimiento, como se definió en el capítulo 2, consiste de diversas tareas de preprocesamiento y análisis, enseguida se describen las primeras 5 etapas para dar solución al problema planteado en este trabajo. Para los dos conjuntos de datos se describe preprocesamiento, limpieza y transformación en el mismo apartado. Las tareas 6 y 7, descritas en la sección 4.3 de este capítulo.

## 4.1. Preprocesamiento Influenza A(H1N1)

1. Comprensión del dominio y objetivos: El planteamiento del problema y los objetivos, definidos en el capítulo 1, detallan la problemática y las metas del presente trabajo. Adicionalmente, el modelo de predicción de casos para este conjunto de datos se divide en 4 sub-modelos, separados por el rango de edad de los pacientes, para garantizar modelos más precisos por edad.

2. Selección y unión: Los datos utilizados son registros de pacientes mexicanos, que fueron sometidos a pruebas de laboratorio para la detección del virus, también se les tomó un registro de los síntomas que presentaban y con ellos se generó una base de datos. A continuación se presentan las características para estos conjuntos de datos:

   **Influenza A(H1N1):** Datos de pacientes, provenientes del Hospital General de México. Consiste de un conjunto de datos de 26 atributos y 5342 registros. Los atributos son los siguientes:

   - IDCASO
   - SEXO
   - FIEBRE
   - TOS
   - ODINOFAGIA
   - DISNEA
   - IRRITABILIDAD
   - DIARREA
   - DOLOR_TORACICO
   - ESCALOFRIOS
   - CEFALEA
   - MIALGIAS
   - ATRALGIAS
   - ATAQUE_EDOGRAL
   - RIORREA
   - POLIPNEA
   - VOMITO
   - DOLOR_ABDOMINAL

- CONJUNTIVITIS
- CIANOSIS
- INIC_SUBITO_SINTOMAS
- CONTACTOOTROSCASOS
- CONTACTO_AVES
- CONTACTO_CERDOS
- ETAPA_POR_EDAD
- RESULTADO_INFLUENZA

El atributo IDcaso es una cadena identificadora del registro y es única por paciente. El atributo etapa_por_edad es numérico, representada con valores enteros entre 1 y 4 dependiendo de la edad del paciente, donde 1 indica el rango de 0 a 6 años, 2 es el rango entre 7 a 15 años, 3 de 16 a 60 años y el 4 es el rango de 60 años en adelante.

El atributo resultado_influenza, es de tipo cadena que indica el resultado de la prueba de laboratorio de detección de Influenza A(H1N1), “neg” es negativo y “POS” es un resultado positivo a la prueba. Para el resto de los atributos, los cuales son sintomáticos, el rango de sus valores es el mismo: 0 es ausente, 1 es presente y -1 indica que el valor se desconoce.

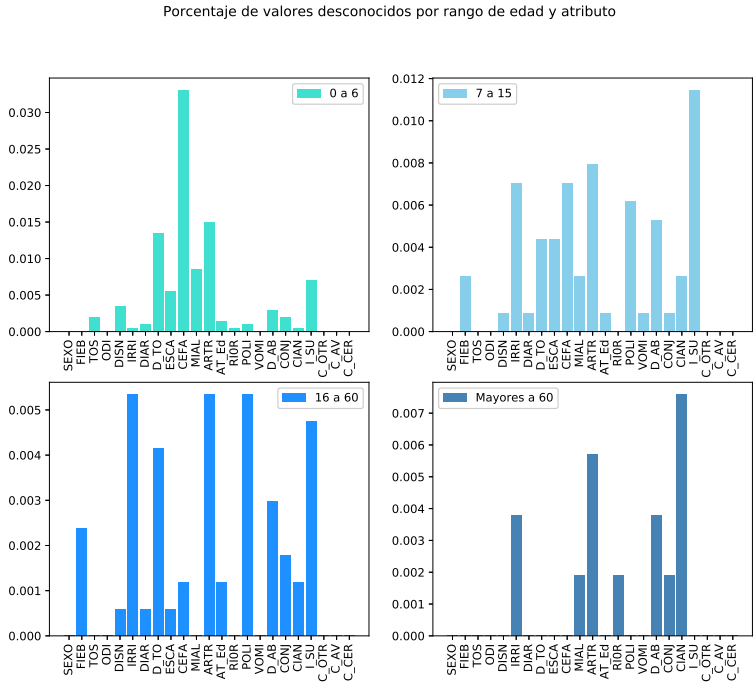



Figura 4.1: Valores desconocidos por etapa_por_edad, Influenza A(H1N1).

3. Preprocesamiento, limpieza y transformación: El conjunto de datos de influenza, debido al tipo de atributos y el rango en que se encuentran sus valores, no requiere de un gran trabajo en esta fase, primeramente, se busca tener únicamente valores binarios en sus atributos, dado que la distribución de los valores desconocidos son muy bajos.

Como primer paso, el atributo resultado_influenza se cambió a tipo binario, donde 0 representa un valor negativo y 1 un valor positivo al virus.

Posteriormente el conjunto de datos se dividió en 4 subconjuntos, uno para cada tipo de valor respecto al atributo etapa_por_edad y se obtuvo el promedio de los valores desconocidos por atributo, figura 4.1. Dado que el porcentaje de valores con -1 es menor al 1 % para la mayoría de los atributos en todos los subconjuntos, se optó por reemplazar el valor desconocido por el valor más común de cada atributo dentro de su conjunto por edad.

## 4.2. Preprocesamiento SARS-CoV-2

1. Comprensión del dominio y objetivos: El planteamiento del problema y los objetivos, definidos en el capítulo 1, detallan la problemática y las metas del presente trabajo.

2. Selección y unión: Datos de pacientes mexicanos que se sometieron a pruebas clínicas para la detección del virus SARS-CoV-2 antes del 23 de marzo de 2021. Los registros publicados diariamente por la Dirección General de Epidemiología en el portal de datos abiertos del Gobierno de México, disponibles en: https://www.gob.mx/salud/documentos/datos-abiertos-152127. A continuación se muestran los atributos de este conjunto de datos:

   **COVID-19:** Conjunto de datos en formato csv, 40 atributos, aproximadamente 6 millones de registros de enero de 2020 al 23 de marzo de 2021:

   - FECHA_ACTUALIZACION
   - ID_REGISTRO
   - ORIGEN
   - SECTOR
   - ENTIDAD_UM
   - SEXO
   - ENTIDAD_NAC
   - ENTIDAD_RES
   - MUNICIPIO_RES
   - TIPO_PACIENTE
   - FECHA_INGRESO
   - FECHA_SINTOMAS
   - FECHA_DEF
   - INTUBADO
   - NEUMONIA
   - EDAD
   - NACIONALIDAD
   - EMBARAZO
   - HABLA_LENGUA_INDIG
   - INDIGENA
   - DIABETES
   - EPOC
   - ASMA
   - INMUSUPR
   - HIPERTENSION
   - OTRA_COM
   - CARDIOVASCULAR
   - OBESIDAD
   - RENAL_CRONICA
   - TABAQUISMO
   - OTRO_CASO
   - TOMA_MUESTRA_LAB
   - RESULTADO_LAB

- TOMA_MUESTRA_ANTIGENO
- RESULTADO_ANTIGENO
- CLASIFICACION_FINAL
- MIGRANTE
- PAIS_NACIONALIDAD
- PAIS_ORIGEN
- UCI

De los atributos anteriores, se seleccionaron los relacionados a enfermedades, como son NEUMONIA, DIABETES, EPOC, ASMA, INMUSUPR, HIPERTENSION, OTRA_COM, CARDIOVASCULAR, OBESIDAD, RENAL_CRONICA, TABAQUISMO, OTRO_CASO y UCI, cuyo rango de atributo es 1:si, 2:no, 97: no aplica, 98: se ignora, 99: no especificado.

También se seleccionaron características de los pacientes, como son ENTIDAD_UM, SEXO, EDAD y EMBARAZO. La ENTIDAD_UM corresponde a la entidad federativa de la Unidad Médica a la que un paciente acudió. Los valores para SEXO son 1:mujer, 2:hombre y 99:no especificado. Para el atributo EMBARAZO, el rango de valores es 1:si, 2:no, 97: no aplica, 98: se ignora, 99: no especificado. La EDAD es un valor entero positivo que representa la edad en años del paciente.

Como atributo etiqueta o de clase, se seleccionó el atributo CLASIFICACION_FINAL, que contiene el valor final determinado por una prueba de laboratorio o por el comité de dictaminación de COVID-19; el rango de valores son (1,2 o 3):positivo, (4 o 7):negativo y (5 o 6) sin resultado.

Por otro lado, aunque se tienen registros de pacientes extranjeros, se tomaron en cuenta únicamente los de nacionalidad mexicana, esto debido a que pudiesen tener distintos hábitos alimenticios respecto a la población mexicana y/o aspectos de vida que pudiesen influir en el comportamiento del virus dentro de su sistema inmune.

3. Preprocesamiento, limpieza y transformación: Los datos de este conjunto, una vez descargados de la plataforma, fueron trabajados en varias etapas.

   En la primera, se realizó la lectura y preprocesamiento del archivo por partes, en la cual se filtraron los pacientes mexicanos con resultado positivo o negativo (valores 1, 2, 3, 4 o 7 en CLASIFICACION_FINAL), los casos en que no se tiene resultado fueron descartados del conjunto, pertenecientes a pacientes que estaban en espera de un resultado positivo o negativo. Además se redujo el rango del atributo CLASIFICACION_FINAL a 2 valores:
   ´

$$CLASIFICACION\_FINAL_{final}(r_i) = \begin{cases} 1 & \text{si } CLASIFICACION\_FINAL(r_i) = [1,2,3] \\ 2 & \text{si } CLASIFICACION\_FINAL(r_i) = [4,7] \end{cases}$$

   Por último en esta fase, se dividió el conjunto de datos por ENTIDAD_UM, para así generar más adelante un modelo por entidad federativa.

La segunda fase, consistió en la aplicación de Isolation Forest a cada uno de los conjuntos por entidad, con el fin de eliminar registros atípicos que pudieran influir en la calidad de los datos y por consecuencia, en el modelo. La implementación de Isolation Forest utilizada, fue la del módulo sklearn de python, considerando un porcentaje de contaminación de 0.1.

A continuación se muestra el conteo de las anomalías encontradas, organizadas por atributo:

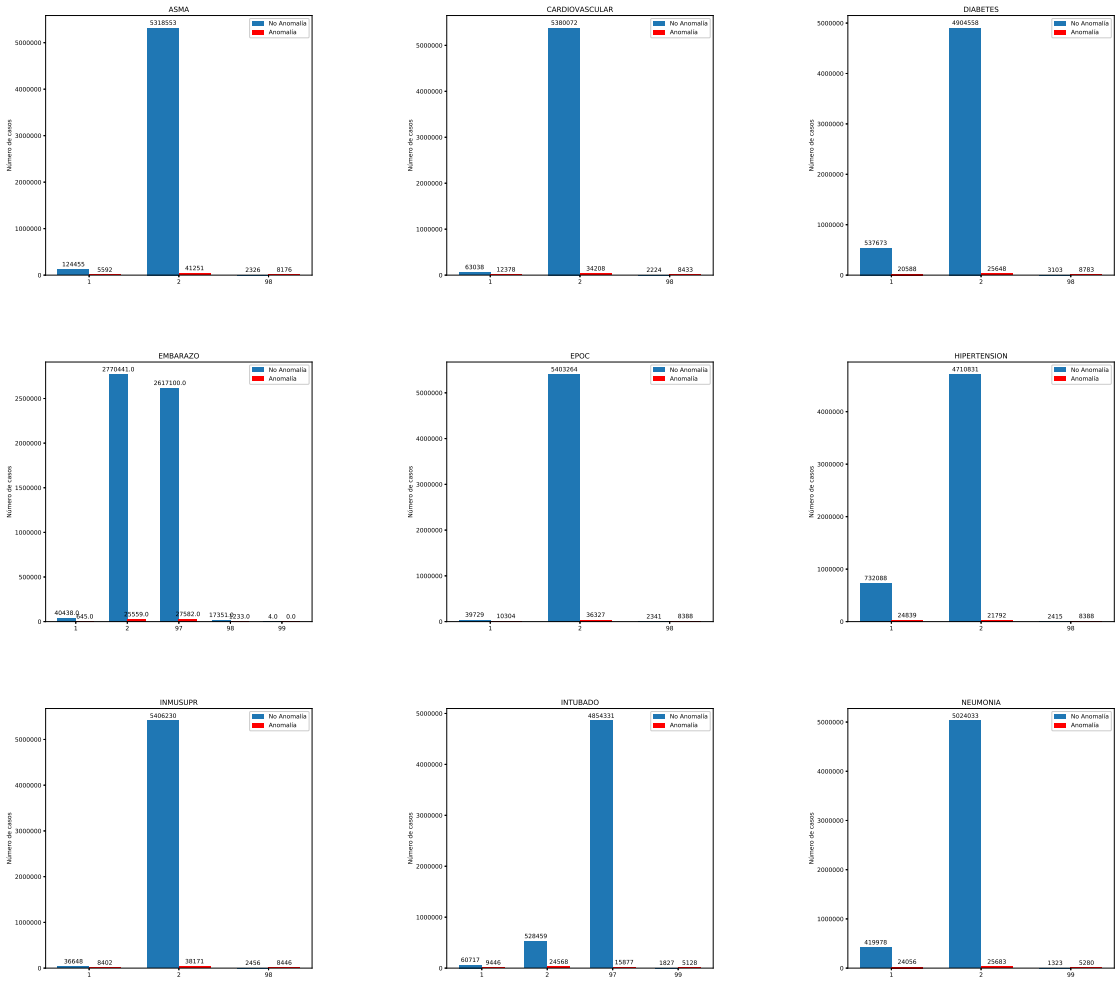



Figura 4.2: Anomalías en COVID-19 con Isolation Forest, parte 1.

Puede observarse, que el número de anomalías encontrado es muy bajo con respecto a la cantidad de registros en el conjunto total de pacientes, además, para el caso de los atributos que representan enfermedades, el número de anomalías encontradas para los valores desconocidos o no especificados, valores 98 y 99 respectivamente, es mayor en porcentaje al de los valores conocidos con respecto al número de registros por valor. Es decir, se redujo la cantidad de valores desconocidos en el conjunto de datos.

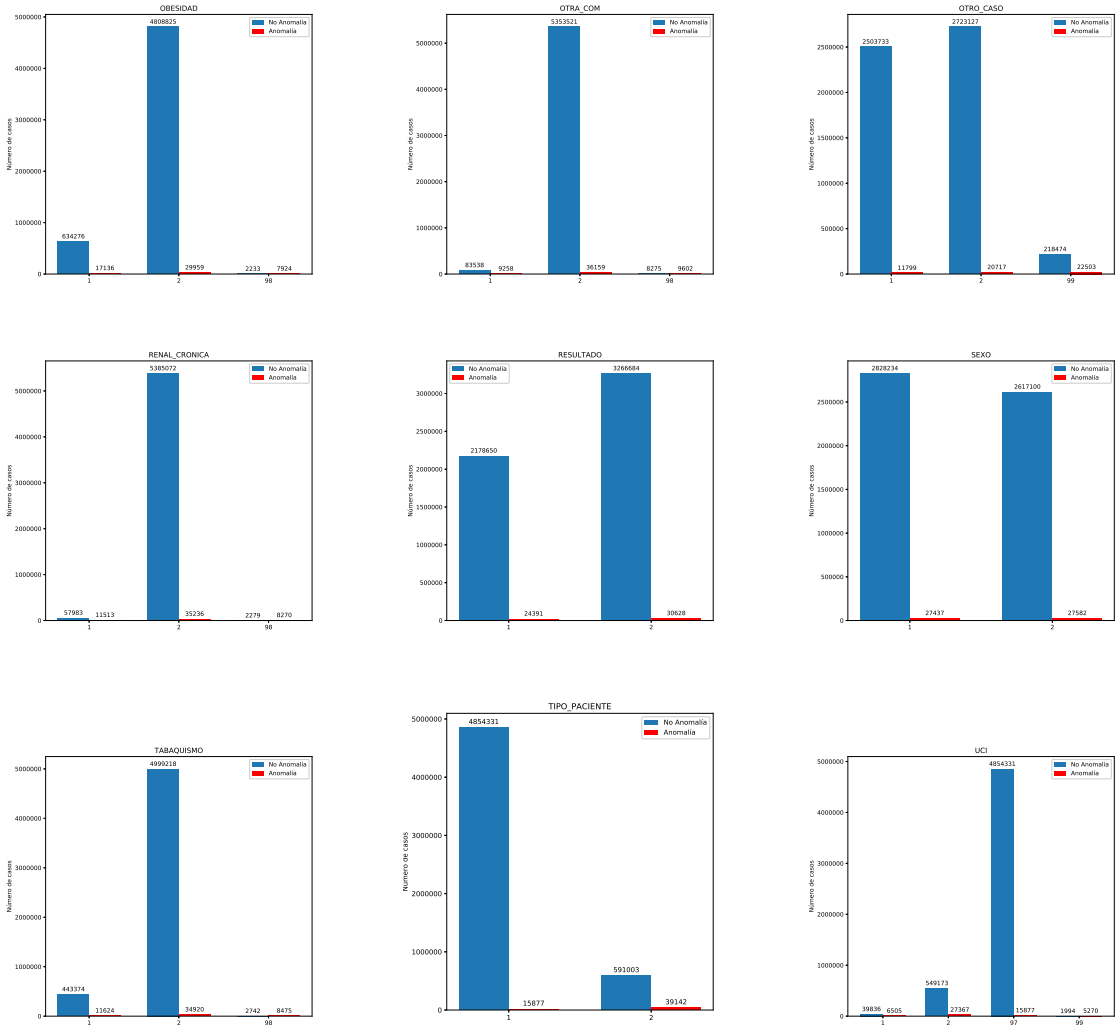



Figura 4.3: Anomalías en COVID-19 con Isolation Forest, parte 2.

## 4.3. Algoritmo de clasificación

De manera general, el desarrollo de los modelos consiste en dos etapas, la primera es la de aprendizaje, el cual es supervisado y es donde se construye un modelo de clasificación a partir del conjunto de datos de entrenamiento, uno para la influenza A (H1N1) y otro para SARS-CoV-2. En esta fase se crea un algoritmo de clasificación, el cual trata de descubrir una relación predictiva entre los atributos predictores y las clases, produciendo como salida un modelo de clasificación que expresa la relación de predicción descubierta.

La segunda fase utiliza el modelo creado para predecir etiquetas de clase para el conjunto de datos de prueba, con los cuales el clasificador es evaluado, asignando a cada ejemplo del conjunto una

clase basada en los valores de los atributos predictores, tal como se vio en el capítulo 2, figura 2.2. Los datos fueron divididos de manera aleatoria, con un 70% para entrenamiento y 30% de prueba para realizar la evaluación.

Para medir qué tan bueno es el clasificador, se consideraron las métricas de precisión, f-score y sensibilidad para la clase positiva, ya que es la clase más importante del modelo y representan lo bueno que el modelo es capaz de detectar los casos positivos. No debe perderse de vista, que la clase negativa también importa, valores en especificidad muy bajos tampoco son tan convenientes, pues un valor muy bajo o cercano a 0 en especificidad indica que el modelo clasifica todas las instancias como positivas, por tanto, la especificidad también es considerada aunque con menor importancia que las anteriores. Además de esto, la métrica que es ampliamente utilizada y que da un panorama general es la exactitud, por lo que también es considerada para la evaluación en conjunto con las anteriores.

La programación genética toma parte en la construcción del algoritmo de clasificación, la figura 4.4 muestra el proceso general, en el cual se aprecia cada una de las fases del algoritmo genético, desde la creación de la población inical, sus operadores genéticos y el término del algoritmo, que representa el modelo de clasificación. Por otro lado, se puede ver que se utiliza una gramática en la generación de la población inicial y los operadores genéticos de mutación y cruza, la justificación resulta de la estructura de los individuos que sigue a continuación.

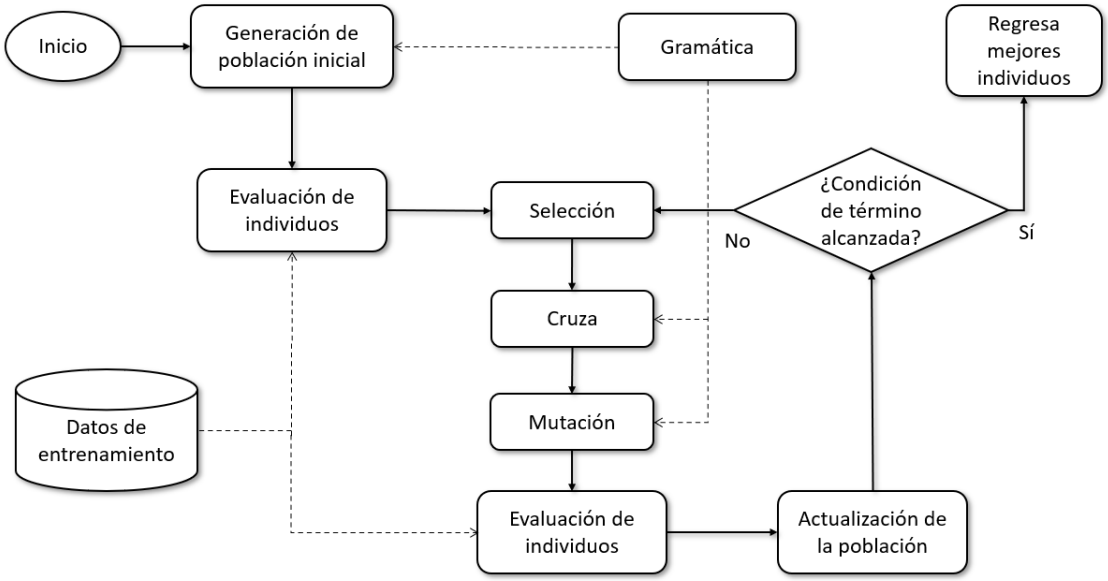



Figura 4.4: Algoritmo de Clasificación

Para el conjunto de datos de influenza, cada individuo del algoritmo, corresponde a una regla de tipo *if (condiciones) then (clase A)* donde las condiciones son pares *atributo-valor*, que a su vez pueden ser reducidas a reglas *if (condiciones) then 1*, lo que indica que cualquier paciente que satisfaga las

condiciones será un paciente positivo al virus de la Influenza A(H1N1), como ejemplo la siguiente regla:

```
 IF ((MIALGIAS != 0) | (CONJUNTIVITIS == 1)) &
    ((ARTRALGIAS != 0) | ((POLIPNEA != 1) & (FIEBRE != 0)))
THEN 1
```

Considerando que todas las reglas serán formadas con ese tipo de estructura y basándose en la gramática general propuesta por [15], la gramática de la influenza quedó definida mediante las siguientes reglas:

$$
\begin{aligned}
< expr > &:= < log >< bool >< bool > | \\
&:= < claseBinaria > \\
< bool > &:= < log >< bool >< bool > | \\
&:= < compBin >< varBinaria >< claseBinaria > | \\
< compBin > &:= != | == \\
< log > &:= and|or|xor \\
< claseBinaria > &:= 0|1 \\
< varBinaria > &:= sexo|fiebre|tos|odinofagia|disnea|irritabilidad| \\
&:= diarrea|dolor\_toracico|escalofrios|cefalea| \\
&:= mialgias|artralgias|ataque\_edogral|riorrea|polipnea| \\
&:= vomito|dolor\_abdominal|conjuntivitis|cianosis| \\
&:= inic\_subito\_sintomas|contactootroscasos \\
&:= contacto\_aves|contacto\_cerdos
\end{aligned}
$$

En el caso del SARS-CoV-2 (COVID-19), las primeras pruebas que se realizaron fueron utilizando una gramática similar a la anterior, donde solo se consideraban reglas *if (condiciones) then 1*, sin embargo, los resultados no fueron satisfactorios. Probando con reglas tipo *if (condiciones) then (clase A) else (clase B)*, se mejoraba un poco el desempeño, pero tampoco tan bueno, por lo que tomando nuevamente en cuenta la gramática en [15], se consideró otro nivel de evaluación de condiciones dentro de las cláusulas *then* y *else*, generando expresiones de la forma:

```
IF (EPOC <= 99) | (INTUBADO < 97)
THEN IF UCI >= 2
     THEN 1  ELSE 2
ELSE IF (EDAD <= 15) | ((INTUBADO > 98) & (EPOC < 98))
     THEN 2 ELSE 1
```

Cabe mencionar que las reglas generadas por las gramáticas que se presentan en este trabajo, contienen una estructura en pre-orden, es decir, los operadores van a la izquierda de los operandos y no generan directamente una regla como la anterior, sino que pasan por una transformación hacia una estructura in-orden, donde los operadores se encuentran al centro y la cual es mucho más fácil

de leer y comprender por las personas.

Dicho lo anterior, la gramática utilizada para COVID-19 quedó definida de la siguiente manera:

$$
\begin{aligned}
< expr > &:= if - else < bool >< expr1 >< expr1 > \\
< expr1 > &:= if - else < bool >< resultado >< resultado > \\
< bool > &:=< log >< bool >< bool > | \\
&:=< compCat >< varSintoma >< sintomaClase > | \\
&:=< compCat > sexo < sexoClase > | \\
&:=< compNum > edad < edadInt > \\
< compCat > &:=< | <= | == | >= | > | != \\
< compNum > &:=<= | >= \\
< log > &:= and|or|xor \\
< resultado > &:= 1|2 \\
< varSintoma > &:= intubado|neumonia|embarazo|diabetes| \\
&:= epoc|asma|inmusupr|hipertension \\
&:= otras\_com|cardiovascular|obesidad \\
&:= renal\_cronica|tabaquismo|otro\_caso|uci \\
< sintomaClase > &:= 1|2|97|98|99 \\
< sexoClase > &:= 1|2|99 \\
< edadInt > &:= 0|1|2|3|4|...|118|119|120
\end{aligned}
$$

### 4.3.1. Algoritmos implementados

Evaluar el valor de aptitud de un individuo, es una de las características de un algoritmo genético que puede no resultar sencillo al tratar problemas de clasificación como los del presente trabajo. Deben considerarse los enfoques que existen, principalmente si se trata de problemas multiobjetivos con una gran cantidad de datos.

Los métodos que se eligieron para resolver el problema de clasificación para Influenza y COVID-19, son 3:

1. GP Simple: Sigue el enfoque de programación genética optimizando un solo objetivo, el cual es el valor del $f - score$ para la clase positiva al virus, es decir:

   $$f(x) = \frac{2VP}{2VP+FP+FN}$$

   El método de selección utilizado es selección de la ruleta con elitismo, en el cual el mejor individuo de la población pasa directamente a la siguiente generación. Se utiliza la cruza aleatoria en un punto, restringiendo los nodos participantes a únicamente nodos

que no son hoja (vista la expresión como un árbol). La mutación es por rama, y para prevenir el crecimiento en el tamaño de las expresiones, se restringe el tamaño de las nuevas expresiones a una de longitud no mayor a la que va a ser reemplazada. De manera similar, en la inicialización de los individuos se colocó una altura límite en el tamaño del árbol.

2. GP Agregación: Similar al método anterior en los operadores genéticos de selección, cruza y mutación; al igual que al momento de inicializar la población de individuos. La diferencia con respecto a GP Simple, es la función de aptitud de los individuos, pues se eligió el enfoque de una función de agregación, tomando en cuenta 5 objetivos principales: precisión, f-score, exactitud, sensibilidad y especificidad:

$$f(x) = w_1 \times precision + w_2 \times fscore \\ + w_3 \times exactitud + w_4 \times sensibilidad \times w_5 * especificidad$$

donde $\sum_{i=1}^{5} w_i = 1$, y los valores de cada una de las funciones objetivo se encuentran en el intervalo [0,1].

3. GP VEGA: Con los mismos métodos de cruza y mutación que los algoritmos anteriores. Como se vio en el capítulo anterior, la selección en VEGA se realiza primeramente particionando aleatoriamente la población en cada generación, donde el número de particiones es igual al número de objetivos (5), pues se utilizan los mismos que en el caso anterior: precisión, f-score, exactitud, sensibilidad y especificidad. Posteriormente se seleccionan los mejores individuos por objetivo y menor tamaño para generar a los padres, que una vez completos, se seleccionan de forma aleatoria para formar la siguiente generación.

   Cabe mencionar que el número de generaciones utilizado en este enfoque no es tan grande, pues las pruebas iniciales que se realizaron con un mayor número de iteraciones, mostraron que los individuos se especializaban demasiado en el objetivo con el cual fueron evolucionando, mientras que su valor en los otros objetivos tendía a valores muy bajos, lo que no mantenía un equilibrio entre todos los objetivos.

Los métodos mencionados, tienen la ventaja de ser fáciles de implementar y no consumen tantos recursos computacionales a diferencia de otros métodos evolutivos multiobjetivo, pues no hay que olvidar que el conjunto de datos, principalmente SARS-CoV-2 es demasiado grande. Además, en la implementación de estos métodos se utilizó una paralelización con un enfoque maestro-esclavo para la evaluación de la función objetivo de los individuos, con el fin de reducir el tiempo de procesamiento del algoritmo.

Las pruebas para el caso de la influenza A(H1N1) fueron realizadas en una computadora con 4 CPU y 12 GB de memoria RAM. Para el caso de la COVID-19, se utilizaron recursos del cluster del IIMAS (UNAM), en específico: 20 CPU y 128GB de memoria RAM.

Capítulo 5

# RESULTADOS

A continuación se presentan los resultados de las ejecuciones para los dos conjuntos de datos, Influenza A(H1N1) y SARS-CoV-2. Cada uno de ellos fue analizado con los métodos de programación genética descritos en el capítulo anterior y diversos parámetros en cada uno. Los mejores valores encontrados, de acuerdo al resultado obtenido, se ilustran dentro de cada sección.

**GP Simple**, indica el algoritmo de programación genética base, cuya función de aptitud es el valor del F-score. **GP Agregación** cambia la función de aptitud por una de agregación que contempla 5 objetivos: precisión, f-score, exactitud, sensibilidad y especificidad; en cada sección se especifican los pesos utilizados. **GP VEGA** vuelve a contemplar los 5 objetivos mencionados, pero visto desde el enfoque de VEGA. J48 de weka, se refiere al algoritmo C4.5.

Los operadores que aparecen en las reglas de los modelos obtenidos, no son exactamente iguales a los de la gramática del capítulo anterior, aunque sí tienen el mismo significado, se utilizaron así debido al lenguaje de programación en que fueron implementados: python 3.8. La correspondencia es la siguiente:

- $and \rightarrow$ &
- $or \rightarrow |$
- $xor \rightarrow \wedge$

Además, se omitieron las palabras THEN y ELSE de las expresiones IF anidadas, por simplicidad en la implementación.

## 5.1. Influenza A(H1N1)

En cada conjunto de datos, el número de las mejores reglas que se tomaron para evaluar cada conjunto de datos de prueba fue dependiente del algoritmo, para el caso del GP Simple se seleccionó 1 regla, para el caso de Agregación fueron 3 y para VEGA fueron 6. El número de reglas utilizadas fue obtenido realizando pruebas combinando desde 1 hasta las 10 mejores reglas, los valores obtenidos fueron los mostrados en este trabajo; cabe señalar que en los casos de Agregación y VEGA, se trata de un problema multiobjetivo, por lo que es difícil que una sola regla contenga toda la información valiosa. Y de manera contraria, agregar demasiadas reglas genera un

modelo donde todos los pacientes son clasificados de una misma clase (este comportamiento fue observado durante las pruebas realizadas).

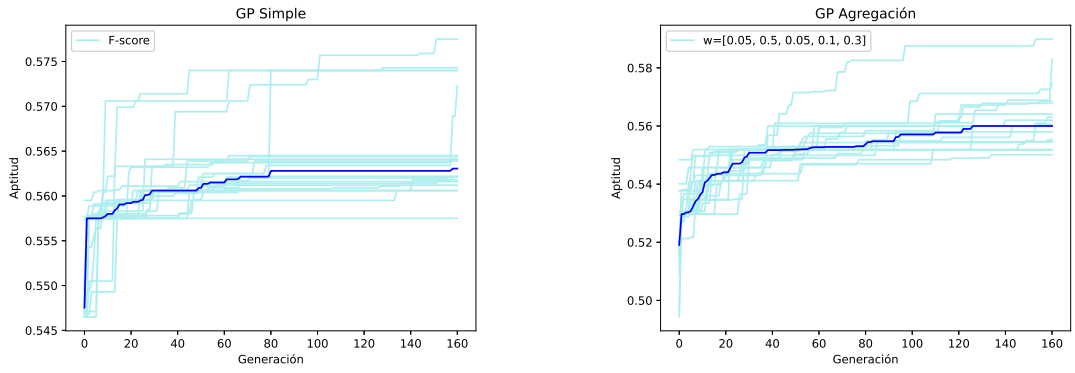
Las reglas seleccionadas se unieron mediante un *or* para así formar un solo modelo de clasificación por algoritmo. También se realizaron ejecuciones con otros operadores como el *and* y el *xor*, pero unirlos con *or* arrojó mejores resultados. El número de ejecuciones para cada conjunto de datos fue de 30, el mejor modelo obtenido de todas las ejecuciones, se eligió dependiendo principalmente del valor de f-score y de la sensibilidad.

### 5.1.1. Pacientes 0-6 años

Para el grupo de pacientes de 0 a 6 años de edad, los mejores resultados se obtuvieron con los parámetros descritos en la tabla 5.1. En el algoritmo de GP Agregación, los pesos utilizados son $w = [.05, .5, .05, .1, .3]$ para las métricas de $[precision, f-score, exactitud, sensibilidad, especificidad]$ respectivamente. Para el caso de VEGA, los 5 objetivos son iguales que los 5 para GP Agregación y las reglas utilizadas para la evaluación sobre el conjunto de prueba son las dos mejores para f-score, más las dos mejores para exactitud y las dos mejores para especificidad.

| Algoritmo | N. Individ. | N. Gener. | P. Mutación | P. Cruza | F. Aptitud | Selecc. |
|---|---|---|---|---|---|---|
| GP Simple | 80 | 160 | 0.2 | 0.8 | F-score | Ruleta |
| GP Agregación | 80 | 160 | 0.2 | 0.8 | .05,.5,.05,.1,.3 | Ruleta |
| GP VEGA | 60 | 80 | 0.25 | 0.8 | 5 objetivos | Elitismo |

Tabla 5.1: Parámetros en algoritmos para pacientes de 0 a 6 años.




(a) (b)

Figura 5.1: Función de aptitud para pacientes de 0 a 6 años.

La figura 5.1, muestra el valor de aptitud alcanzada en la fase de entrenamiento para los casos de GP Simple y GP Agregación. Se observa que existe un mayor crecimiento en las generaciones menores a la 50 y a partir de la 80 comienza a haber un incremento menor. La tabla 5.2 muestra

los valores de las métricas principales para los mejores individuos encontrados en cada uno de los algoritmos con programación genética y weka.

Puede verse que el valor de la precisión es mejor en weka que los demás clasificadores, aunque la diferencia no es tan grande, además, se sabe que un alto valor en precisión no garantiza que un clasificador sea mejor, sino que deben analizarse otras métricas. Por otro lado, el valor de f-score, que es la métrica más importante, es mayor para todos algoritmos de programación genética, lo que indica una mejor clasificación de los casos positivos en estos métodos.

El valor de la exactitud en weka es mayor, pero no debe perderse de vista el número de instancias por clase que hay, pues existe un desbalanceo en el que predomina la clase negativa, esto también se refleja en el valor de la especificidad, sin embargo, no son las instancias negativas tan importantes como las positivas y el valor de la exahustividad para los modelos de programación genética descritos en este trabajo, es mucho mayor que el valor en weka.

| Algoritmo | Precisión | Fscore | Exactitud | Sensibilidad | Especificidad |
|---|---|---|---|---|---|
| GP Simple | 0.4070 | 0.5588 | 0.4841 | 0.8914 | 0.2447 |
| GP Agregación | 0.4328 | 0.5585 | 0.5442 | 0.7873 | 0.4 |
| GP VEGA | 0.4542 | 0.5020 | 0.5926 | 0.5610 | 0.6078 |
| weka J48 | 0.477 | 0.354 | 0.6227 | 0.282 | 0.821 |

Tabla 5.2: Evaluación sobre el conjunto de prueba para pacientes de 0 a 6 años

La figura 5.2 muestra la evaluación sobre el conjunto de prueba del algoritmo J48 de weka, se observa que el árbol generado tiene 79 hojas y es de tamaño 157. Por otro lado, las reglas seleccionadas como las mejores para los algoritmos implementados se muestran a continuación. Los modelos obtenidos por GP Simple y GP VEGA tienen un tamaño menor:

```
Number of Leaves  :     79

Size of the tree :     157


Time taken to build model: 0.1 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.02 seconds

=== Summary ===

Correctly Classified Instances         373               62.2705 %
Incorrectly Classified Instances       226               37.7295 %
Kappa statistic                          0.112
Mean absolute error                      0.4404
Root mean squared error                  0.5128
Relative absolute error                 94.2538 %
Root relative squared error            106.3578 %
Total Number of Instances              599

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.821    0.718    0.663      0.821    0.733      0.120    0.568     0.660     0
               0.282    0.179    0.477      0.282    0.354      0.120    0.568     0.430     1
Weighted Avg.  0.623    0.520    0.595      0.623    0.594      0.120    0.568     0.576

=== Confusion Matrix ===

   a   b   <-- classified as
 311  68 |   a = 0
 158  62 |   b = 1
```

Figura 5.2: Evaluación J48 en weka, pacientes de 0 a 6 años.

```
GP Simple, Matriz de confusión [93, 286, 23, 197] :

(FIEBRE == 1) & ((((POLIPNEA != 0) |
(((((RIORREA != 0) | (ESCALOFRIOS != 1)) |
(CONTACTO_AVES != 0)) ^ (ESCALOFRIOS != 1)) ^
(POLIPNEA != 0) ))  ^ (CONTACTO_CERDOS == 0)) |
(RIORREA != 0))


GP Agregación, Matriz de confusión [152, 227, 46, 174] :

 (MIALGIAS == 1) | ((MIALGIAS == 1) |
 ((CONJUNTIVITIS != 0) |
 ((CONTACTOOTROSCASOS == 1) |
 (((ODINOFAGIA == 1) & (MIALGIAS == 1)) &
 ((CONTACTOOTROSCASOS == 1) & (DISNEA == 0))))))
 or

((MIALGIAS == 1) ^ (CONTACTOOTROSCASOS == 1)) | (DISNEA == 0)
or

(((TOS == 1) | ((((ODINOFAGIA != 1) | (MIALGIAS == 1)) |
((MIALGIAS != 0) ^ (MIALGIAS == 1))) &
(((MIALGIAS == 1) & (MIALGIAS == 1)) | ((CONJUNTIVITIS != 0) |
(DISNEA != 0))))) ^ ((IRRITABILIDAD == 0) |
(CONJUNTIVITIS != 1))) | (DISNEA == 0)



GP VEGA, Matriz de confusión [231, 148, 96, 124] :

MIALGIAS != 0 or
CONJUNTIVITIS != 0 or
CONTACTO_CERDOS != 0 or
CONTACTOOTROSCASOS != 0 or
ESCALOFRIOS != 0 or
0
```

### 5.1.2. Pacientes de 7 a 15 años

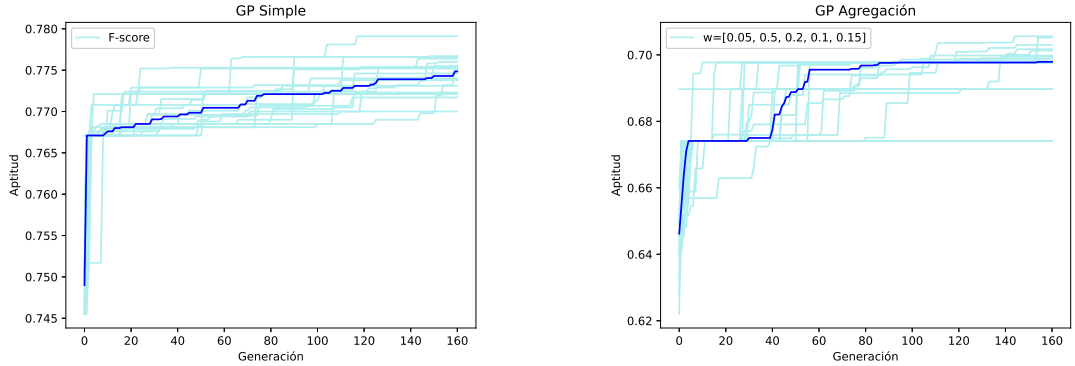
Para el conjunto de pacientes en el rango de edad de 7 a 15 años, los parámetros de los algoritmos se presentan en la tabla 5.3 son similares al caso anterior, solo que se modificó el vector de pesos para el algoritmo GP Agregación, pues el número de instancias positivas en este caso es mayor que

el número de instancias negativas.

| Algoritmo | N. Individ. | N. Gener. | P. Mutación | P. Cruza | F. Aptitud | Selecc. |
|---|---|---|---|---|---|---|
| GP Simple | 80 | 160 | 0.2 | 0.8 | F-score | Ruleta |
| GP Agregación | 80 | 160 | 0.2 | 0.8 | .05,.5,.2,.1,.15 | Ruleta |
| GP VEGA | 60 | 80 | 0.25 | 0.8 | 5 objetivos | Elitismo |

Tabla 5.3: Parámetros en algoritmos para pacientes de 7 a 15 años.

La figura 5.3 muestra que antes de la generación 60, para el caso de GP agregación, se presenta el mayor incremento en la función de aptitud, mientras que para GP simple, el incremento es un poco más uniforme salvo por las primeras 3 generaciones, donde fue mucho mayor.




(a) (b)

Figura 5.3: Función de aptitud para pacientes de 7 a 15 años.

Al observar los valores resultantes de los mejores individuos, tabla 5.4, no existe tanta diferencia para los cuatro clasificadores en los tres primeros objetivos, precision, f-score y exactitud; sin embargo, en los valores de exahustividad y especificidad la brecha es mayor, lo que indica que a pesar de que hay mayor número de instancias positivas, los métodos basados en programación genética clasifican de mejor manera como positivos los casos realmente positivos.

| Algoritmo | Precisión | Fscore | Exactitud | Sensibilidad | Especificidad |
|---|---|---|---|---|---|
| GP Simple | 0.6204 | 0.7234 | 0.6246 | 0.8673 | 0.2925 |
| GP Agregación | 0.6263 | 0.7292 | 0.6334 | 0.8724 | 0.3061 |
| GP VEGA | 0.6204 | 0.7234 | 0.6246 | 0.8673 | 0.2925 |
| weka J48 | 0.644 | 0.701 | 0.6246 | 0.769 | 0.432 |

Tabla 5.4: Evaluación sobre el conjunto de prueba para pacientes de 7 a 15 años

El número de nodos en el árbol que genera weka es de 119, mientras que la longitud de la mejor

```
Number of Leaves  :     60

Size of the tree :     119


Time taken to build model: 0.01 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances         213               62.4633 %
Incorrectly Classified Instances       128               37.5367 %
Kappa statistic                          0.2075
Mean absolute error                      0.4047
Root mean squared error                  0.5188
Relative absolute error                 83.228  %
Root relative squared error            104.7307 %
Total Number of Instances              341

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.432    0.231    0.583      0.432    0.496      0.214    0.652     0.553     0
                 0.769    0.568    0.644      0.769    0.701      0.214    0.652     0.675     1
Weighted Avg.    0.625    0.424    0.618      0.625    0.613      0.214    0.652     0.623

=== Confusion Matrix ===

   a   b   <-- classified as
  63  83 |   a = 0
  45 150 |   b = 1
```

Figura 5.4: Evaluación J48 en weka, pacientes de 7 a 15 años.

regla obtenida por GP simple es mucho menor, parecido a las obtenidas por GP VEGA, figura 5.4.

La regla de GP Simple, es fácilmente interpretable, pues para que la condición se cumpla y el caso sea clasificado como positivo, ambos operandos de la cláusula *and* deben ser positivos, en cuyo caso se cumple si $(TOS == 0)$ es falso y $(FIEBRE == 1)$ es verdadero, en otras palabras, si el paciente tiene fiebre y tos. A continuación se muestran los modelos de cada algoritmo:

```
GP Simple, Matriz de confusión [43, 103, 25, 170]

 (FIEBRE == 1) & (((FIEBRE == 0) |
 (FIEBRE == 1)) ^ (TOS == 0))


GP Agregación, Matriz de confusión [45, 101, 24, 171]

 (((TOS != 0) | ((RIORREA != 0) &
 ((ESCALOFRIOS != 1) | (ESCALOFRIOS != 1)))) &
 (FIEBRE == 1)) &
 (((TOS == 1) & (((CIANOSIS != 1) | (RIORREA == 0)) |
 (ODINOFAGIA != 1))) | ((((((TOS == 1) & (((FIEBRE != 0) &
 (((RIORREA != 0) & (RIORREA != 0)) | (ESCALOFRIOS == 1))) &
 (ESCALOFRIOS == 1))) ^ (TOS != 1)) | ((FIEBRE == 0) |
 ((MIALGIAS != 1) ^ (((((CONJUNTIVITIS == 1) & (RIORREA != 0)) |
 (RIORREA != 0)) & (FIEBRE == 1)) & ((CIANOSIS != 1) &
 (FIEBRE == 0)))))) & ((((ESCALOFRIOS == 1) | (FIEBRE == 1)) &
```

```
 (MIALGIAS == 1)) | (ESCALOFRIOS == 1))) & (RIORREA != 0)))
 or

(((TOS != 0) | ((RIORREA != 0) & ((FIEBRE == 0) |
(ESCALOFRIOS != 1)))) & (FIEBRE == 1)) & (((TOS == 1) &
(((CIANOSIS != 1) | (RIORREA == 0)) | (ODINOFAGIA != 1))) |
((((((TOS != 1) & (((FIEBRE != 0) & ((((CIANOSIS == 0) |
((TOS != 0) | ((RIORREA != 0) | (CONTACTO_AVES != 1)))) &
(RIORREA != 0)) | (ESCALOFRIOS == 1))) & (ESCALOFRIOS == 1))) ^
(TOS != 1)) | ((ESCALOFRIOS == 1) | ((MIALGIAS != 1) ^
(((((CONJUNTIVITIS == 1) & (RIORREA != 0)) | (RIORREA != 0)) &
(FIEBRE == 1)) & ((CIANOSIS != 1) & (FIEBRE == 0)))))) &
((((ESCALOFRIOS == 1) | (FIEBRE == 1)) & (MIALGIAS == 1)) |
(ESCALOFRIOS == 1))) & (RIORREA != 0)))
or

(((TOS != 0) | ((RIORREA != 0) & ((FIEBRE == 0) |
(ESCALOFRIOS != 1)))) & (FIEBRE == 1)) & (((TOS == 1) &
(((CIANOSIS != 1) | (RIORREA == 0)) | (ODINOFAGIA != 1))) |
((((((TOS != 1) & (((FIEBRE != 0) & ((((CIANOSIS == 0) |
(ESCALOFRIOS == 1)) & (RIORREA != 0)) | (ESCALOFRIOS == 1))) &
(ESCALOFRIOS == 1))) ^ (TOS != 1)) | ((ESCALOFRIOS == 1) |
((MIALGIAS != 1) ^ (((((CONJUNTIVITIS == 1) & (RIORREA != 0)) |
(RIORREA != 0)) & (FIEBRE == 1)) & ((CIANOSIS != 1) &
(FIEBRE == 0)))))) & ((((ESCALOFRIOS == 1) | (FIEBRE == 1)) &
(MIALGIAS == 1)) | (ESCALOFRIOS == 1))) & (RIORREA != 0)))
```

El modelo generado por GP VEGA, también es sencillo de interpretar, pues todas las reglas obtenidas son similares y son expresiones *xor* donde el segundo operando siempre resulta verdadero, luego para que las expresiones sean verdaderas, el primer operando del *xor* debe ser falso y dado que son *or*, cada una de las expresiones debe ser falsa, i.e, cuando $(FIEBRE \neq 1)$ y $(TOS \neq 1)$ no se cumplen, o lo que es lo mismo que el paciente tiene fiebre y tiene tos, tal como la regla obtenida por GP Agregación.

```
GP VEGA, Matriz de confusión [43, 103, 25, 170]

((FIEBRE != 1) | (TOS != 1)) ^ ((FIEBRE != 1) | (1)) or
((TOS != 1) | (FIEBRE != 1)) ^ ((TOS != 1) | (1)) or
((TOS != 1) | (FIEBRE != 1)) ^ ((FIEBRE != 1) | (1)) or
((FIEBRE != 1) | (TOS != 1)) ^ ((TOS != 1) | (1)) or
((TOS != 1) | (FIEBRE != 1)) ^ ((TOS != 1) | (1)) or
((TOS != 1) | (FIEBRE != 1)) ^ ((FIEBRE != 1) | (1))
```

### 5.1.3. Pacientes de 16 a 60 años

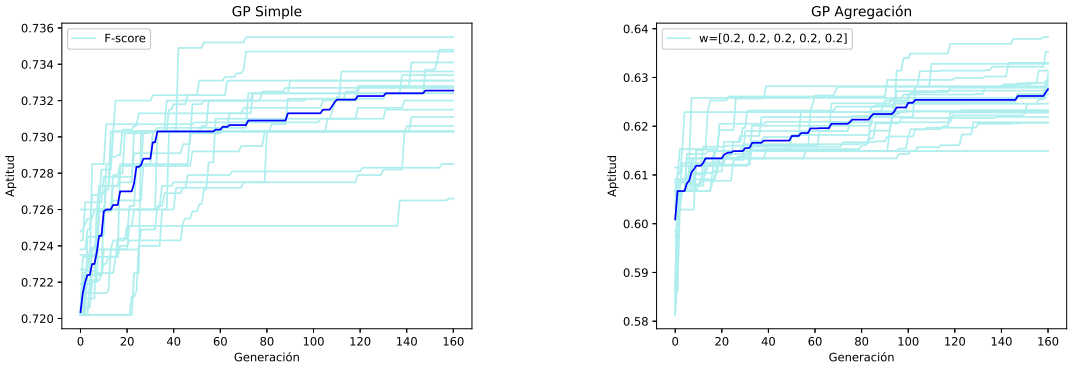
Para el conjunto de pacientes en el rango de edad de 16 a 60 años, los parámetros de ejecución se encuentran en la tabla 5.5, aunque se trató también, al igual que en los casos anteriores, un conjunto de clases no balanceadas, el vector de pesos para GP Agregación, fue $w = [0.2, 0.2, 0.2, 0.2, 0.2]$:

| Algoritmo | N. Individ. | N. Gener. | P. Mutación | P. Cruza | F. Aptitud | Selecc. |
|---|---|---|---|---|---|---|
| GP Simple | 80 | 160 | 0.2 | 0.8 | F-score | Ruleta |
| GP Agregación | 80 | 160 | 0.2 | 0.8 | .2, .2, .2, .2, .2 | Ruleta |
| GP VEGA | 60 | 80 | 0.25 | 0.8 | 5 objetivos | Elitismo |

Tabla 5.5: Parámetros en algoritmos para pacientes de 16 a 60 años.

Las gráficas en 5.7 muestran que a los algoritmos GP Agregación y GP Simple, les toma más generaciones llegar a un valor constante con respecto a los conjuntos de datos anteriores.

En la tabla 5.6 se observa que los algorimos de programación genética alcanzan un alto valor en f-score, exactitud y exahustividad con respecto a J48, mientras que en el valor de la precisión todos son cercanos. La especificidad también debe tomarse en cuenta, una baja especificidad indica que el clasificador no es capaz de identificar casos negativos, como en el caso de GP simple, aunque presenta el máximo valor para sensibilidad su valor en especificidad es muy bajo, por lo que no es tan conveniente tomarlo como el mejor, es preferible optar por un clasificador con mayor especificidad que clasifique bien los casos positivos, como son GP Agregación y GP VEGA.

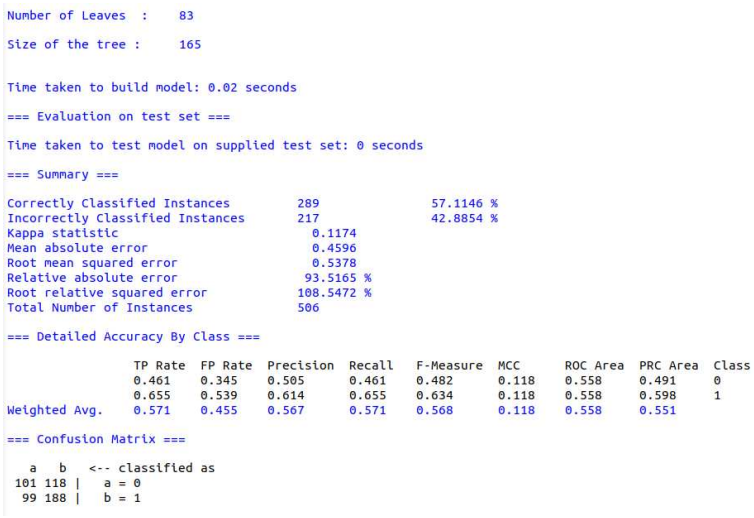



(a) (b)

Figura 5.5: Función de aptitud para pacientes de 16 a 60 años.

| Algoritmo | Precisión | Fscore | Exactitud | Sensibilidad | Especificidad |
|---|---|---|---|---|---|
| GP Simple | 0.5891 | 0.7245 | 0.5968 | 0.9409 | 0.1409 |
| GP Agregación | 0.6076 | 0.6809 | 0.5909 | 0.7743 | 0.3454 |
| GP VEGA | 0.6 | 0.7186 | 0.6047 | 0.8958 | 0.2181 |
| weka J48 | 0.614 | 0.634 | 0.5711 | 0.655 | 0.461 |

Tabla 5.6: Evaluación sobre el conjunto de prueba para pacientes de 16 a 60 años



```
Number of Leaves  :     83

Size of the tree :     165


Time taken to build model: 0.02 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances         289               57.1146 %
Incorrectly Classified Instances       217               42.8854 %
Kappa statistic                          0.1174
Mean absolute error                      0.4596
Root mean squared error                  0.5378
Relative absolute error                 93.5165 %
Root relative squared error            108.5472 %
Total Number of Instances              506

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.461    0.345    0.505      0.461    0.482      0.118    0.558     0.491     0
               0.655    0.539    0.614      0.655    0.634      0.118    0.558     0.598     1
Weighted Avg.  0.571    0.455    0.567      0.571    0.568      0.118    0.558     0.551

=== Confusion Matrix ===

   a   b   <-- classified as
 101 118 |   a = 0
  99 188 |   b = 1
```

Figura 5.6: Evaluación J48 en weka, pacientes de 16 a 60 años.

```
GP Simple, Matriz de confusión [31, 188, 16, 271]

(FIEBRE != 0) | (MIALGIAS == 1)


GP Agregación, Matriz de confusión [76, 143, 64, 223]

(FIEBRE != 0) & (((CONTACTOOTROSCASOS != 0) |
(ARTRALGIAS != 0)) | (ESCALOFRIOS == 1))
or

(FIEBRE != 0) & (((((CONTACTO_AVES != 1) & (ESCALOFRIOS != 1)) &
(ARTRALGIAS != 0)) | ((ARTRALGIAS != 0) | ((ARTRALGIAS != 0) &
((ARTRALGIAS != 0) | ((((CIANOSIS != 0) ^ (FIEBRE != 0)) &
(((DOLOR_TORACICO != 0) ^ ((ARTRALGIAS != 1) & (ODINOFAGIA != 1))) |
(((((FIEBRE != 0) & (CONTACTO_CERDOS != 0)) & ((CONTACTO_AVES != 1) &
(CONTACTO_AVES == 0))) | (CIANOSIS == 1)) &
((((CONTACTOOTROSCASOS == 1) & (ARTRALGIAS != 1)) | (CIANOSIS == 1)) |
(INIC_SUBITO_SINTOMAS != 0))))) | ((DISNEA == 1) |
(ARTRALGIAS != 1))))))) | (ESCALOFRIOS == 1))
```

```
or

(FIEBRE != 0) & ((((ARTRALGIAS != 0) & (ESCALOFRIOS != 1)) |
((CIANOSIS == 1) | ((ARTRALGIAS != 0) & ((ARTRALGIAS != 0) |
((((CIANOSIS != 0) ^ (ARTRALGIAS != 0)) & ((ARTRALGIAS == 1) |
(((((((DISNEA == 1) ^ (CEFALEA == 0)) | (FIEBRE != 0)) &
(FIEBRE != 0)) & (((ARTRALGIAS != 0) ^ (DISNEA == 1)) &
(CONTACTO_AVES == 0))) | (ARTRALGIAS == 0)) & ((((((DISNEA != 0) ^
(DIARREA != 0)) | (FIEBRE != 0)) & (ARTRALGIAS != 1)) |
(CIANOSIS == 1)) | (ARTRALGIAS == 0))))) | ((((CIANOSIS == 0) |
(POLIPNEA == 1)) | (DIARREA != 0)) | (ARTRALGIAS != 1))))))) |
(ESCALOFRIOS == 1))
```

El tamaño del árbol generado por C4.5 es de 165 nodos, y el modelo generado por GP Agregación también es grande. En el caso de VEGA, el modelo es mucho más pequeño y se puede reducir de manera fácil. En este caso, como las reglas son disyunciones cuyo segundo operando es $(FIEBRE == 1)$, si el paciente presenta este síntoma, se clasifica como positivo. O si $(VOMITO! = 0)$ se cumple y además si $(DISNEA == 1)$ o $(MIALGIAS \neq 0)$ o $(RIORREA \neq 0)$ o $(CONTACTOOTROSCASOS \neq 0)$, el paciente es positivo, en otras palabras, si el paciente tiene vómito y tiene además disnea, mialgias, riorrea o tuvo contacto con otro caso positivo, entonces es clasificado como positivo a influenza.

```
GP VEGA, Matriz de confusión [48, 171, 29, 258]

((VOMITO != 0) & ((DISNEA == 1) | ((MIALGIAS != 0) |
(RIORREA != 0)))) | (FIEBRE == 1) or
((VOMITO == 1) & ((DISNEA == 1) | ((RIORREA != 0) |
(RIORREA != 0)))) | (FIEBRE == 1) or
((VOMITO != 0) & ((DISNEA == 1) | ((MIALGIAS != 0) |
(RIORREA != 0)))) | (FIEBRE == 1) or
((VOMITO != 0) & ((DISNEA == 1) | ((FIEBRE == 1) |
(RIORREA != 0)))) | (FIEBRE == 1) or
((VOMITO == 1) & ((DISNEA == 1) | ((RIORREA != 0) |
(RIORREA != 0)))) | (FIEBRE == 1) or
((VOMITO == 1) & ((DISNEA == 1) | ((CONTACTOOTROSCASOS != 0) |
(RIORREA != 0)))) | (FIEBRE == 1)
```

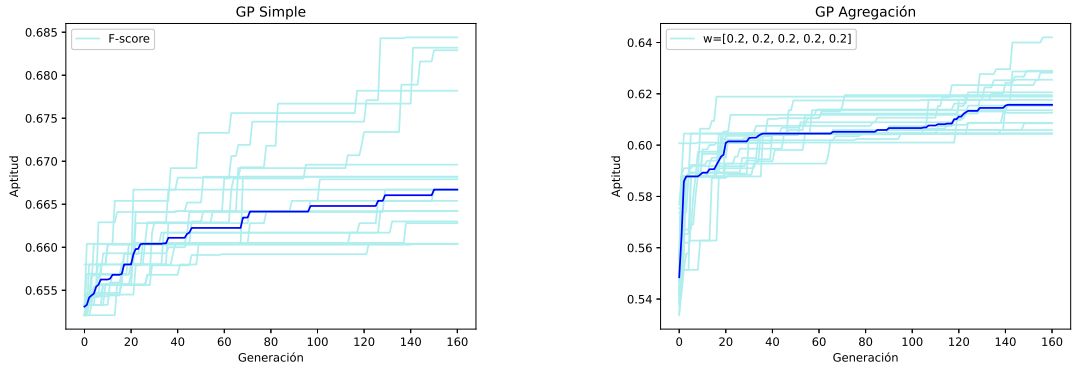
### 5.1.4. Pacientes mayores a 60 años

Para el conjunto de pacientes en el rango de edad mayores a 60 años, los parámetros de ejecución se mantuvieron como en el caso anterior, tabla 5.7:

| Algoritmo | N. Individ. | N. Gener. | P. Mutación | P. Cruza | F. Aptitud | Selecc. |
|---|---|---|---|---|---|---|
| GP Simple | 80 | 160 | 0.2 | 0.8 | F-score | Ruleta |
| GP Agregación | 80 | 160 | 0.2 | 0.8 | .2, .2, .2, .2, .2 | Ruleta |
| GP VEGA | 60 | 80 | 0.25 | 0.8 | 5 objetivos | Elitismo |

Tabla 5.7: Parámetros en algoritmos para pacientes mayores a 60 años.

Para AG Simple y AG agregación, durante las primeras 100 generaciones es donde la función de aptitud incrementa su valor más rápidamente.




(a) (b)

Figura 5.7: Función de aptitud para pacientes mayores a 60 años.

Los algoritmos de programación genética presentan un mejor rendimiento que weka en precisión, f-score, exactitud y sensibilidad, a excepción de GP VEGA que presenta un valor ligeramente bajo en exactitud, pero no hay que olvidar que las clases están desbalanceadas con mayor número de casos negativos y weka al tener una mayor especificidad clasifica mejor estos casos.

| Algoritmo | Precisión | Fscore | Exactitud | Sensibilidad | Especificidad |
|---|---|---|---|---|---|
| GP Simple | 0.4414 | 0.5568 | 0.522 | 0.7538 | 0.3541 |
| GP Agregación | 0.4561 | 0.581 | 0.5408 | 0.8 | 0.3541 |
| GP_VEGA | 0.419 | 0.5176 | 0.4968 | 0.6769 | 0.3645 |
| weka J48 | 0.408 | 0.43 | 0.5157 | 0.453 | 0.558 |

Tabla 5.8: Evaluación sobre el conjunto de prueba para pacientes mayores a 60 años

```
Number of Leaves  :     45

Size of the tree :     89


Time taken to build model: 0.01 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.01 seconds

=== Summary ===

Correctly Classified Instances          82               51.5723 %
Incorrectly Classified Instances        77               48.4277 %
Kappa statistic                          0.0108
Mean absolute error                      0.4855
Root mean squared error                  0.5809
Relative absolute error                 97.615  %
Root relative squared error            116.7636 %
Total Number of Instances              159

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.558    0.547    0.602      0.558    0.579      0.011    0.535     0.630     0
               0.453    0.442    0.408      0.453    0.430      0.011    0.535     0.426     1
Weighted Avg.  0.516    0.505    0.524      0.516    0.519      0.011    0.535     0.548

=== Confusion Matrix ===

  a  b   <-- classified as
 53 42 |  a = 0
 35 29 |  b = 1
```

Figura 5.8: Evaluación J48 en weka, pacientes mayores a 60 años.

Por otro lado, el tamaño del árbol de j48 es de 89 nodos, mientras que los modelos generados por GP Simple y GP VEGA son mucho más cortos e interpretables. En el primer caso, el modelo indica que si un paciente presenta riorrea o el *xor* de las condiciones de cianosis, disnea, mialgias y contacto_aves se cumple, entonces se clasifica el caso como positivo.

GP Simple, Matriz de confusión [34, 61, 15, 49]

```
(RIORREA == 1) | ((((CIANOSIS != 1) ^ (DISNEA == 0)) ^
(MIALGIAS == 1)) ^ (CONTACTO_AVES == 0))
```

GP Agregación, Matriz de confusión [34, 61, 12, 52]

```
((((SEXO != 0) | (INIC_SUBITO_SINTOMAS != 1)) ^
((DISNEA != 0) & (ESCALOFRIOS != 1))) | ((ODINOFAGIA != 0) |
((CONJUNTIVITIS == 1) & (SEXO != 0)))) & (TOS != 0)
or

((((SEXO != 0) | (INIC_SUBITO_SINTOMAS != 1)) ^
((DISNEA != 0) & (ESCALOFRIOS != 1))) | ((ODINOFAGIA != 0) |
((((((SEXO != 0) ^ (INIC_SUBITO_SINTOMAS != 1)) ^
((DISNEA != 0) & (ESCALOFRIOS != 1))) | ((ODINOFAGIA == 0) |
((CONJUNTIVITIS == 1) & (ODINOFAGIA != 0)))) &
(TOS != 0)) & (SEXO != 0)))) & (TOS != 0)
or

((((SEXO != 0) | (INIC_SUBITO_SINTOMAS != 1)) ^
((DISNEA != 0) & (ESCALOFRIOS != 1))) | ((ODINOFAGIA != 0) |
((((((SEXO != 0) ^ (INIC_SUBITO_SINTOMAS != 1)) ^
```

```
((DISNEA != 0) & (ESCALOFRIOS != 1))) | ((ODINOFAGIA == 0) |
((((ODINOFAGIA != 1) | (SEXO != 0)) &
((INIC_SUBITO_SINTOMAS != 1) ^ (ESCALOFRIOS != 1))) &
(ODINOFAGIA != 0)))) & (TOS != 0)) & (SEXO != 0))))
& (TOS != 0)
```

En el caso de GP VEGA, las reglas son parecidas y los síntomas que aparecen con cefalea, diarrea y conjuntivitis. Este modelo clasifica un paciente como positivo si presenta cefalea, o tiene conjuntivitis pero no diarrea.

```
GP VEGA, Matriz de confusión [35, 60, 20, 44]

CEFALEA != 0 or
(CEFALEA != 0) & (CEFALEA != 0) or
(DIARREA == 0) & (CONJUNTIVITIS != 0) or
(CONJUNTIVITIS != 0) & (DIARREA == 0) or
(DIARREA == 0) & (CONJUNTIVITIS != 0) or
(DIARREA == 0) & ((CONJUNTIVITIS != 0) & (DIARREA == 0))
```

## 5.2. SARS-CoV-2

A continuación se presentan los resultados de las ejecuciones para SARS-CoV-2. En el caso de Baja California Sur, para la evaluación del modelo GP Agregación, se seleccionaron 4 reglas y para GP VEGA, 2 reglas. Para los casos del Estado de México y Veracruz, en GP Agregación fueron 2 reglas y 3 para GP VEGA. En GP Simple, en las tres entidades federativas se eligió una sola regla.

Al igual que en el caso de Influenza, el número de ejecuciones para cada conjunto de datos fue de 50, el mejor modelo obtenido se eligió dependiendo principalmente del valor de f-score y de la sensibilidad, sin dejar que el valor de la especificidad fuera muy bajo.

Por otro lado, debido al desbalanceo de clases y a la naturaleza del conjunto de datos, fue necesario implementar una técnica de submuestreo durante el algoritmo de clasificación. Esta técnica consistió de una selección aleatoria de registros de la clase mayoritaria tal que hubiese el mismo número de registros de la clase minoritaria, luego sobre este subconjunto de datos se realiza la evaluación de la función de aptitud de los individuos y dicho subconjunto de datos, es generado cada cierto número de generaciones. Para los tres algoritmos implementados, el submuestreo ocurre cada 10 generaciones.

### 5.2.1. Baja California Sur

Los parámetros de ejecución se muestran en la tabla 5.9, para el caso de GP VEGA, como en el caso de influenza, el número de generaciones e individuos es menor, pues al aumentar el número de generaciones los individuos se especializan demasiado en un solo objetivo, disminuyendo la

calidad en los demás; por otro lado, los resultados de este algoritmo con 50 o 60 individuos fueron casi iguales.

| Algoritmo | N. Individ. | N. Gener. | P. Mutación | P. Cruza | F. Aptitud | Selecc. |
|---|---|---|---|---|---|---|
| GP Simple | 60 | 100 | 0.25 | 0.8 | F-score | Ruleta |
| GP Agregación | 60 | 100 | 0.25 | 0.8 | .05,.5,.05,.1,.3 | Ruleta |
| GP VEGA | 50 | 80 | 0.25 | 0.8 | 5 objetivos | Elitismo |

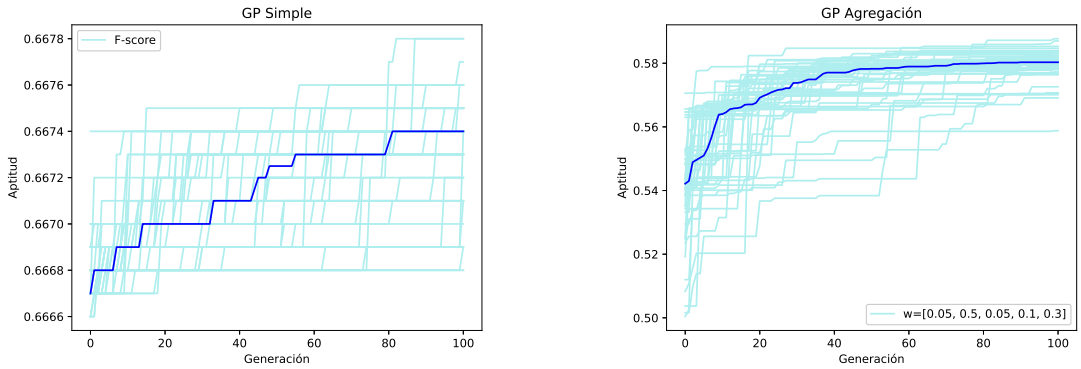
Tabla 5.9: Parámetros en algoritmos para Baja California Sur

Los resultados de las evaluaciones del mejor modelo encontrado por algoritmo, se enlistan en la tabla 5.10, en la cual se observa que los valores de f-score y sensibilidad para los algoritmos de programación genética, son considerablemente mayores que C4.5 de weka. En el caso de GP Simple, ocurre similar a los pacientes de 16 a 60 años en el conjunto de influenza, pues aunque presenta el mayor valor en sensibilidad, el valor de especificidad es demasiado bajo. GP Agregación y GP VEGA, presentan un mejor balanceo de estas dos métricas, priorizando la que clasifica mejor los casos positivos.

Aunque el valor de exactitud es menor para los primeros tres modelos, debe considerarse que las clases no están balanceadas, habiendo más casos negativos para este conjunto, por lo que weka cuya especificidad es mayor, presenta un valor más alto, sin embargo, esto no representa que clasifica mejor los casos.

| Algoritmo | Precisión | Fscore | Exactitud | Sensibilidad | Especificidad |
|---|---|---|---|---|---|
| GP Simple | 0.3584 | 0.5272 | 0.3631 | 0.9964 | 0.0124 |
| GP Agregación | 0.4346 | 0.54604 | 0.5553 | 0.73408 | 0.4527 |
| GP VEGA | 0.4183 | 0.5126 | 0.5516 | 0.6618 | 0.4905 |
| weka J48 | 0.568 | 0.196 | 0.654 | 0.118 | 0.950 |

Tabla 5.10: Evaluación sobre el conjunto de prueba para Baja California Sur.

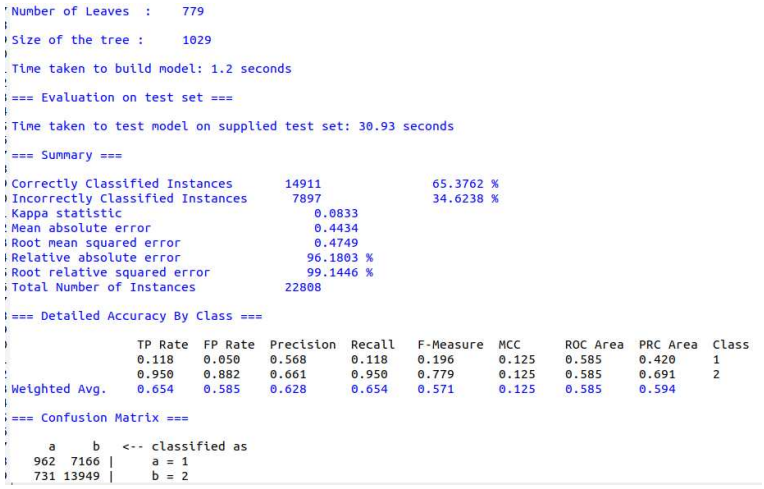



(a) (b)

Figura 5.9: Función de aptitud para Baja California Sur.

El tamaño del modelo para este conjunto, tanto para weka como para los algoritmos de programación genética, es más grande. Para weka se tiene un árbol de 1029 nodos, mientras que los modelos de programación genética se visualizan más pequeños:

```
GP Simple, Matriz de confusión [183, 14497, 28, 8100]

IFELSE (((UCI <= 98) ^ (EPOC >= 97)) & (INMUSUPR != 1)) ^
       ((NEUMONIA >= 98) | (((EMBARAZO > 99) ^
       (((DIABETES != 97) ^ ((INTUBADO != 2) | (((EDAD <= 111) ^
       (OTRO_CASO != 98)) & ((SEXO == 99) ^ ((UCI < 97) |
       (DIABETES < 1)))))) ^ (OTRO_CASO != 1))) ^ (UCI < 97)))
 THEN IFELSE INTUBADO > 98 2 1
 ELSE IFELSE (EMBARAZO >= 98) ^ ((EDAD >= 8) & (EDAD >= 8)) 1 2
```



```
Number of Leaves  :     779

Size of the tree :     1029

Time taken to build model: 1.2 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 30.93 seconds

=== Summary ===

Correctly Classified Instances       14911               65.3762 %
Incorrectly Classified Instances      7897               34.6238 %
Kappa statistic                          0.0833
Mean absolute error                      0.4434
Root mean squared error                  0.4749
Relative absolute error                 96.1803 %
Root relative squared error             99.1446 %
Total Number of Instances            22808

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.118    0.050    0.568      0.118    0.196      0.125    0.585     0.420     1
               0.950    0.882    0.661      0.950    0.779      0.125    0.585     0.691     2
Weighted Avg.  0.654    0.585    0.628      0.654    0.571      0.125    0.585     0.594

=== Confusion Matrix ===

     a     b   <-- classified as
   962  7166 |     a = 1
   731 13949 |     b = 2
```

Figura 5.10: Evaluación J48 en weka, Baja California Sur.

```
GP Agregación, Matriz de confusión [7383, 7297, 2655, 5473]

IFELSE ((OBESIDAD == 2) | (ASMA == 97)) ^ (OBESIDAD == 98)
 THEN IFELSE (OTRO_CASO == 2) & (NEUMONIA == 2) 2 1
 ELSE IFELSE ASMA == 97 2 1
or
IFELSE ((OBESIDAD == 2) | (HIPERTENSION < 99)) ^ (OBESIDAD == 98)
 THEN IFELSE ((OTRO_CASO == 2) & (NEUMONIA == 2)) &
                    (NEUMONIA == 2) 2 1
 ELSE IFELSE (OTRO_CASO == 2) & (NEUMONIA == 2) 2 1
or
IFELSE (OTRO_CASO == 2) & (NEUMONIA == 2)
 THEN IFELSE NEUMONIA == 2 2 1
 ELSE IFELSE HIPERTENSION == 99 2 1
or
```

```
IFELSE (OTRO_CASO == 2) ^ (ASMA == 98)
 THEN IFELSE NEUMONIA == 2 2 1
 ELSE IFELSE HIPERTENSION == 99 2 1

GP VEGA, Matriz de confusión [7202, 7478, 2748, 5380]

IFELSE ((SEXO == 99) | (OTRO_CASO == 1)) & (TABAQUISMO != 99)
    THEN IFELSE INTUBADO <= 97 1 2
    ELSE IFELSE EPOC >= 97 1 2
or

IFELSE (((UCI >= 2) ^ (((((SEXO == 99) & (CARDIOVASCULAR != 97)) ^
    ((((HIPERTENSION >= 98) ^ (TABAQUISMO != 98)) | (ASMA > 98)) &
    (((INTUBADO <= 97) | (OTRA_COM != 97)) ^
    (HIPERTENSION > 98)))) | (INTUBADO > 99)) ^ ((((((UCI >= 2) ^
    (((((SEXO == 99) & (ASMA >= 99)) ^ ((((HIPERTENSION >= 98) ^
    (INTUBADO != 1)) | (ASMA > 98)) & (((NEUMONIA == 2) |
    (OTRA_COM != 97)) ^ (SEXO == 99)))) | (NEUMONIA == 1)) ^
    (((CARDIOVASCULAR != 2) & (ASMA <= 1)) & (((EDAD <= 50) &
    (((EDAD >= 25) & ((((EDAD >= 119) & ((EDAD >= 36) &
    ((UCI >= 2) ^ (((((CARDIOVASCULAR != 2) &
    (CARDIOVASCULAR != 97)) ^ ((((CARDIOVASCULAR >= 97) ^
    (INTUBADO != 1)) | (ASMA > 98)) & (((NEUMONIA == 2) |
    (OTRA_COM != 97)) ^ (SEXO == 99)))) | (INTUBADO > 99)) ^
    (((CARDIOVASCULAR != 2) & (EPOC != 99)) & ((((HIPERTENSION > 98) |
    ((CARDIOVASCULAR != 2) & (NEUMONIA == 2))) ^ (RENAL_CRONICA < 2)) ^
    (EDAD >= 10)))))))) ^ (EPOC >= 99)) ^ (INMUSUPR == 2))) &
    (EMBARAZO >= 98))) | ((NEUMONIA >= 1) |
    (CARDIOVASCULAR != 2)))))) | (OTRA_COM < 97)) & (ASMA <= 2)) &
    (HIPERTENSION > 98)) & ((((HIPERTENSION > 98) | ((ASMA > 98) &
    (OTRA_COM < 98))) ^ (RENAL_CRONICA < 2)) ^
    (EDAD >= 10))))) | (EDAD <= 50)) & (ASMA <= 2)
    THEN IFELSE NEUMONIA == 1 1 2
    ELSE IFELSE OBESIDAD < 97 1 2
```

### 5.2.2. Estado de México

Los parámetros de ejecución mostrados en la tabla 5.11, iguales a los parámetros del conjunto anterior.

| Algoritmo | N. Individ. | N. Gener. | P. Mutación | P. Cruza | F. Aptitud | Selecc. |
|---|---|---|---|---|---|---|
| GP Simple | 60 | 100 | 0.25 | 0.8 | F-score | Ruleta |
| GP Agregación | 60 | 100 | 0.25 | 0.8 | .05,.5,.05,.1,.3 | Ruleta |
| GP VEGA | 50 | 80 | 0.25 | 0.8 | 5 objetivos | Elitismo |

Tabla 5.11: Parámetros en algoritmos para Estado de México.

La tabla 5.12 nos indica los algoritmos de programación genética tienen un mejor desempeño en las métricas de f-score y sensibilidad. El valor en exactitud mayor en weka, al igual que antes se mencionó, se debe al desbalanceo de clases con mayor número de casos negativos y el hecho de que weka clasifica mejor estos casos.

Por otro lado, la precisión es menor para los tres primeros casos, pero esto no indica que el clasificador sea peor, pues la precisión mide de los casos que fueron clasificados como positivos, el porcentaje de los que realmente son positivos, pero los casos que clasifica weka como positivos son muy pocos con respecto a los que realmente lo son. En la figura 5.11, puede observarse que C4.5 solo clasifica 16,662 como positivos, mientras que GP Simple clasifica 40,507, GP Agregación 27,704 y GP VEGA 27,567. Por otro lado, aunque GP Simple tiene una mayor cifra en el razonamiento anterior, su valor en especificidad es muy bajo, por lo que tampoco puede considerarse tan bueno.

| Algoritmo | Precisión | Fscore | Exactitud | Sensibilidad | Especificidad |
|---|---|---|---|---|---|
| GP Simple | 0.4888 | 0.6544 | 0.4939 | 0.9898 | 0.0285 |
| GP Agregación | 0.5440 | 0.6032 | 0.5689 | 0.6770 | 0.4674 |
| GP VEGA | 0.5451 | 0.6026 | 0.5698 | 0.6736 | 0.4724 |
| weka J48 | 0.699 | 0.515 | 0.628 | 0.407 | 0.835 |

Tabla 5.12: Evaluación sobre el conjunto de prueba para 15-Estado de México.

Otro aspecto importante a destacar es el tamaño del modelo generado, para el caso de C4.5 de weka, el árbol es de 1026 nodos, lo que es considerablemente mayor a los modelos genéticos de este trabajo, mostrados a continuación:

```
Number of Leaves  :     787

Size of the tree :     1026

Time taken to build model: 3.44 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 64.45 seconds

=== Summary ===

Correctly Classified Instances       53081               62.8007 %
Incorrectly Classified Instances     31442               37.1993 %
Kappa statistic                          0.2456
Mean absolute error                      0.4579
Root mean squared error                  0.4803
Relative absolute error                 91.6749 %
Root relative squared error             96.1046 %
Total Number of Instances            84523

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.407    0.165    0.699      0.407    0.515      0.269    0.635     0.605     1
               0.835    0.593    0.600      0.835    0.698      0.269    0.635     0.601     2
Weighted Avg.  0.628    0.386    0.648      0.628    0.609      0.269    0.635     0.603

=== Confusion Matrix ===

     a     b   <-- classified as
 16662 24258 |     a = 1
  7184 36419 |     b = 2
```

Figura 5.11: Evaluación J48 en weka, Estado de México.

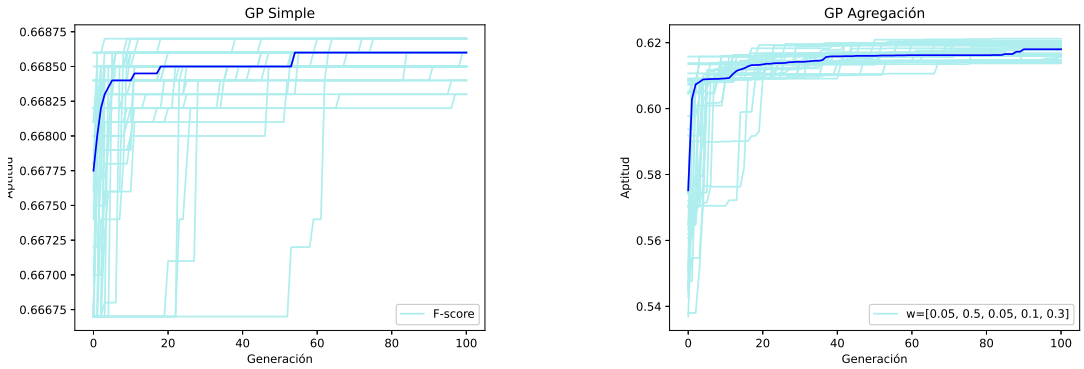
```
GP Simple, Matriz de confusión [1244, 42359, 413, 40507]

 IFELSE EDAD <= 11
 THEN IFELSE INTUBADO >= 1 2 1
 ELSE IFELSE EDAD <= 11 2 1
```

Para el caso de GP Simple, la edad de 11 años parece ser el atributo clasificador, pero como se vio antes, no se obtienen buenos resultados en especificidad. Para GP Agregación, las dos reglas que conforman el modelo son sencillas y fáciles de interpretar: Se sabe que los casos donde INTUBADO es diferente a 97, son casos de pacientes hospitalizados (97 = no aplica = casos ambulatorios), y la primer regla indica que si además su edad es mayor a 24, entonces es positivo a covid. Cuando el valor en INTUBADO es 97, un caso ambulatorio, si no tiene obesidad o se sabe su estado de embarazo, entonces es negativo, en otro caso, es clasificado positivo. En la segunda regla, la segunda parte también representa los casos ambulatorios, donde el valor de EDAD y EMBARAZO determinan el valor del clasificador. Cabe mencionar que existen mayor número de casos ambulatorios que hospitalizados, tal como se visualiza en las gráficas de la detección de anomalías 4.2, además el valor de EMBARAZO $\leq 2$ solo aplica a mujeres, por lo que este modelo clasifica un hombre ambulatorio sabiendo su edad y si tiene obesidad.

```
GP Agregación, Matriz de confusión [20384, 23219, 13216, 27704]

IFELSE INTUBADO != 97
 THEN IFELSE EDAD >= 24 1 2
 ELSE IFELSE (OBESIDAD == 2) | (EMBARAZO <= 2) 2 1
or
IFELSE INTUBADO != 97
 THEN IFELSE ((NEUMONIA < 97) ^ (TABAQUISMO != 2)) &
        (EMBARAZO == 98) 2 1
```

```
 ELSE IFELSE (EMBARAZO <= 2) | (EDAD >= 101) 2 1


GP VEGA, Matriz de confusión [20600, 23003, 13353, 27567]

IFELSE CARDIOVASCULAR <= 2
 THEN IFELSE INTUBADO <= 2 1 2
 ELSE IFELSE (((EDAD >= 7) ^ ((TABAQUISMO == 2) ^
        (EMBARAZO == 99))) ^ ((EDAD <= 38) &
        (TABAQUISMO > 97))) | ((CARDIOVASCULAR <= 2) |
        (((HIPERTENSION == 98) & (SEXO != 1)) ^
        (CARDIOVASCULAR <= 98))) 1 2
 or

IFELSE ((CARDIOVASCULAR < 1) | (ASMA > 2)) |
        (INTUBADO >= 97)
 THEN IFELSE SEXO != 1 1 2
 ELSE IFELSE NEUMONIA <= 98 1 2
or

IFELSE OTRA_COM == 97
 THEN IFELSE ((INTUBADO <= 97) | (((((OTRO_CASO == 1) |
        (CARDIOVASCULAR <= 2)) | ((EDAD >= 7) &
        ((CARDIOVASCULAR <= 2) | (RENAL_CRONICA >= 1)))) &
        (EPOC == 1)) & (CARDIOVASCULAR <= 2))) &
        (NEUMONIA <= 98) 1 2
 ELSE IFELSE SEXO != 1 1 2
```

Las siguientes imágenes muestran que los algoritmos GP Simple y Agregación, tienen su mayor crecimiento en la función de aptitud dentro de las primeras 30 generaciones.




(a) (b)

Figura 5.12: Función de aptitud para Estado de México.

### 5.2.3. Veracruz

Para el caso de Veracruz, los parámetros de cada algoritmo se muestran en la tabla 5.13. Los resultados de la tabla 5.14, muestran que los métodos de programación genética presentan mayor sensibilidad, mientras que el valor del f-score es muy cercano entre todos.

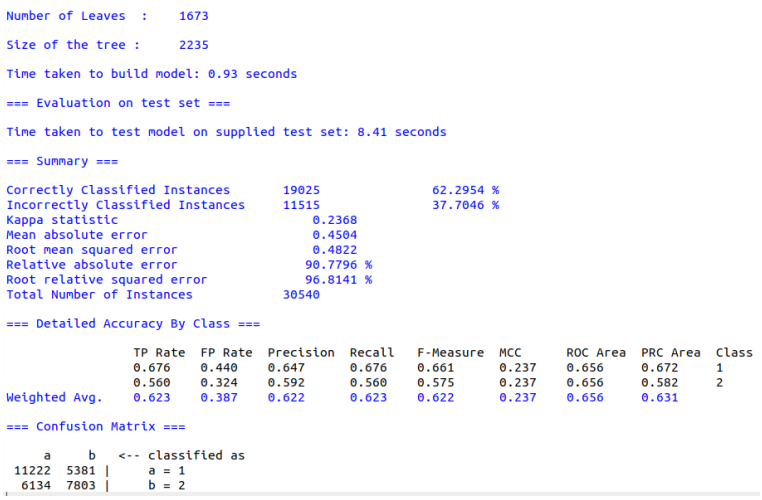
| Algoritmo | N. Individ. | N. Gener. | P. Mutación | P. Cruza | F. Aptitud | Selecc. |
|---|---|---|---|---|---|---|
| GP Simple | 60 | 100 | 0.25 | 0.8 | F-score | Ruleta |
| GP Agregación | 60 | 100 | 0.25 | 0.8 | .05,.5,.2,.1,.15 | Ruleta |
| GP VEGA | 50 | 80 | 0.25 | 0.8 | 5 objetivos | Elitismo |

Tabla 5.13: Parámetros en algoritmos para Veracruz.

| Algoritmo | Precisión | Fscore | Exactitud | Sensibilidad | Especificidad |
|---|---|---|---|---|---|
| GP Simple | 0.5754 | 0.7092 | 0.5881 | 0.9241 | 0.1878 |
| GP Agregación | 0.6145 | 0.6778 | 0.6095 | 0.7556 | 0.4355 |
| GP VEGA | 0.6057 | 0.6419 | 0.5860 | 0.6828 | 0.4705 |
| weka J48 | 0.647 | 0.661 | 0.623 | 0.676 | 0.560 |

Tabla 5.14: Evaluación sobre el conjunto de prueba para Veracruz.

Aunque en este conjunto de datos los valores de las métricas son similares, es importante señalar el tamaño de los modelos generados. Para C4.5 de weka, se generó un modelo de 2235 nodos, mientras que los modelos del presente trabajo son mucho menores; se muestran a continuación:



```
Number of Leaves  :     1673

Size of the tree :     2235

Time taken to build model: 0.93 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 8.41 seconds

=== Summary ===

Correctly Classified Instances       19025               62.2954 %
Incorrectly Classified Instances     11515               37.7046 %
Kappa statistic                          0.2368
Mean absolute error                      0.4504
Root mean squared error                  0.4822
Relative absolute error                 90.7796 %
Root relative squared error             96.8141 %
Total Number of Instances            30540

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.676    0.440    0.647      0.676    0.661      0.237    0.656     0.672     1
               0.560    0.324    0.592      0.560    0.575      0.237    0.656     0.582     2
Weighted Avg.  0.623    0.387    0.622      0.623    0.622      0.237    0.656     0.631

=== Confusion Matrix ===

     a     b   <-- classified as
 11222  5381 |     a = 1
  6134  7803 |     b = 2
```

Figura 5.13: Evaluación J48 en weka, Veracruz.

```
GP Simple, Matriz de confusión [2618, 11319, 1259, 15344]

 IFELSE (OTRO_CASO < 97) ^ ((OTRO_CASO < 2) ^
            (DIABETES > 99))
```

```
 THEN IFELSE (EDAD <= 28) & (OTRO_CASO < 97) 2 1
 ELSE IFELSE EMBARAZO == 99 2 1
```

GP Agregación, Matriz de confusión [6070, 7867, 4056, 12547]

```
  IFELSE EDAD <= 26
 THEN IFELSE (((DIABETES != 97) ^ (OTRO_CASO != 99)) ^
            (INTUBADO != 2)) & (INTUBADO < 1) 1 2
 ELSE IFELSE ((EDAD <= 39) ^ (OTRO_CASO == 98)) &
            ((OTRO_CASO == 2) ^ (EMBARAZO > 97)) 2 1
or

IFELSE ((ASMA < 1) & ((OBESIDAD == 1) |
            ((RENAL_CRONICA < 2) & (EDAD >= 51)))) |
            (EDAD >= 82)
 THEN IFELSE (EMBARAZO > 97) & ((RENAL_CRONICA >= 98) ^
            ((ASMA == 98) & ((((SEXO != 99) & ((SEXO > 2) |
            (((HIPERTENSION >= 99) ^ (EDAD >= 113)) |
            (((SEXO == 2) ^ (INMUSUPR >= 99)) | (SEXO >= 99))))) |
            (((OTRO_CASO < 1) ^ (((EMBARAZO != 99) |
            (OTRA_COM != 2)) & ((OTRO_CASO == 2) |
            (SEXO <= 99)))) | (EDAD >= 21))) ^ (EPOC != 1)))) 1 2
 ELSE IFELSE EDAD >= 42 1 2
```

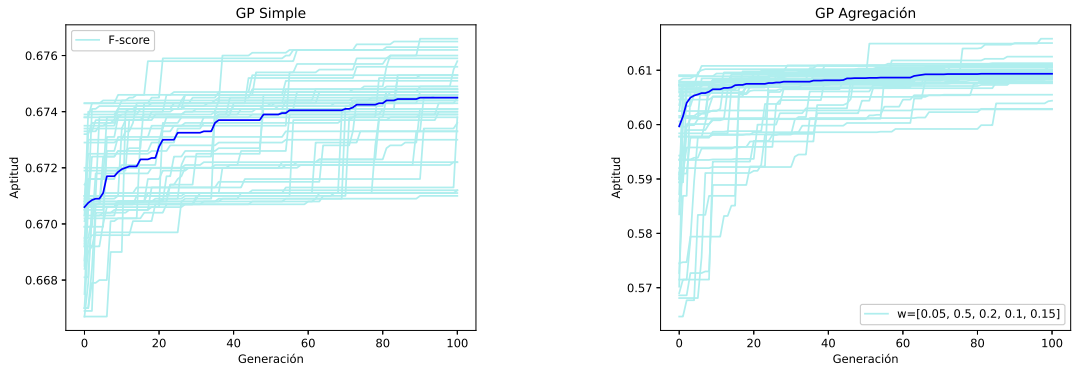
Aunque GP Simple, regresa el modelo más simple, sus evaluaciones no son tan buenas como en el caso de GP VEGA. En este último, el modelo también es sencillo de interpretar. Por la figura 4.3, se observa que hay muy pocos valores con UCI mayor o igual a 98 y por la primer regla se tiene que los que cumplen esta condición serán casos negativos, por otro lado si NEUMONIA menor o igual a 1, es decir, si tiene neumonía, entonces se clasifica como positivo. En la segunda regla, si se trata de un hombre de edad mayor o igual a 19 años, entonces es positivo, en otro caso es negativo; para las mujeres, si HIPERTENSION es 98, entonces son clasificadas como positivas. En la tercer regla, el valor de HIPERTENSION nunca es mayor a 99, por lo que si DIABETES es 1 (tiene diabetes) entonces se clasifica como positivo.

GP VEGA, Matriz de confusión [6559, 7378, 5265, 11338]

```
 IFELSE UCI >= 98
 THEN IFELSE UCI >= 98 2 1
 ELSE IFELSE NEUMONIA <= 1 1 2
or

IFELSE SEXO == 2
 THEN IFELSE EDAD >= 19 1 2
 ELSE IFELSE HIPERTENSION != 98 2 1
```

```
or

IFELSE HIPERTENSION > 99
 THEN IFELSE UCI >= 98 2 1
 ELSE IFELSE DIABETES != 1 2 1
```

La figura 5.14 muestra que el valor de la aptitud en GP Agregación crece más durante las primeras 30 generaciones, mientras que a GP Simple, le toma al menos 60 generaciones.




(a) (b)

Figura 5.14: Función de aptitud para Veracruz.

## 5.3. Síntomas en modelos de programación genética

En esta sección se muestra el conteo de las características de los pacientes y los síntomas que aparecen en las reglas obtenidas, por cada modelo de programación genética para cada uno de los conjuntos de datos. Se intenta analizar las características importantes que definen a la influenza y al COVID-19.

### 5.3.1. Influenza A(H1N1)

Como primer caso se muestran los síntomas para el caso de los pacientes de 0 a 6 años, los mejores modelos encontrados fueron para GP Agregación y GP VEGA, aunque el modelo basado en VEGA es mucho más pequeño que el basado en agregación, el número de síntomas o características que los modelos requieren para realizar la evaluación, son muy similares. Los valores que son comunes para ambos métodos son CONJUNTIVITIS, CONTACTOOTROSCASOS y MIALGIAS, por lo que se consideran características importantes en la detección de influenza para este grupo de edad; Figura 5.15.
En el caso de los pacientes de 7 a 15 años, figura 5.16, también GP Agregación y GP VEGA obtuvieron los mejores resultados y muy parecidos entre sí. Similar al caso anterior, el modelo generado por GP Agregación es mucho más grande que el de VEGA, salvo que en este caso

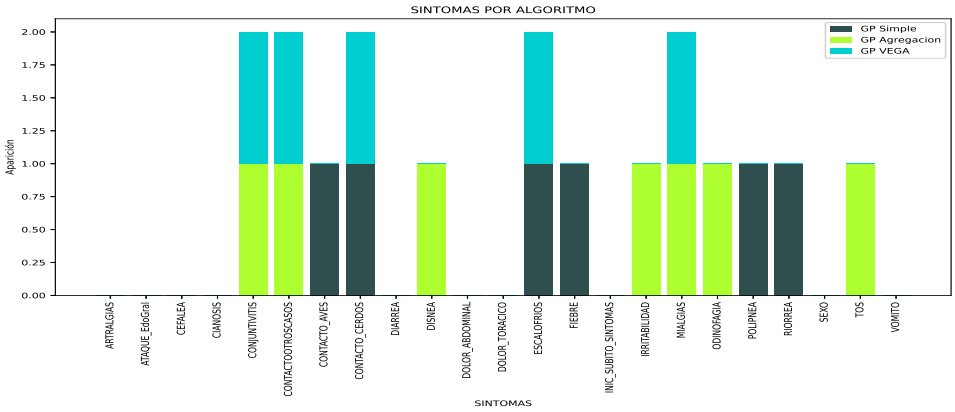



Figura 5.15: Síntomas de influenza detectados en pacientes de 0 a 6 años.

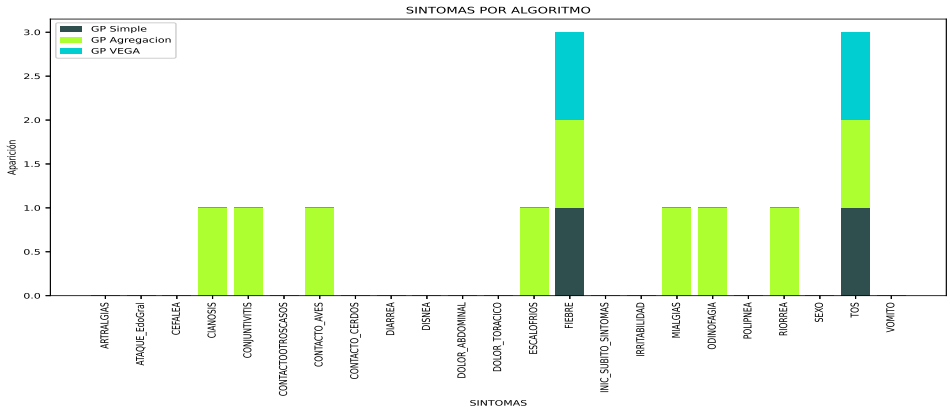



Figura 5.16: Síntomas de influenza detectados en pacientes de 7 a 15 años.

la diferencia se ve más marcada por el número de atributos que cada modelo utiliza. Por otro lado, los dos métodos mencionados y GP Simple, comparten los síntomas de FIEBRE y TOS, lo que los convierte en características importantes para detección de influenza en este grupo de edad.

Para el conjunto de datos de pacientes de 16 a 60 años, el modelo por GP Agregación es uno de los mejores junto con GP VEGA, aunque VEGA presenta un valor más bajo en especificidad, por lo que GP Agregación podría considerarse el mejor en este conjunto, tabla 5.6. En la figura 5.17 se puede observar que el modelo de agregación utiliza la mayor cantidad de atributos y que los más comunes entre los mejores modelos son CONTACTOOTROSCASOS, DISNEA y FIEBRE.

La figura 5.18 muestra las características seleccionadas en los modelos para los pacientes mayores a 60 años, donde al igual que para los pacientes de 16 a 60 años, GP Agregación generó el mejor modelo siendo este de mayor tamaño y con más atributos con respecto a GP VEGA y a GP Simple (figura 5.8). Se aprecia que CONJUNTIVITIS y DISNEA, fueron seleccionados por dos modelos,

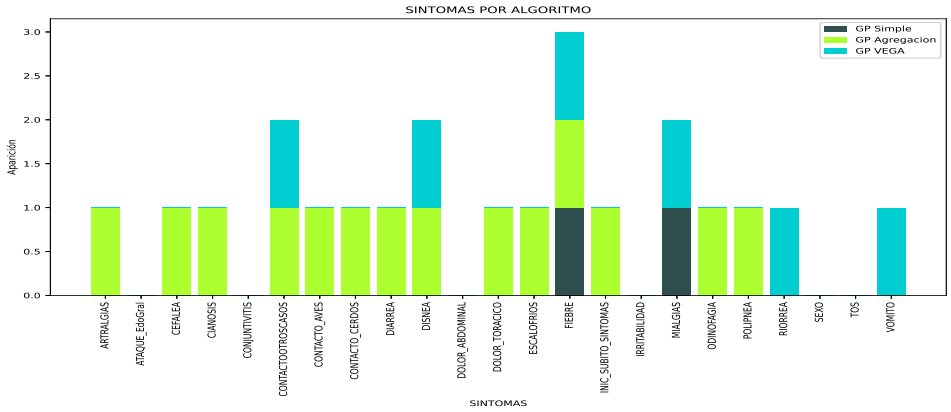



Figura 5.17: Síntomas de influenza detectados en pacientes de 16 a 60 años.

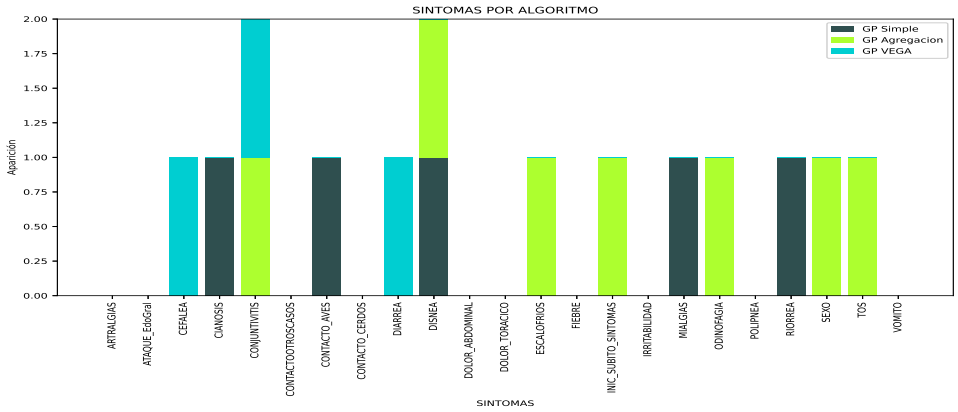



Figura 5.18: Síntomas de influenza detectados en pacientes mayores a 60 años.

por lo que se les podría considerar entre los más importantes para la detección de influenza.

### 5.3.2. SARS-CoV-2

Para el caso de SARS-CoV-2 (COVID-19) en Baja California Sur, los mejores modelos fueron para GP Agregación y GP VEGA, (figura 5.10), ambos modelos no tan pequeños aunque en 5.19 se observa que GP agregación utiliza un número menor de atributos con respecto a VEGA. Además, los dos atributos que fueron detectados por los tres modelos son NEUMONIA y OTRO_CASO, anexando ASMA, HIPERTENSION y OBESIDAD a los atributos más importantes pues fueron detectados por los dos mejores modelos.

En el Estado de México, como en los casos anteriores, GP VEGA y GP Agregación son los mejores modelos y con métricas de evaluación muy parecidas entre los modelos. Caso contrario

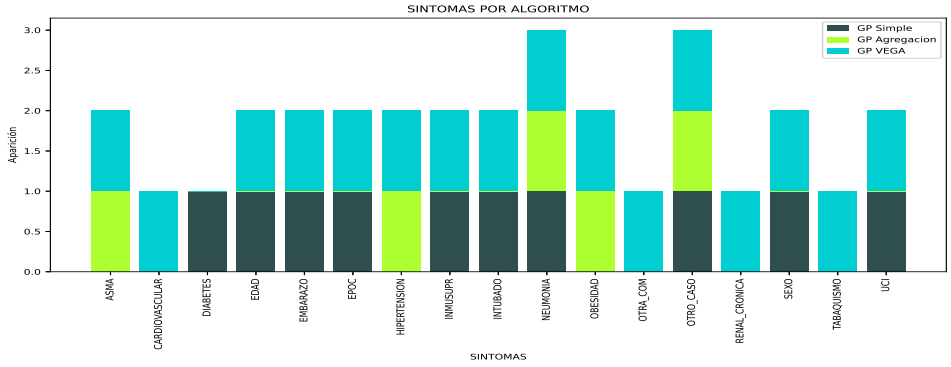



Figura 5.19: Síntomas de covid19 detectados en pacientes de Baja California Sur.

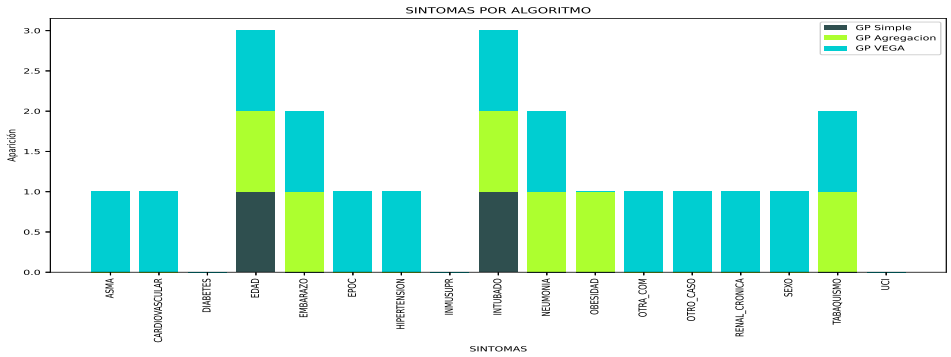



Figura 5.20: Síntomas de covid19 detectados en pacientes del Estado de México.

a los modelos de influenza, GP VEGA generó un modelo más grande que GP Agregación; en la figura 5.20 se observa que utiliza casi todos los atributos del conjunto de datos, siendo EDAD, INTUBADO, EMBARAZO, NEUMONIA y TABAQUISMO los más comunes entre los modelos.

Para el caso de Veracruz, 5.21, se observa que DIABETES y EDAD aparecen en los tres modelos, agregando SEXO e HIPERTENSION comunes entre los dos mejores modelos para este conjunto de datos.

### 5.3.3. Influenza y COVID-19

Una de las interrogantes importantes que surgen al desarrollar este trabajo, es saber si existen características o síntomas que estén presentes en ambas enfermedades para conocer la similitud entre ellas. Dado que los atributos que existen entre los conjuntos de datos no son compatibles, pues los datos de COVID-19 están mayormente asociados a comorbilidades, mientras que los datos de influenza son más sintomáticos, realizar este análisis no resulta sencillo o tan directo a partir de

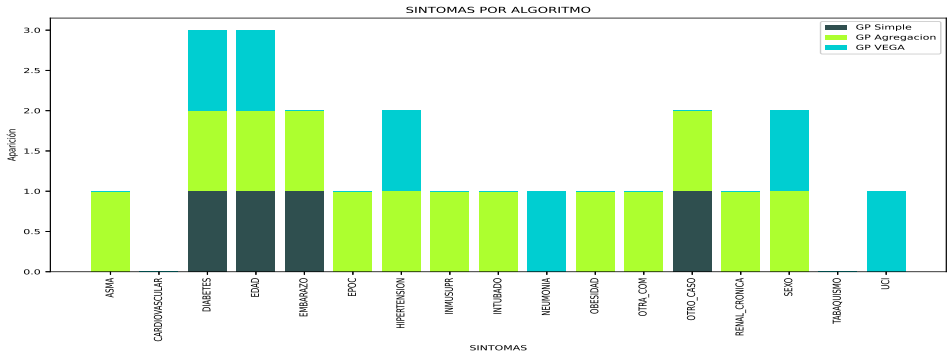



Figura 5.21: Síntomas de covid19 detectados en pacientes de Veracruz.

la información con que se cuenta. En esta sección se busca obtener un poco de información a partir de los mejores modelos encontrados de programación genética.

Entre los dos conjuntos de datos, los dos atributos que son comunes es SEXO y si el paciente tuvo o no contacto con un caso positivo, CONTACTOOTROSCASOS para el conjunto de influenza y OTRO_CASO para SARS-CoV-2. En las figuras 5.22, 5.23 y 5.24 se grafica el número de conjuntos de datos (los utilizados en el trabajo) en los cuales aparece un síntoma por cada uno de los modelos de programación genética, se observa que únicamente GP Agregación detectó el SEXO como característica utilizada en los modelos de predicción para ambos casos, influenza y covid19. De manera similar, si el paciente tuvo contacto con otro caso positivo, es común para ambas enfermedades en los modelos de GP Agregación y GP VEGA, lo que indica que ambas son enfermedades con un alto nivel de contagio.

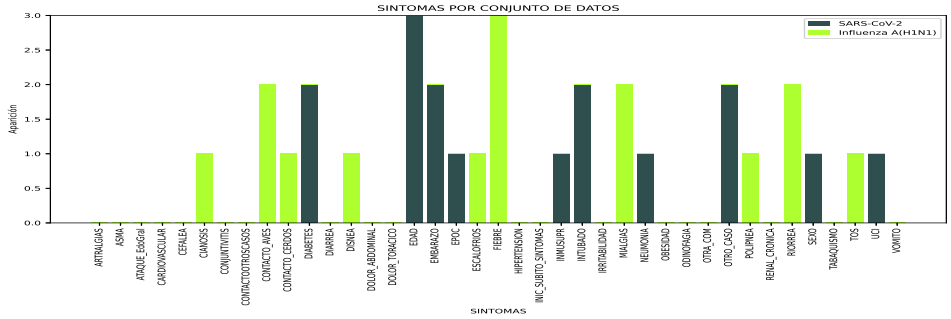



Figura 5.22: Síntomas en GP Simple

Además de los atributos anteriores, cabe señalar que hay enfermedades que pueden contener otros síntomas y que no están explícitamente dentro del conjunto de datos, como es el caso de la NEUMONIA en el conjunto de SARS-CoV-2, que puede generar FIEBRE, TOS y ESCALOFRIOS, síntomas que sí se encuentran explícitamente en el conjunto de influenza. Dado que los tres últimos atributos mencionados son detectados en los tres modelos junto con la NEUMONIA, y si siempre

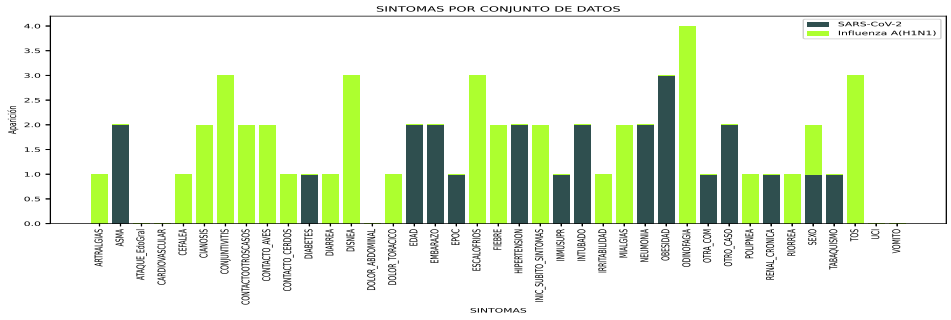



Figura 5.23: Síntomas en GP Agregación

que existe NEUMONIA se tienen esos tres síntomas, estos también podrían considerarse comunes entre las dos enfermedades.

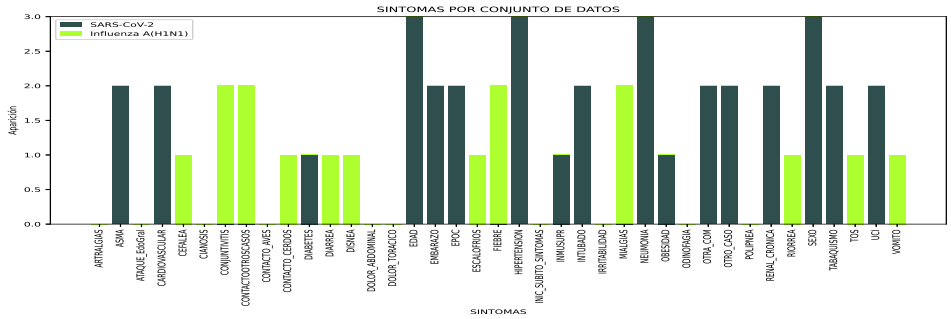



Figura 5.24: Síntomas en GP VEGA

Capítulo 6

# CONCLUSIONES Y TRABAJO FUTURO

En este trabajo se generaron modelos basados en programación genética para la detección de las enfermedades causadas por los virus de la Influenza A(H1N1) y el SARS-CoV-2. Dichos modelos se obtuvieron con datos reales de pacientes mexicanos, lo cual brinda un panorama amplio sobre los síntomas y las características que poseen las personas infectadas, con el fin de aportar conocimiento para detener la propagación de los virus entre las personas e incentivar la investigación computacional de los métodos evolutivos.

Como parte del desarrollo, se implementaron tres algoritmos basados en programación genética: GP Simple, GP Agregación y GP VEGA. De los tres, GP Simple no obtuvo tan buenos resultados como los otros, sin embargo, los tres generaron modelos que optimizaron su función objetivo en pocas generaciones, la mayoría en menos de 100. Además, GP VEGA y GP Agregación, generaron modelos cuyos valores en f-score y sensibilidad fueron mejores que el algoritmo C4.5 de weka (J48), cabe señalar que estos valores indican la efectividad de un clasificador para detectar correctamente los casos positivos, incluso cuando los conjuntos de datos presentan un desbalanceo de clases.

También el tamaño de los modelos obtenidos fue importante, pues el tamaño del modelo y el número de síntomas y características utilizadas en cada uno, fue mucho menor que el algoritmo J48. Añadiendo, la fácil interpretabilidad de algunos de los mejores modelos obtenidos por los algoritmos de programación genética.

Aún existe mucho trabajo e investigación por realizar dentro de este tema, por ejemplo, mejoras en el algoritmo de clasificación: probar el desempeño con otras gramáticas (añadir uno o más niveles de anidamiento en las cláusulas IF-THEN-ELSE), paralelizar el algoritmo de clasificación en un modelo de subpoblaciones en red o de anillo, etc. También se debe tener en cuenta, que la interpretación de las reglas y el desarrollo de todo el trabajo se realizó sin la intervención de un profesional de la salud, por lo que integrar y solicitar asesoría de una persona en este sector daría un mejor panorama y ayudaría en la comprensión de las reglas generadas.

Como se vio anteriormente, para el conjunto de datos de COVID-19 no se tomaron las 32 entidades federativas de México, por lo que el siguiente paso es generar el modelo para estas regiones y

ver si existe alguna relación que tenga que ver con la localización geográfica y/o sus actividades económicas. Ya que ambos conjuntos están creciendo constantemente, generar modelos con los datos más recientes es lo ideal, o particionarlos con respecto a otros objetivos particulares que puedan surgir, como el grupo de edad, la temporalidad en que se presentan las variantes de los virus, gravedad de las enfermedades, etc.

Esta Tesis supone un primer acercamiento con los conjuntos de datos trabajados, que con mejoras posteriores y supervisión de un profesional de la salud, podría ser parte del estudio clínico de los pacientes y tratarse como una ayuda en la detección de la influenza A(H1N1) y la COVID-19.

# Referencias


[1] F. A. A. *Data Mining and Knowledge Discovery With Evolutionary Algorithms*. Springer, 1998.

[2] E. L. Castillo. Selecci?n de caracter?sticas con algoritmos gen?ticos paralelos y sprint. *Tesis, Facultad de Ciencias, UNAM*, 2017.

[3] COVID-19 Tablero México CONACYT. https://datos.covid-19.conacyt.mx/, Mayo 2021.

[4] Fajardo-Dolc, Germán and Hernández-Torres, Francisco et al. Perfil epidemiológico de la mortalidad por influenza humana A(H1N1) en México. *Salud Pública de México / Vol 51, no 5.*, 2009.

[5] H. Jiawei, M. Kamber, and J. Pei. *Data mining : concepts and techniques*. Third edition, 2012.

[6] H. R. L. and S. E. Haupt. *Practical Genetic Algorithms*. John Wiley & Sons Inc, second edition, 2004.

[7] P. G. L. and A. A. Freitas. *Automating the Design of Data Mining Algorithms: An Evolutionary Computation Approach*. Springer, 2010.

[8] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008.

[9] M. Melanie. *An Introduction to Genetic Algorithms*. MIT Press, 1999.

[10] Z. Mohammed and W. Meira. *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press, 2014.

[11] M. Oded and L. Rokach. *Data Mining and Knowledge Discovery Handbook*. Springer, second edition, 2010.

[12] Organización Mundial de la Salud. ¿Qué es el virus gripal A(H1N1) 2009 pandémico? https://www.who.int/csr/disease/swineflu/frequently_asked_questions/about_disease/es/, Febrero 2010.

[13] S. K. Pal and P. Mitra. *Pattern Recognition Algorithms for Data Mining*. CHAPMAN & HALL/CRC, first edition, 2004.

[14] K. D. Patterson and G. F. Pyle. The geography and mortality of the 1918 influenza pandemic. *PMID: 2021692*, 1991.

[15] V. Rafael.V., B. H.J., and B. H. et al. Multiobjective grammar-based genetic programming applied to the study of asthma and allergy epidemiology. *BMC Bioinformatics 19*, 2018.

[16] J. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. pages 93–100, 01 1985.

[17] F. Usama, G. Piatetsky-Shapiro, and P. Smyth. The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 1996.