
# Desarrollo de un sistema de clasificación de eventos sismo volcánicos usando librerías de Machine Learning en Python

Vásconez Gómez, Francisco Sebastián

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Trabajo de titulación, previo a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones

Director: MSc. Larco Bravo, Julio Cesar

Sangolquí

30 de junio del 2020

URKUND

## Document Information

| | |
|---|---|
| **Analyzed document** | trabajo_titulacion_vasconez.pdf (D75308792) |
| **Submitted** | 6/19/2020 5:31:00 PM |
| **Submitted by** | |
| **Submitter email** | mgutierrez@difusion.com.mx |
| **Similarity** | 5% |
| **Analysis address** | mgutierrez1.GDC@analysis.urkund.com |

## Sources included in the report

| | | |
|---|---|---|
| **SA** | URL: Tesis Aguirre Gallegos Completa Version Final.pdf<br>Fetched: 6/18/2019 5:14:00 PM | 3 |
| **SA** | URL: Trabajo_Titulacion_Rosero_Jacome_Karen.pdf<br>Fetched: 12/18/2019 11:02:00 PM | 7 |
| **SA** | URL: Trabajo_Titulacion_Aguinaga_Diana_VF.docx<br>Fetched: 11/15/2018 9:02:00 PM | 16 |
| **SA** | URL: TESIS MARCELO VILLALVA.docx<br>Fetched: 12/19/2019 1:45:00 PM | 1 |
| **W** | URL: http://amsantac.co/blog/es/2016/09/20/balanced-image-classification-r-es.html<br>Fetched: 6/19/2020 5:34:00 PM | 2 |
| **W** | URL: https://towardsdatascience.com/feature-selection-using-wrapper-methods-in-python-f ...<br>Fetched: 6/19/2020 5:34:00 PM | 2 |

**Larco Bravo, Julio Cesar**

**DIRECTOR**

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

**DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES**

**CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES**

## CERTIFICACIÓN

Certifico que el trabajo de titulación, "**Desarrollo de un sistema de clasificación de eventos sismo volcánicos usando librerías de Machine Learning en Python**" fue realizado por el señor **Vásconez Gómez, Francisco Sebastián** el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí,30 de junio del 2020

**Larco Bravo, Julio Cesar**

C.C.: 1710638808


## DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

## CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

### RESPONSABILIDAD DE AUTORÍA

Yo **Vásconez Gómez, Francisco Sebastián**, con cédula de ciudadanía n°1804588653, declaro que el contenido, ideas y criterios del trabajo de titulación**: Desarrollo de un sistema de clasificación de eventos sismo volcánicos usando librerías de Machine Learning en Python** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

**Sangolquí, 30 de junio del 2020**

**Vásconez Gómez, Francisco Sebastián**

C.C.: 1804588653

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

## DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

## CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

### AUTORIZACIÓN DE PUBLICACIÓN

Yo **Vásconez Gómez, Francisco Sebastián,** con cédula de ciudadanía n°1804588653, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **Desarrollo de un sistema de clasificación de eventos sismo volcánicos usando librerías de Machine Learning en Python** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

**Sangolquí, 30 de junio del 2020**

**Vásconez Gómez, Francisco Sebastián**

C.C.: 1804588653

## DEDICATORIA

Las metas y logros alcanzados se los debo a mis padres que me mostraron un apoyo incondicional durante toda mi vida y en especial en esta etapa universitaria. Sus enseñanzas, su dedicación, sus virtudes son aspectos que han forjado la persona que soy ahora; la libertad y confianza que me dieron para seguir mi camino son el motivo de este éxito.

Gracias madre y padre.

## AGRADECIMIENTO

Agradezco a todas las personas que acompañaron mi recorrido en la etapa universitaria, a mis compañeros que se convirtieron en mis amigos con quienes superamos dificultades académicas juntos. A toda mi familia quienes me brindaron una sonrisa, alegrías, y ayuda sin importar la situación, en especial a mis tíos Jorge, Anita, a mi abuela Fanny, mi primo Joaquín, y mi hermano Mateo con quienes compartí 5 hermosos años de mi vida. Por ultimo a Cristina quien ha sido mi fortaleza, mi felicidad, y mi apoyo en todo momento.

Gracias a todos de corazón.

**ÍNDICE DE CONTENIDOS**

**ÍNDICE DE TABLAS**

## ÍNDICE DE FIGURAS

## RESUMEN

El Ecuador posee 31 volcanes activos lo que representa una amenaza para el país debido a que cerca de los volcanes se encuentran regiones densamente pobladas y en caso de una erupción miles de personas estarían en riesgo. Es por eso que el Ecuador tiene un sistema de alerta temprana (Ecuador, 2020), que consta de una red de sismómetros de banda ancha que brinda los datos obtenidos de ciertos volcanes como es el Cotopaxi, cuya monitorización es constante y de suma importancia para generar un aviso anticipado de los riesgos del volcán Cotopaxi. El Instituto Geofísico de la Escuela Politécnica Nacional (IGEPN) se encarga de monitorear, analizar y clasificar los Eventos Sismo Volcánicos (ESV) en el país, mismos que al aumentar su ocurrencia incrementa la probabilidad de una erupción volcánica. La gran cantidad de datos generada por los sensores generalmente se la analiza de forma visual por un operador, por lo tanto, este trabajo busca clasificar los ESV de manera automática mediante el uso de algoritmos de clasificación usando Machine Learning (ML) en un software libre como Python. La clasificación se realizará en base a 103 eventos de largo periodo (LP) y 101 eventos vulcano tectónicos (VT) utilizando seis algoritmos de clasificación distintos. También se realiza un estudio comparativo de los algoritmos implementados, con trabajos realizados utilizando Matlab en cuanto al mismo volcán. Por último, se realiza una prueba a los modelos de clasificación con una base de 1187 ESV. Los resultados obtenidos del mejor clasificador RF son exactitud del 95% y BER del 0.0052 utilizando la técnica de selección hacia adelante reduciendo las características a 30 de 84.

**PALABRAS CLAVES:**

- **MACHINE LEARNING**
- **CLASIFICACION DE ESV**
- **SISMICIDAD VOLCÁNICA**

**ABSTRACT**

Ecuador has 31 active vulcanoes which represents a threat to the country, due to the fact that densely populated regions are close to the vulcanoes and if an eruption occurs thousands of lives will be in risk. That is why Ecuador has an early warning system (Ecuador, 2020) which consists of a network of broadband seismometers that provides data obtained from certain vulcanoes such as Cotopaxi, this system is key for monitor the vulcano and for generate an early warning of an eruption. The Geophysical Institute of the National Polytechnic School (IGEPN) is responsible for monitoring, analyzing and classifying the Volcanic Earthquake Events (ESV), when the ESV increases the probability of a volcanic eruption is higher. The large amount of data generated by the sensors is generally analyzed visually by an operator, therefore, its analysis and classification becomes a complicated task, which takes a long time and is prone to errors, so this work seeks to classify ESVs automatically by using classification algorithms using Machine Learning (ML) in free software like Python. The classification will be made based on 103 long-period events (LP) and 101 vulcan tectonic events (VT) using six different classification algorithms. A comparative study of the implemented algorithms is also carried out, with work carried out using Matlab for the same vulcano. Finally, a test is performed on the classification models with a base of 1187 ESV. The results obtained from the best classifier RF are 95% accuracy and 0.0052 BER, using the forward selection technique reducing the characteristics to 30 of 84.

**KEYWORDS:**

- **MACHINE LEARNING**
- **CLASIFICATION OF EVS**
- **VOLCANIC SISMICITY**

# CAPÍTULO I

## 1. Descripción

### 1.1. Introducción

Existen unos 1,900 volcanes considerados activos en el mundo. El 60 por ciento de los volcanes activos se encuentra a lo largo del borde de las placas tectónicas y cerca del 90 por ciento de dichos volcanes se encuentran en el Anillo de Fuego o Cinturón de Fuego del Pacífico, esto se da debido a la colisión de las placas tectónicas de Sudamérica y Nazca. En el Ecuador hay 98 volcanes, de los cuales 31 están activos, potencialmente activos y en erupción. (GeoEnciclopedia, 2016)

El volcán activo más peligroso del Ecuador y uno de Iberoamérica es el volcán Cotopaxi (Notimerica, 2018). El cual se encuentra ubicado en la Cordillera Real (Oriental) con un diámetro de 20km y a una altura de 5897 m sobre el nivel del mar. Su peligrosidad radica en sus repetidas erupciones, la forma eruptiva, el glacial que lo cubre, y la gran cantidad de zonas pobladas cercanas al volcán que se encuentran bajo amenaza. A lo largo de la historia el Cotopaxi ha pasado por cinco periodos de erupción, los cuales presentan un ciclo repetitivo de 100 a 200 años su ultimo siendo en el año de 1880, por lo que es posible que en las siguientes décadas el volcán entre en un nuevo periodo eruptivo. Con las evidencias de anteriores erupciones se predice el nivel de destrozo que podría ocasionar con los enormes lahares, flujos y oleadas piroclásticas, y que afectara a más de 300,000 personas. Debido a esto, el volcán Cotopaxi posee la primera estación sísmica permanente a nivel de Sudamérica instalada en 1976. Hoy en día existe una red de Monitorización que posee equipos sísmicos, sensores, cámaras, radios y funciona 24 horas los 7 días a la semana. (IGEPN, Cotopaxi, 2017)

Por las razones expuestas es de suma importancia el Monitorización del Cotopaxi debido a todos los peligros que una erupción de este presenta, como: los gases volcánicos, que en contacto directo provocaría irritación en boca nariz y ojos de humanos y animales, destrucción de la vegetación, se

generaría lluvias acidas altamente corrosivas afectando a la agricultura, ganadería, y podría contaminar fuentes de agua potable. Flujos piroclásticos, son casi impredecibles y la probabilidad de supervivencia ante un impacto de estos es nula para zonas que se encuentran en el alcance como Refugio del Cotopaxi, Mudadero, Tambopaxi, Limpiopungo, y el Campamento Mariscal Sucre, se convierten en zonas de alto riesgo. Pero esto es el comienzo ya que dichos flujos serían los causantes del derretimiento de los glaciales formando enormes volúmenes de agua en pocos minutos formando los temidos lahares. Lluvia de cenizas, afectando a la agricultura con capaz desde 1mm de grosor, a la población con problemas de salud (irritación de ojos y de las vías respiratorias), problemas al ganado, daños a motores, contaminación de reservorios y fuentes de agua. El principal peligro los lahares que es lodo en conjunto con materiales sueltos, escombros, Si una persona es alcanzada por esto tiene pocas probabilidades de sobrevivir, la evacuación se debe realizar en las zonas que por donde pasan los drenajes naturales del volcán. Los lahares de gran tamaño pueden arrastrar puentes, vehículos, arboles, rocas, etc. La vegetación y edificios serán destruidos o afectados de gravedad. El Instituto Geofísico y de la Secretaría de Gestión de Riesgos muestra las zonas de riesgo Latacunga y sus parroquias Mulaló y Aláquez; Salcedo, el valle de Los Chillos, Rumiñahui y Tena serán las zonas a evacuar inmediatamente. (Daniel Andrade, 2005)

El volcán Cotopaxi es monitoreado sin descanso con una red de 23 sismómetros, 6 periodo cortos, 12 de banda ancha, y 5 de banda ancha con infrasonido (Hall, y otros, 2005). De igual forma el Cotopaxi es monitoreado con 4 sensores de banda ancha con telemetría digital y cámaras en caso de la producción de lava, lahares, lodo, o nieve (IGEPN, Cotopaxi Red Monitoreo, 2018).

Un sismógrafo se usa para medir la duración y amplitud de los sismos realizando un registro de la vibración del movimiento de las capaz telúricas, esto va conectado a un sistema de registro para guardar la información temporalmente. Los datos obtenidos del volcán Cotopaxi pasan en tiempo real al IGEPN, y con el Sistema Informático de Análisis y Procesamiento de Señales Sísmicas (SIPASS) los científicos y

técnicos pueden analizar los datos mediante espectros, aplicación de filtros, obtener y calcular parámetros de forma automática para el procesamiento y análisis de las señales obtenidas. (Viracucha & De la Bastida, 2014)

Por lo tanto, la presente investigación documenta el desarrollo de un algoritmo de clasificación de los ESV del volcán Cotopaxi. Las señales sísmicas son tratadas en el IGEPN, los científicos se encargan de su detección, clasificación y extracción de características de los eventos. De esta forma se dispone de una base de datos con 1187 ESV etiquetados y con 84 características propias de cada evento con la cual se trabajará. Debido a que se tiene una base de datos etiquetada se opta por usar un sistema supervisado de clasificación en ML, el proceso de ML es entrenar un modelo con datos de prueba, los cuales son ESV para así poder clasificar nuevos ESV (James, Witten, Hastie, & Tibshirani, 2015), con el fin de usar esta información para generar una alerta temprana en caso de una erupción volcánica.

Investigaciones realizadas sobre la clasificación de ESV usan ML como base, dichos trabajos varían los algoritmos utilizados para generar un modelo que logre clasificar correctamente los ESV. Algunos de los algoritmos empleados son, *Naive Bayes* (NB), *Artificial Neural Networks*, *k- Nearest Neighbor* (KNN), *Support Vector Machine* (SVM), *Decision Tree* (DT), *X- Gradient Boost* (XGB), entre otros. El presente trabajo de investigación emplea varios de estos modelos para elegir cual es el mejor clasificador, todos entrenados con la misma de base de datos y de igual forma se realizan pruebas con los mismos ESV. En este trabajo se usará un software libre Python para su desarrollo, de forma que no se necesite ningún programa que requiera licencia para su uso. Con los resultados de las clasificaciones se espera incrementar los parámetros de rendimiento en la clasificación de ESV desarrollados usando software con licencia y de esta forma aportar en la ejecución de una alerta temprana basado en software libre que pueda ser usada en todo el país en especial en las zonas aledañas al Cotopaxi y así disminuir el riesgo para las personas en caso de una erupción.

La base de datos con la que se trabajara consta de 1187 señales sísmicas etiquetadas por el IGEPN, cada ESV en la base de datos posee 84 características como entropía, kurtosis, desviación estándar, energía, los picos obtenidos, etc. Estas características son de suma importancia para la presente investigación, ya que con ellas se entrena a los algoritmos de clasificación para que se logre clasificar ESV futuros.

Este trabajo de investigación se ha desarrollado con el apoyo del IGEPN y bajo la tutela de los docentes anexados al proyecto de investigación "SRASI - Desarrollo de un sistema de clasificación de eventos sísmicos usando librerías de ML y Python" registrado con número N.- ESPE –RES-IETL-2019-028 en la Universidad de las Fuerzas Armadas ESPE.

### 1.2. Justificación e Importancia

El Ecuador posee la famosa ¨ruta de los volcanes¨ un emblema turístico del país, pero dichos paisajes guardan un peligro constante, la actividad de estos bolsones de magma que pugnan por salir causa nerviosismo, temor, y angustia en la población. La reactivación de uno de sus más peligrosos volcanes el Cotopaxi provoca en la ciudadanía un estado de alerta constante, los lahares provocados por una erupción pondrían en riesgo de muerte a 100,000 personas y las perdidas serian minimas serian de 36,000 millones de dólares, dice Theo Toulqueridis vulcanólogo experto. (NationalGeographic, 2015). Debido a las consecuencias destructivas de los lahares, magma, ceniza, lluvia acida, etc, su monitorización es fundamental para el bienestar de los habitantes cercanos y así como para los recursos del país. Al monitorear el Cotopaxi se obtiene una gran cantidad de datos que son revisados de forma semiautomática, con la detección de los eventos de manera automática y la clasificación de forma visual con la previa extracción de características fundamentales como la frecuencia de los sismos. La parte de clasificación al ser de manera manual consume demasiado tiempo para el científico o tecnólogo por lo que su revisión esta sustenta a fallos.

La evolución de las redes de telecomunicaciones, sensores, transmisión de datos en tiempo real, predicciones, etc., se debe a los nuevos diseños e implementaciones de algoritmos que tienen como finalidad satisfacer la demanda y calidad de vida del ser humano. Con esta ayuda la tecnología colabora con trabajos repetitivos hechos por personas, por lo que apoyarse en ella es la mejor solución, ya que se puede hacer uso las 24 horas y tiende a menos fallas. (C. Jiang, 2017) Esto ha llevado al desarrollo de sistemas que se encuentren en la capacidad de tomar decisiones por si solos mediante la utilización de diferentes algoritmos de ML los cuales utilizan diferentes tipos de algoritmos como clasificación y regresión.

El algoritmo de clasificación posee tres ramas la supervisada, no supervisada, y de refuerzo. El algoritmo supervisado se usa cuando se tiene datos etiquetados y se intenta clasificar un evento usando las características de los datos. En el algoritmo no supervisado se usa datos no etiquetados y se intenta agrupar los mismo por datos similares basados en características impuestas. Cada uno de ellos posee varios algoritmos como: Arboles de decisión (DT), NB, ANN, KNN, SVM, XG, entre otros. En base a trabajos previos y la investigación de estos algoritmos se seleccionará el mejor algoritmo para trabajar. (Portilla, 2019)

En varias investigaciones sobre la clasificación de eventos volcánicos como en (Benitez, Paillacho, Rojo, & Lara, 2017) (Lara, Benítez, Carrera, Ruiz, & Rojo, 2016) (Mario Ruiz, 2016), (Valeria Paillacho, 2016) el software usado es MatLab, que para su uso requiere licencia por lo tanto no puede ser distribuido a entidades públicas, ni libremente a los usuarios. El uso de software libre se ve requerido para estas investigaciones, ya que los usuarios tienen la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software (Python, 2012). El uso de software libre posee la libertad de ejecutar el programa como se desee, con cualquier propósito, por lo tanto, es necesario el uso de este para la clasificación de eventos del volcán Cotopaxi. En este trabajo se realiza la clasificación de ESV en Python debido a que

posee un soporte de cálculo, editores avanzados de texto, velocidad para la programación y la depuración de código, ventanas de gráficos y datos, además en los últimos años ha tenido un gran impacto en Inteligencia Artificial y ML (Pythones, 2017).

Estas herramientas permiten desarrollar un sistema de clasificación de eventos del volcán implementado algoritmos para clasificar eventos en el menor tiempo posible de manera óptima y eficaz. La implementación de este sistema con un software libre es importante debido a que no existen muchas referencias a estudios realizados en el país con Python enfocados a la clasificación de eventos sismo volcánicos.

### 1.3. Alcance del Proyecto

El presente proyecto de investigación busca aportar un algoritmo de clasificación de ESV realizado en un software libre Python para que pueda ser distribuido y usado sin inconvenientes, de igual forma se pretende apoyar a "SRASI - Implementación de un Sistema de Reconocimiento Automático de Señales Sísmicas del Volcán Cotopaxi", dentro del área de la clasificación de los ESV en obtenidos a partir de los sismógrafos del volcán Cotopaxi.

La etapa de experimentación se la realiza gracias a la base de datos entregada por el IGEPN que consta de señales clasificadas previamente. Debido a que se tiene una base de datos etiquetadas se usara los algoritmos de clasificación supervisados, el cual permite entrenar varios modelos cada uno con un algoritmo distinto de clasificación. Para entrenar a estos modelos se usa un porcentaje de la base de datos ya etiquetada, no se pude ocupar el 100% de dicha base debido a que el resto se necesita para calcular las medidas de rendimiento, que son: precisión (P, del inglés *precision*), exactitud (A, del inglés *accuracy*), especificidad (S, del inglés *specificity*), sensibilidad (R, del inglés *recall* o *sensitivity*) y tasa de error balanceado (BER, del inglés *Balanced Error Rate*) que se detallaran más adelante. Gracias a estas medidas

se puede escoger el mejor modelo para ser usado en la clasificación de nuevos ESV producidos por el volcán Cotopaxi.

## 1.4. Objetivos

### 1.4.1. General

Desarrollar de un sistema de clasificación de Eventos Sísmico Volcánicos usando librerías de Machine Learning en software libre.

### 1.4.2. Específicos

- Investigar y analizar algoritmos existentes de Machine Learning aplicados a la clasificación de eventos.
- Determinar las librerías de Python que permitan implementar algoritmos de Machine Learning para la clasificación de eventos sismo volcánicos.
- Desarrollar un sistema que permita clasificar eventos sismo volcánicos usando Machine Learning en Python.
- Analizar y comparar los resultados obtenidos de los algoritmos de Machine Learning implementados con trabajos previos.
- Proporciónar un sistema autónomo y entrenado de manera eficiente.

## 1.5. Trabajos Relacionados

La clasificación de eventos volcánicos siempre ha sido un área de interés para la ciencia, por lo que existen varios trabajos relacionados a la clasificación de ESV. Gracias al IGEPN se han realizado algunos estudios sobre el volcán Cotopaxi donde usan varios métodos, algoritmos, y procesos distintos, todos buscando mejorar los parámetros de rendimiento del proceso de clasificación de ESV.

En el trabajo (Lara, Carrera, Benítez, Ruiz, & Rojo, 2016) se realiza un sistema de detección y clasificación automática de los ESV del volcán Cotopaxi, que considera la extracción de características y las etapas de selección de características, para reducir el tiempo de procesamiento hacia un sistema confiable de alerta temprana de volcán en tiempo real (RT-VEWS). Con dichas características usan el algoritmo de SVM y DT, obteniendo un 97% de P y R en la etapa de clasificación, estos realizados en el dominio del tiempo y la frecuencia.

Un estudio realizado en Italia (Hajian, Cannavò, Greco, & Nunnari, 2019) tratan con dos modelos de clasificación, DT y KNN, implementados con una base de datos del Mount Etna (Italia), en su base de datos logran recopilar datos en 2 estados del volcán, cuando está tranquilo y en una erupción estromboliana. Donde los resultados muestran que DT logra un 87.92% de P en estado tranquilo y 99.51% de S en erupción, y KNN un 83.70% en P estado tranquilo y 99.84% de S en erupción, siendo estos los mejores resultados.

Otro estudio realizado en Peru (Malfante, Mura, Métaxian, & Mars, 2018) estandariza las señales a una frecuencia de muestreo de 100 Hz, luego procede a una etapa de extracción de características con un vector de 102 caracteres que describen la señal en tiempo y frecuencia, para finalizar con estas características pasan a la etapa de clasificación donde se genera un modelo con SVM, donde se construye un hiperplano en el espacio de características, separando así los datos en clases. El hiperplano se elige para maximizar el margen, que es la distancia entre el hiperplano y los vectores de soporte. Este sistema de predicción utiliza un registro que contienen 109,434 ESV adquiridos del volcán Ubinas (el volcán más activo en Perú). El nuevo modelo propuesto se construye utilizando algoritmos y alcanza el 92.2% de la clasificación correcta en seis clases.

El estudio de (Benitez, Paillacho, Rojo, & Lara, 2017) presenta un sistema automático basado en ML usando algoritmos de reconocimiento de señales, en la etapa de clasificación se usa el algoritmo de

SVM multiclase junto a la función de kernel. Y aplican el método de *Hyperplane Separability* y *Trade-off factor C* para la optimización de la clasificación. La etapa de clasificación alcanzó el 90% de P. Los parámetros óptimos que maximizan la clasificación del rendimiento fueron el núcleo lineal, con una compensación de 10 a 80, y la optimización mínima secuencial.

El estudio de (Ren, Peltier, Ferrazzini, Rouet-Leduc, & Jhonns, 2018) se basa en datos obtenidos del volcán Piton de la Fournaise (La reunión de islas), trabajan en una banda de 25 frecuencias en un rango espectral de 0.5 – 26 Hz con espaciamiento de 1 Hz, esto para generar una ventana que será usada en el algoritmo de *Gradient Bosst Decision Trees* (GBDT). Usan un 70% de los datos para entrenar al modelo y el resto para pruebas. Al realizar los test en el modelo se obtienen como resultados una P de 0,99, una A de 0,97 y un R de 0,83.

Una obtención de datos muy particular se muestra en (Witsil & Jhonnson, 2019) se usa las cámaras como herramienta de Monitorización, los algoritmos automatizados de visión por computadora captan las imágenes en bruto y las pasan a señales de series temporales relevantes. Luego se usa un algoritmo de detección de gotas para resaltar la actividad observable en el Volcán Villarrica, Chile. Con estas señales pasan a la clasificación donde se usa una ANN, este es un conjunto de redes conectadas entre sí en las que el peso asignado varia con cada conexión, por lo que el aprendizaje se va dando en sus capas intermedias al momento de reajustar los pesos de las conexiones. El ANN usado es supervisado y sirve para clasificar la actividad observable del volcán en cinco clases, logrando una P del 92%.

El leer e informarse de trabajos anteriores para la clasificación de ESV sirve como base para el presente trabajo de investigación, se aprende que métodos han tenido mayor porcentaje de aciertos en la clasificación, los softwares más utilizados, y herramientas que serán de utilidad para el trabajo. Es de suma importancia recalcar la importancia en esta investigación el uso de un software libre ya que la mayoría de trabajos previos utilizan softwares bajo licencia.

### 1.6. Organización del Trabajo

Para realizar la investigación y con el objetivo de una correcta implementación del sistema las siguientes tareas serán llevadas a cabo. Realizar un estudio de los algoritmos de ML aplicados a sistemas de clasificación supervisada y no supervisada que puedan aportar a un desempeño óptimo del sistema, desarrollando e implementando diferentes algoritmos de ML estudiados para la clasificación de las señales volcánicas. Prosiguiendo con el análisis de resultados de los algoritmos empleados para la clasificación de señales, comparando A, P, R, y S, (Ghoneim, 2019) y la comparación de resultados con técnicas aplicadas de ML en la clasificación usando el software de Matlab vs un software libre. Finalizando el trabajo con la preparación y redacción de reporte técnico y la respectiva documentación del trabajo de titulación. La investigación realizada presenta cinco capítulos que son organizados de manera que exista una comprensión lectora del tema en todo momento.

# CAPÍTULO II

## 2. Marco Teórico

### 2.1. Vulcanología

#### 2.1.1. Conceptos Básicos

El volcán es el resultado de un largo proceso geológico que aflora a la superficie terrestre, el proceso de emisión de los gases y magma a través de la corteza se conoce como vulcanismo, y la ciencia que estudia esto se denomina vulcanología. (Sieron, 2015) Al pasar de los años el planeta ha ido variando sus paisajes por la salida del magma del interior del centro de la tierra lo que fue dando origen a seis tipos de volcanes:

- Estratovolcán, formado por capas sucesivas de lava, piroplastos y roca, su forma es cónica con un cráter central y muy grandes tanto así que sus picos nevados son superiores a los 1000 km cuadrados de superficie, 2500m de altura, y 400 km cúbicos en volumen.
- Volcán en Escudo, es el resultado de lava fluida con la superposición de ríos con una pendiente de máximo 7 grados, son volcanes de inmensos tamaños donde la lava se va derramando por grupos de orificios de ventilación.
- Calderas, una fuerte erupción causa el derrumbe parcial o total de todo el edificio volcánico que como resultado deja un inmenso cráter o caldera.
- Volcán Somma, edificio volcánico creado sobre la caldera de otro volcán.
- Volcán Tuya, su formación se da cuando la lava sale por debajo de un glaciar, en el momento que la lava y el hielo interactúan se va formando capaz de hyaloclastito dando origen a un volcán con flancos casi verticales.
- Cono escoria, formados sobre un ventiladero so el resultado de ceniza, rocas, y fragmentos de lava, su tamaño es pequeño y no sobrepasa los 1000m de altura. (Sieron, 2015)

Existen cinco tipos de erupciones volcánicas clasificadas por la viscosidad del magma:

- Erupción Pliniana, explosión de mínimo 20km sobre el nivel del cráter, caracterizada por alternar erupciones con piroclastos y coladas lávicas, causando que arroje grandes cantidades de gases, piedra pómez, y ceniza, logrando cubrir un amplio terreno con ceniza.
- Erupción Vulcaniana, provocada por una fragmentación pequeña en el magma que al contacto con el agua se produce una erupción violenta arrojando ceniza, vapor, y materiales fragmentarios.
- Erupción Peleana, expulsa lava viscosa junto con nubes de gases a altas temperaturas, que destruyen todo a su paso.
- Erupción Estromboleana, explosión de lava fluida que pueden llegar a cientos de metros del cráter, acompañados de enormes cantidades de gases.
- Erupción Hawaiana, explosión más tranquila de los eventos debido a que son erupciones efusivas de lava, el magma es de tipo basalto y producen pocos gases y poca ceniza. (INPRES, 2016)

Cualquier tipo de erupción volcánica acarrea muchos riesgos y peligros, las grandes erupciones y sus consecuencias pueden afectar miles de personas con daños materiales, a los cultivos, el ganado, e incluso pone en riesgo a las vidas humanas. Ahora se hablará de los peligros ante una erupción volcánica.

- Nubes y columnas eruptivas, se da en erupciones explosivas donde fragmentos de roca fundida y sólida, junto con gases volcánicos son expulsados con una increíble fuerza. Los fragmentos grandes se les denomina bombas y pueden llegar hasta 4km del cráter y los fragmentos pequeños de minerales, vidrio volcánico y ceniza, formando las columnas eruptivas que se elevan por los aires y alcanzan más de 20 km.

- Gases Volcánicos, estos gases son expulsados antes, durante, y luego de la explosión. Su composición es casi todo vapor de agua, sin embargo, contiene otros gases peligrosos para los seres vivos, como dióxido de carbono, azufre, ácido clorhídrico, entre otros. El peligro de los gases radica en que se pueden mezclar con el agua atmosférica causando lluvias acidas, que pueden destruir cultivos, ganado, afectar el agua potable, y afectar a la piel humana.
- Flujos de Lava, el magma que emerge a la superficie a través del cráter o por una fisura del volcán forma los flujos de lava. El peligro de estos flujos radica en si los volcanes poseen glaciares, el contacto entre ambas superficies generaría una transferencia de calor lenta que al solidificarse se derrumbaría provocando flujos piroclásticos cuya temperatura supera los 500°C.
- Flujos Piroclásticos, son avalanchas de gases, fragmentos de roca, y ceniza, que descienden por los flancos del volcán a velocidades de 160-240 km por hora. Son impredecibles y ocurren durante toda la erupción, por la temperatura y velocidades que alcanzan son tan peligrosos que derriban y queman todo por donde pasan.
- Derrumbes, el hielo, nieve, y/o rocas desprendidos se convierten en escombros que bajan la pendiente del volcán ganando velocidad y tamaño, pueden variar desde pequeños escombros hasta colapsos de la cima o flancos del volcán. Estos deslizamientos se generan cuando la lluvia intensa, terremotos, o erupciones de gran magnitud causan que dichos materiales se desprendan y caigan por la pendiente.
- Lahares, la mezcla de los flujos de roca, arena, lodo con agua de los glaciares, lagos o fuertes lluvias generan los lahares. Las desembocaduras naturales como quebradas, valles, ríos son los lugares por donde bajan estos flujos de escombros y van arrasando todo lo que los drenajes

naturales tengan en frente, aquí radica su peligro ya que muchas comunidades viven rio abajo de volcanes con glaciares.

### 2.1.2. Ecuador y sus volcanes

Ecuador se encuentra sobre el "Anillo de Fuego" del Pacifico y atravesado por la "Cordillera de los Andes" separando al país en tres zonas geográficas, Costa, Sierra y Amazonia, generando una enorme cantidad de paisajes únicos, diversidad de flora y fauna, y con atractivos naturales únicos de cada región, lo que hace del Ecuador un país turístico por excelencia. Uno de los principales atractivos turísticos de la sierra es la famosa "Ruta de los Volcanes", esto debido a que el Ecuador es el tercer país con más volcanes en el mundo contando con 84 ejemplares de ellos 33 considerados activos. (GoRaymi, 2016)

Estos 33 volcanes presentan posibilidades de erupciones que son uno de los espectáculos naturales más grandes de la tierra, pero atrás de su belleza se esconde un peligro muy alto, una explosión produce cambios que pueden modificar la tierra, contaminar aguas, y hasta pueden cambiar el clima. El volcán Sangay es uno de los más activos del mundo se encuentre en erupción desde 1628. El Tungurahua y el Reventador que se activaron en el año 1999 y 2002 respectivamente con un lapso de explosiones de más de una década. El Cotopaxi que en 2015 reactivo su actividad volcánica es uno de los más peligrosos del mundo debido a la cantidad de glaciales que posee, poniendo en peligro la vida de 300,000 personas. Estos algunos ejemplos de los volcanes activos del país, pero se esconde un peligro aún más grande y es el super volcán Chalupas, cuya erupción podría acabar con toda la vida sobre el Ecuador, su última erupción se calcula que fue hace 200,000 años y en la actualidad no existe probabilidad de que ocurra de nuevo. (Astronoo, 2012)

Por la cantidad de actividad volcánica del país y por el riesgo a una posible erupción el Ecuador posee a un instituto encargado del Monitorización de los volcanes es el IGEPN, que tienen la labor de avisar con tiempo a la población si se da una erupción. Los volcanes presentan ciertas anomalías antes de

una erupción las cuales son monitoreadas con varios sensores por parte del IGEPN. Las anomalías y la tecnología usada son incrementos en la actividad sísmica debido al magma en ascenso donde se usan sismómetros para detectar el movimiento de fluidos, variación en la presión de la atmosfera del volcán para lo que se usa barómetros, detectores de gases para registrar la desgasificación del magma al acercarse a la superficie, inclinometros colocados en los flancos para detectar la hinchazón o deflación por el paso del magma, y sensores de frecuencia modulada para detectar flujos piroclásticos o lahares. El numero materiales y sensores usados para cada volcán va en directa relación al peligro que signifique para la población. (IGEPN, Clasificacion de Volcanes en el Ecuador, 2019)

### 2.2. Actividad Sísmica

Los patrones de la actividad sísmica son el principal propósito para realizar un estudio de la sismología volcánica, ya que la ocurrencia de dichos sismos son los primeros precursores y más fácilmente detectables que permiten establecer con tiempo la probabilidad de una erupción. (UNESCO, 1998)

La ascensión o descensión del magma produce sismos volcánicos que se relacionan con la reactivación del volcán antes de entrar en erupción. Los sismos producen ondas las cuales viajan a través del volcán para ser detectada por los sismógrafos. (UNESCO, 1998)

Dichas ondas sísmicas son ondas elásticas generadas por sismos que se propagan en todas direcciones desde su origen. (IGEPN, Glosario, 2014). Estas ondas viajan tanto por el interior como por la superficie de la tierra y se las conoce como ondas de volumen y ondas superficiales. Casi al finalizar la onda sísmica los rezagos que quedan de la señal se los conoce como Coda.

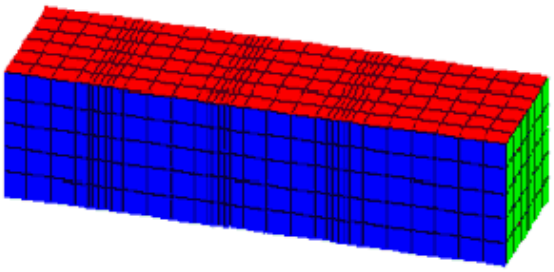
Las ondas de volumen se dividen en dos, ondas primarias (*ondas P*) y secundarias (*ondas S*), ver Figura 1 y Figura 2respectivamente. Las ondas primarias son ondas longitudinales, las cuales consisten en la transmisión de rarefacciones, comprensiones, y dilataciones del suelo en la dirección propagada. Atraviesan gases, líquidos, y solidos por ser comprensibles, debido a esto viajan 1.73 veces con mayor

velocidad a las *ondas S*. Por otro lado, las *ondas S* son ondas transversales donde desplaza el suelo en dirección perpendicular a la dirección de propagación. Estas ondas solo se propagan a través de sólidos y su amplitud es mayor a las *ondas P* como se observa en la Figura 3 (Hernandez, 2017).

Las ondas superficiales son aquellas que se propagan por las capas más cercanas a la superficie de la Tierra, estas son las que tienen efectos más catastróficos, existen de 2 tipos ondas Rayleigh (ondas R) y ondas de Love (ondas L). Las *ondas R* se originan y viajan a lo largo de la superficie de la tierra debido a que la tierra es un sólido con una superficie libre, su forma es elíptica retrograda como se muestra en la Figura 4. Las *ondas L* se forman en un medio estratificado, en otras palabras, en la interfase de dos medios con propiedades químicas y físicas diferentes, su movimiento es el mismo que las *ondas S*, perpendicular a la dirección de propagación, con la característica que solo tienen componentes horizontales a la superficie como se observa en la Figura 5. (Coruna, 2017)

**Figura 1**

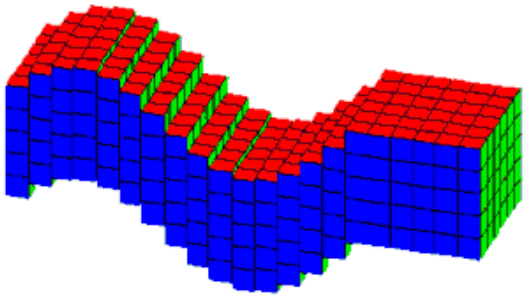
*Propagación de Ondas P*



**Fuente:** (Coruna, 2017)

**Figura 2**

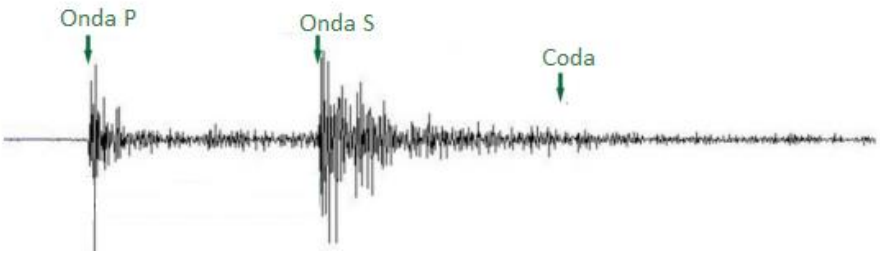
*Propagación de Onda S*



**Fuente:** (Coruna, 2017)

**Figura 3**

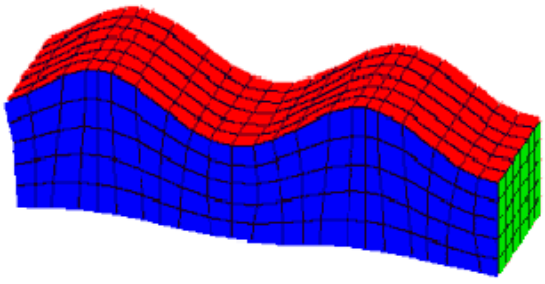
*Amplitudes de ondas P, S, y Coda*




**Fuente:** (Coruna, 2017)

**Figura 4**

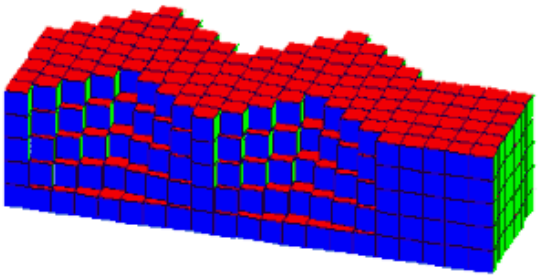
*Propagación de onda R*



**Fuente:** (Coruna, 2017)

**Figura 5**

*Propagación de onda L*



**Fuente:** (Coruna, 2017)

Por último, la Coda es la vibración residual que forma el final de una señal sísmica registrada por el sismógrafo, estas ondas son retro dispersadas en las heterogenidades de la corteza y el manto superior, por eso se va reduciendo paulatinamente hasta confundirse con ruido. (Hernandez, 2017)

### 2.2.1. Señales sísmicas volcánicas transitorias

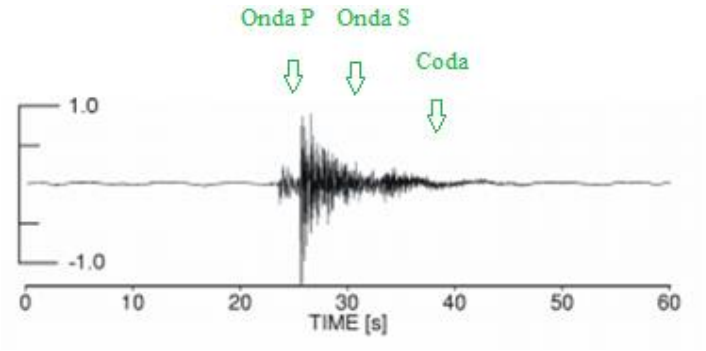
Los tipos de eventos sísmicos son de suma importancia detectarlos, identificarlos, y clasificarlos ya que un aumento o disminución de los mismos es un indicador de movimiento de fluidos dentro del volcán, movimientos telúricos y posibles erupciones. (Perez D. , 2017). Acto seguido se describirán los eventos de señales sísmicas volcánicas los cuales son: Híbridos (HB), Largo Período (LP), Tremores Volcánicos (TRE), y Vulcano Tectónicos (VT)

#### a. Eventos Vulcano Tectónicos

Existen dos eventos VT el tipo A y B. La principal característica de los eventos tipo VT-A es un gran impulso de la *onda P* seguido por la *onda S*, las cuales son fácilmente reconocibles en el sismograma. También conocido como eventos de alta frecuencia debido a que su espectro es mayor a los 5 Hz, ver Figura 6. (Perez D. , 2017)

**Figura 6**

*Ejemplo del evento VT, del volcán Merapi (Indonesia)*




**Fuente:** (Wassermann, 2012)

Estos eventos se forman debido a la baja atenuación en las cortas trayectorias de las ondas, donde el medio tiene una alta reverberación causando así las altas frecuencias y los impulsos de las *ondas P* y *S*. Sus profundidades van a partir de los 2km hasta alcanzar la base del edificio volcánico, sus magnitudes alcanzan hasta los 4 grados en la escala de Richter. (Perez D. , 2017)

Como se muestra en (Wassermann, 2012) los eventos tipo VT-B muestran un arribo de mayor intensidad respecto a las *ondas P* y las *ondas S* casi no se logran distinguir, sus frecuencias son bajas y van desde los 1Hz hasta los 5Hz. Se originan a profundidades entre 1 y 2 km, y su arribo y frecuencias son ocasionadas por la poca profundidad de su origen.

**b. Eventos de Largo Período**

La mayor parte del tiempo estos eventos tienen arribos emergentes de la *onda P* continuados por la *onda S*, por lo que no se distingue el inicio de la *onda S*, su frecuencia va entre los 0.2 Hz y 3 Hz por esto igual son conocidos como eventos de baja frecuencia, ver Figura 7, su profundidad es menor a los 2km en

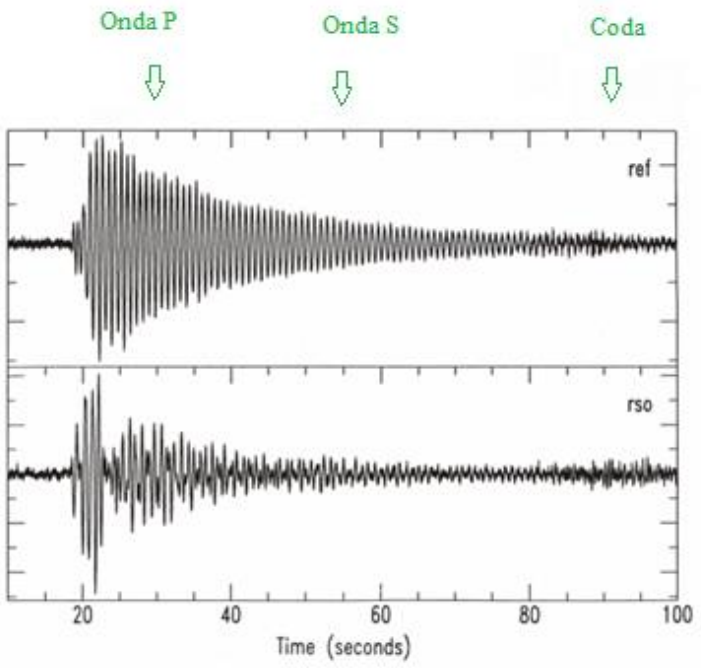
la parte más somera del volcán Cotopaxi, pero en otros volcanes como el Kilahuea va entre los 30 y 60 km (Wassermann, 2012).

Su origen se da cuando el magma sube en forma de cascada hacia la superficie creando una grieta en el volcán, la existencia de la mezcla de fluido y gas generan una presión que causa fenómenos resonantes dentro del magma formando los eventos LP. (Wassermann, 2012)

Los eventos LP ayudan a predecir una erupción volcánica debido a que cuando su número aumenta considerablemente se sabe que el magma está en constante movimiento advirtiendo de esta manera a la población. (Perez D. , 2017)

**Figura 7**

*Ejemplos de eventos LP, del volcán Merapi (Indonesia)*



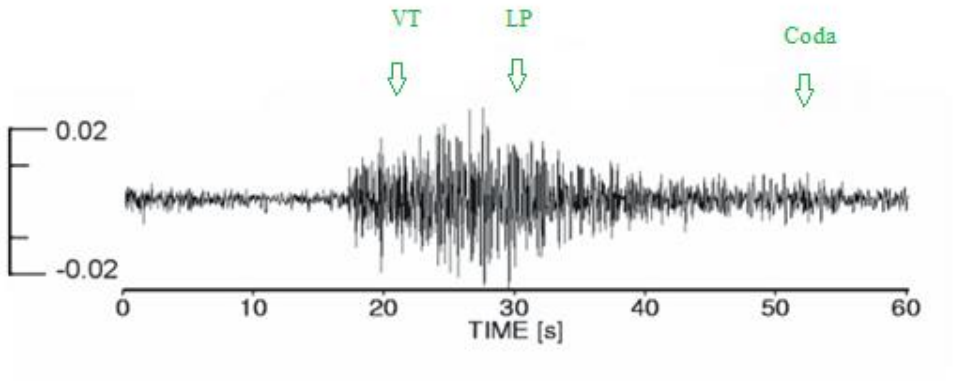

**Fuente:** (Wassermann, 2012)

**c. Eventos Híbridos**

La mezcla de los eventos LP y VT pueden ser los orígenes de los Eventos Híbridos (HYB) ya que estos eventos presentan características de ambos según (Perez D. , 2017), se puede ver un ejemplo en la Figura 8. Su frecuencia de arribo es alta con más de 10 Hz, continuando con frecuencias bajas como las de un evento LP, muestran arribos de *ondas P* y *S* seguidas por la coda de baja frecuencia. Cuando existen HYB se los usa como indicadores de la inestabilidad de las cúpulas de lava con alta viscosidad. (Wassermann, 2012)

**Figura 8**

*Ejemplo de evento HYB, del volcán Merapi (Indonesia)*




**Fuente:** (Wassermann, 2012)

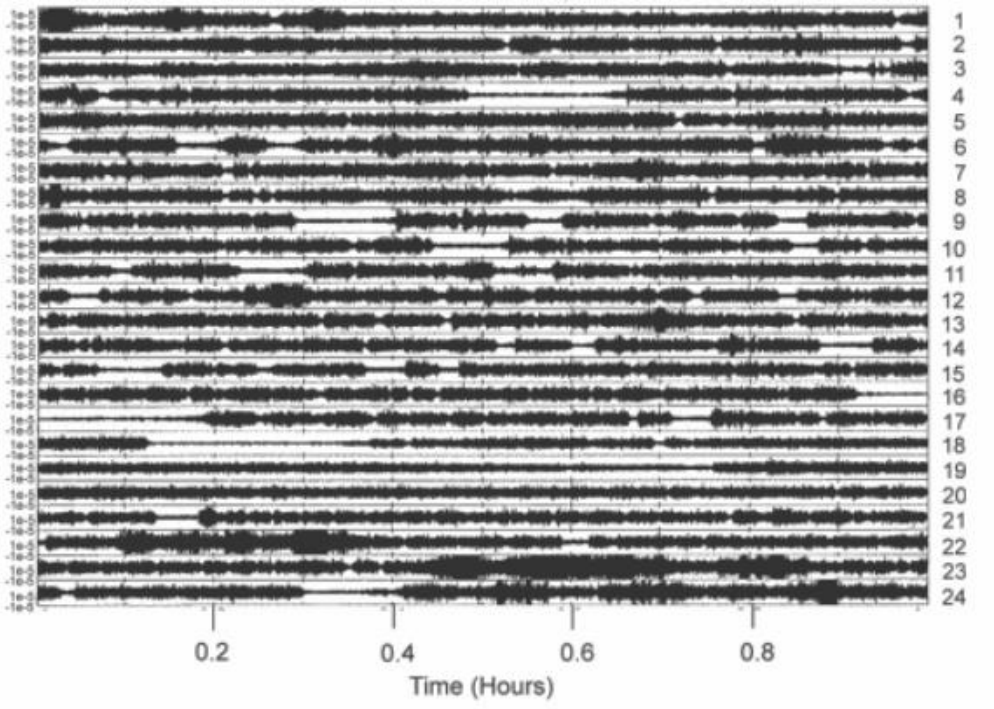
**d. Tremores volcánicos**

Los eventos de Tremores Volcanicos (TRE) tienen una característica fuera de lo común y es la presencia de varios picos agudos con uno dominante como se ve en la Figura 9, tienen frecuencias bajas de 1Hz a 5Hz (Perez D. , 2017). Su señal es continua y puede durar entre minutos, horas, días, o semanas.

Se originan por la desgasificación del conducto del edificio volcánico, cuando expulsa gas, vapor o ceniza las amplitudes de la señal son altas, y cuando expulsan lava su amplitud es baja. de Un resonador similar a los LP puede ser otro origen de los TRE, solo que estos eventos son de acción prolongada por lo que se considera a los LP como fuente elemental de actividad tremórica.

**Figura 9**

*Ejemplos de eventos TRE, del volcán Bromo (Indonesia) durante una fase de actividad alta*




**Fuente:** (Wassermann, 2012)

Es de suma importancia conocer los ESV que se presentan en un volcán para el presente trabajo de investigación, debido a que se busca clasificarlos de manera automática mediante algoritmos de aprendizaje y clasificación con ML. Las características de los eventos son esenciales para su clasificación.

### 2.2.2. Monitorización del volcán Cotopaxi

La monitorización de un volcán activo como el Cotopaxi es de suma importancia para todo el país, por esta razón es el volcán con la primera base de monitorización constante de Sudamérica desde 1970, con el pasar de los años se ha ido mejorando los radares y señores al punto que actualmente cuenta con una red de sismógrafos y 59 estaciones dedicadas a su Monitorización. (Medina, 2015)

Algunos artefactos importantes de la red de monitorización son: cámaras térmicas, GPS, sismógrafos, antenas de banda ancha, tecnología de transmisión con sistemas digitales y análogos,

satélites, entre otros (Guffanti, y otros, 2007). El Ecuador ha recibido ayuda de instituciones extranjeras en la instalación de sensores y en la recopilación de datos, a través de los años los países que más han ayudado han sido Japón, USA, y Francia, con algunos equipos como acelerógrafos, softwares para eventos volcánicos y terremotos, tecnología de infrasonido, GPS, sensores que registren grandes anchos de banda en periodos de más de 1 minuto. (IGEPN, Cotopaxi, 2017)

Dentro de las tecnologías, sensores, y equipos que posee el IGEPN dentro de la red de monitorización, los más importantes para este trabajo de investigación son los que ayuden en la detección, identificación, y clasificación de ESV. En el Cotopaxi se encuentra instalada una red de 23 sismómetros, 6 periodo cortos, 12 de banda ancha, y 5 de banda ancha con infrasonido los cuales transmiten vía radio al IGEPN en tiempo real de forma constante. (IGEPN, Cotopaxi Red Monitoreo, 2018).

El software SIPASS permite identificar los espectros de frecuencia usando la transformada de Fourier en los ESV para poder clasificarlos según la morfología y rangos de frecuencias del evento. Este software igual permite obtener características únicas de las señales, obteniendo un vector de 84 caracteres que describen la señal en tiempo y frecuencia, formando una base de datos etiquetadas con características únicas de cada ESV.

### 2.3. Machine Learning y Python

En esta sección se cubrirán temas sobre el software, los algoritmos, y métodos usados para la clasificación de ESV. Primero se hablará sobre el software elegido Python y sus motivos por el cual es el ideal para el proyecto. Segundo, se trata sobre la Suite de Python la distribución de Anaconda, en el cual se puede desarrollar temas de ML. A continuación, se explicará sobre la rama científica de la Inteligencia Artificial ML, su funcionamiento, y sus clases. Acto seguido, se explicará cuáles son los algoritmos usados para la clasificación de ESV. Una vez se conoce los algoritmos se entrena un modelo de clasificación y se

explica cómo logran la clasificación y predicción de ESV de cada uno de ellos. Finalmente se revisa los parámetros de las medidas de rendimiento para poder ver los resultados porcentuales de la investigación.

### 2.3.1. Software libre Python

Python es un lenguaje de programación orientado a objetos, de alto nivel, con semántica dinámica, y con estructuras de datos integradas. Python admite paquetes y módulos externos, lo que fomenta la reutilización de código y el crear módulos del programa. Las múltiples distribuciones y sus extensas librerías son una fuente que está disponible para todos los usuarios, la forma binaria del programa de igual forma se encuentra apto para la mayoría de plataformas. Por consecuente Python es usado en muchas empresas grandes del mundo como Google, Red Hat, Youtbe, Microsoft, entre otras, con esto se puede ver la utilidad del software y el porque es necesario implementar un clasificador con código abierto. (Python, 2012)

Uno de los principales atractivos para el uso de Python es su Código Abierto o Software libre, se lo denomina así ya que los usuarios tienen la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software o el código (Granada, 2018). Al ser un software de código abierto los usuarios a lo largo del tiempo han ido mejorando sus características y facilitándose entre ellos lo códigos que han permitido que tenga un avance muy rápido y tenga librerías sobre varios tópicos, llegando a que la aplicación en entorno servidor evolucione con Python, esto junto con su sencillez a la hora de programar ha hecho que se situé en el uso de Big Data y en especial al desarrollo de algoritmos de aprendizaje automatizado como Inteligencia Artificial, Deep y Machine Learning y en el Data Science. (Robledano, 2019)

ML va de la mano con Python debido a que desde programadores nuevos hasta expertos entenderán la sintaxis, esta es simple y facilita su escritura como la depuración de su código de forma muy rápida. Las librerías de NumPy, SciPy, Pandas, son ideales para el procesamiento científico y para la

manipulación de datos, aparte Matplotlib para la creación de gráficos que viene junto con scikit-learn para el aprendizaje automático, son las herramientas ideales para la generación de un modelo de clasificación de ML. (Kharkovyna, 2019)

### 2.3.2. Anaconda y Jupyter Notebook

La distribución de Anaconda es considerada de las Suites de código abierto más completas para el Data Science (DS) con Python ya que posee múltiples funcionalidades que aportan al desarrollo de aplicaciones de manera rápida, sencilla y eficiente. Posee varias librerías, aplicaciones, y conceptos para el DS en Python, esta distribución se maneja como un gestor de paquetes, un gestor de entorno, y pose más de 720 paquetes de código abierto. (Anaconda, 2015)

A continuación, se redactarán algunas de las características más relevantes de Anaconda: Multiplataforma funciona en Windows, Linux, y Mac OS, es de código abierto, con documentación bien explicada, instalación rápida y fácil, permite administrar e instalar dependencias, paquetes, y entornos para DS con una línea de comando, tiene varios IDE como Jupyter, Spyder, RStudio, y JupyterLab. Cuenta con herramientas para el análisis de datos como Dask, Numpy, Pandas y Numba, la visualización de datos se la realiza con Matplotlib, Holoviews, Datashader, o Bokeh, la comunidad brinda aplicaciones relacionadas con ML y sus modelos de aprendizaje, la gestión de paquetes de DS con Python se la puede hacer desde el terminal. Brinda el libre acceso a recursos más avanzado, elimina la dependencia de paquetes y control de versiones, permite la compilación de Python en código de máquina, los algoritmos paralelos para la ejecución de códigos son mas fáciles en su escritura. La portabilidad de proyectos permite que sean ejecutados en diferentes plataformas, y lo más importante la implementación de proyectos de DS y ML es muy rápida y sencilla. (Toro, 2018)

Jupyter Notebook (JN) es el IDE que se usara para este trabajo de investigación, por lo tanto, se explicara sus ventajas y utilidades a la hora de generar los algoritmos de ML. Como lo explica su página

web (Jupyter, 2014) JN es una aplicación web donde se crea y comparte documentos con código en vivo, ecuaciones, texto, y gráficos. Unas de sus utilidades son la manipulación y limpieza de datos, modelado estadístico, simulación de procesos numéricos, visualización de datos, aprendizaje automático. La interfaz del usuario está configurada para soportar una gran cantidad de procesos y datos en el ámbito de DS, ML, Deep Learning, y computación científica. Los resultados del código pueden mostrarse de manera interactiva en imágenes, HTML, LaTeX, o videos, y cabe recalcar la característica más importante del IDE para este proyecto son las herramientas para ML, como son, Pandas, scikit-learn, Tensor Flow, ggplot2, Apache Spark para Python. (Guy & Mohamed, 2018)

### 2.3.3. Machine Learning

ML es una rama de la Inteligencia Artificial que genera modelos de aprendizaje automático, la maquina iterativamente aprenden de los datos generando un modelo, esto lo realiza identificando patrones en todos los miles de datos proporcionados. El aprendizaje de la maquina es realmente un algoritmo que revisa todos los datos y es capaz de predecir comportamientos futuros, esto implica que los sistemas pueden ir mejorando con el tiempo, mientras más datos tengan mejor será la calidad de su predicción y clasificación. Acto seguido, se explicará más a detalle el funcionamiento de ML. (Smola & Vishwanathan, 2008)

Existen unos pasos generales a la hora del uso de ML y la generación de su modelo en la Figura 10 se puede ver estos pasos. Este proceso de aprendizaje automático comienza con los datos para el entrenamiento, se debe recolectar datos de lo que se desee predecir o clasificar, para formar una base de datos con la cual se trabajara. El segundo paso será analizar los datos, para esto se debe realizar una limpieza de la base de datos, ya que no deben existir espacios nulos, o en algunos casos existen parámetros que sobresalen del resto indicando anomalías que puede perjudicar al modelo. Luego se localizan los patrones de los datos, que tienen en común dichos datos para poder realizar cierta predicción

esto se lo hace con los algoritmos de ML los cuales son los encargados de analizar uno por uno los datos para encontrar similitudes que permitan que el modelo aprenda de ellas. El cuarto paso es ver el rendimiento del modelo, esto se lo realiza ingresando nuevos datos que no conozca el modelo y analizar las predicciones o clasificaciones que realice, ver el porcentaje de aciertos y fallos. Por último, se realiza un proceso de retroalimentación, de esta forma se puede volver a entrenar al modelo conociendo los fallos para que el proceso siga mejorando.

**Figura 10**

*Proceso de Machine Learning*


**Fuente:** (Ahmed, 2019)

Se ha explicado ML permite a las maquinas mejorar tareas de acuerdo a la experiencia, como sabemos existen innumerables tareas que el humano desea automatizar con ML por lo que existen tres tipos principales categorías de ML, aprendizaje supervisado, aprendizaje no supervisado, y aprendizaje de refuerzo. Aprendizaje supervisado se usa cuando se tiene los datos etiquetados y se intenta predecir o clasificar un evento en base a los conocimientos de los datos. El algoritmo no supervisado se usa con datos no etiquetados y se intenta agrupar los datos similares basados en características que el algoritmo vaya

detectando, o características impuestas. El algoritmo de refuerzo axprende a desarrollar una acción en base a la experiencia, no tiene datos de inicio, sino que va aprendiendo de acuerdo a los sucesos. (Portilla, 2019)

Por simple descarte al saber que los datos de los ESV se encuentran clasificados se puede saber cuál es la categoría que se usara en este trabajo de investigación, siendo el aprendizaje supervisado el cual se usara, por lo que se debe conocer cuáles son los algoritmos que permitan realizar una clasificación correcta de los ESV.

### 2.3.4. Aprendizaje Supervisado

Como se explicó anteriormente el aprendizaje supervisado trabaja con datos etiquetados, esta categoría se subdivide en dos grupos de algoritmos de regresión y clasificación. Los algoritmos de regresión son aquellos usados para predecir un patrón, estos algoritmos son entrenados con los datos para encontrar similitudes en las características de los datos y generar un modelo de predicción. Los algoritmos de clasificación de igual forma encuentran patrones dentro del conjunto de datos de cada una de las etiquetas, de esta forma entrenan al modelo sabiendo las características de las etiquetas, luego cuando ingresan nuevos datos pasan por el modelo y el algoritmo se encarga de encontrar estos patrones para realizar una comparación con las características anteriores y así realizar la clasificación de los nuevos datos.

Existen dos tipos de algoritmos clasificadores, los de aprendizaje perezoso y ansioso. El aprendizaje perezoso, almacena los datos de entrenamiento hasta que los datos de testeo aparezcan. Cuando aparecen, la clasificación se realiza en función de los datos más relacionados con los datos almacenado, unos ejemplos de estos clasificadores son *Case-based Reasoning* o KNN. Los clasificadores ansiosos construyen un modelo basado en los datos de entrenamiento antes de recibir nuevos datos (Asiri, 2018). El modelo debe generar una hipótesis que cubra todos los posibles nuevos datos, unos

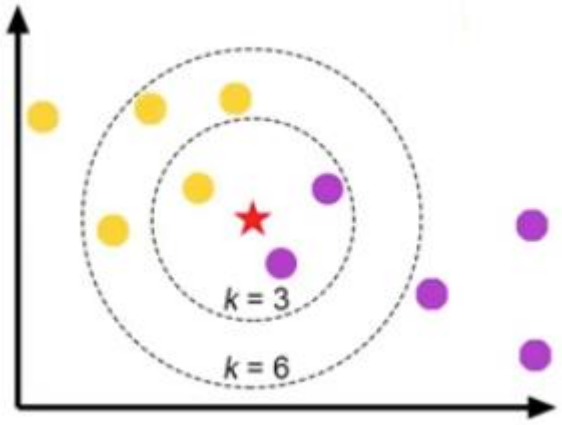
ejemplos de este clasificador son DT, NB, ANN, SVM, GBDT, *Random Forest* (RF), entre otros. A continuación, se explicarán los algoritmos usados en este trabajo de investigación.

#### 2.3.4.1. K Nearest Neighbors (KNN)

Como (Portilla, 2019) lo explica KNN trabaja encontrando los puntos de datos más similares en los datos de entrenamiento y realiza una suposición basada en sus clasificaciones. KNN al ser un método de aprendizaje perezoso no genera un modelo con los datos de entrenamiento, sino que el aprendizaje sucede al mismo tiempo que se van probando los datos de prueba. Para explicar el funcionamiento del algoritmo debemos observar la Figura 11 donde se muestra los datos de entrenamiento en dos clases, un punto a clasificar representado por una estrella roja, y lo más importante *k* que es un valor numérico para determinar el número de datos con los que se compara el dato a clasificar.

**Figura 11**

*Representación de datos para clasificación de k para KNN*




El proceso consta de los siguientes pasos, primero se debe asignar un valor numérico a *k* que serán las instancias del algoritmo, segundo se calculara la distancia Euclidiana como se ve en la fórmula (1), entre el dato a ser clasificado y los *k* puntos más cercanos al dato a clasificar de los datos de entrenamiento. A continuación, el algoritmo selecciona los *k* puntos más cercanos, en otras palabras, los que tengan la distancia Euclidiana más corta. Una vez obtenidos los puntos más cercanos se realiza una

votación entre ellos para ver a que clase pertenece el dato a clasificar, la clase con mayor número de votos entre los puntos cercanos es la dominante y a la cual se clasifica el nuevo dato. Este proceso se lo repite para todos los nuevos datos que serán clasificados. (Ahmed, 2019)

$$Distancia\ Euclidiana = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (1)$$

De forma matemática, se identifica los *k* puntos más cercanos a $x_0$ de los datos de entrenamiento, representado por $N_0$. Luego se estima la probabilidad condicional por la clase *j* como una fracción de puntos de $N_0$ que corresponde a valores iguales de *j,* como lo representa ( 2). (James, Witten, Hastie, & Tibshirani, 2015)

$$\Pr(Y = j \mid X = x_o) = \frac{1}{K} \sum_{i \in N_o} I(y_i = j) \quad (2)$$

Finalmente, se aplica el clasificador de Bayes ( 3), el cual asigna cada observación a la clase más probable, dada sus valores predictores En otras palabras, simplemente se debe asignar una observación de prueba con el vector predictor $x_0$ a la clase *j* para la cual es la más grande. (James, Witten, Hastie, & Tibshirani, 2015)

$$\Pr(Y = j \mid X = x_o) \quad (3)$$

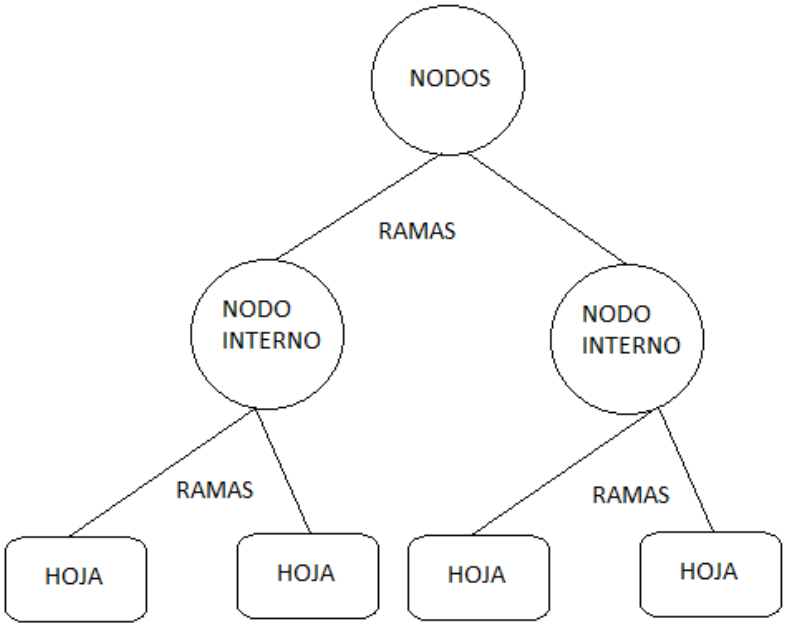
#### 2.3.4.2. Arboles de Decisión

DT es técnica de aprendizaje automático supervisadas en las que los datos se dividen de acuerdo con condiciones específicas. En (James, Witten, Hastie, & Tibshirani, 2015) explica que el algoritmo lo primero que hace para generar el árbol de decisión es dividir por regiones a los datos, esto lo realiza por características que vaya encontrando en la similitud de los datos de entrenamiento, a estos grupos se les denomina *nodos*. Los Arboles de Decisión se van formando en el sentido de que las hojas están en la parte inferior del árbol. Los puntos a lo largo del árbol donde se divide el espacio predictor se denominan *nodos*

*internos* y los segmentos de los árboles que conectan los nodos son las denominadas *ramas,* la construcción del árbol culmina con las llamadas *hojas* que son la clasificación final, se puede observar un esquema de DT en la Figura 12.

**Figura 12**

*Estructura de un Árbol de Decisión*




El objetivo de este método es generar un modelo que pueda clasificar los *c* valores posibles de una variable objetivo mediante el aprendizaje de reglas de decisión simples inferidas de las características de los datos. Cada nodo en el árbol especifica una regla para algún atributo de la instancia, y cada rama que desciende de ese nodo corresponde a uno de los valores posibles para ese atributo. Una instancia se clasifica comenzando en el nodo raíz del árbol, probando el atributo especificado por este nodo y luego bajando la rama correspondiente al valor del atributo. Este proceso se repite para el subárbol que nace de la raíz en el nuevo nodo. El algoritmo promueve árboles pequeños en lugar de árboles grandes, lo que produce clasificadores con buenas capacidades de generalización. La profundidad o la ligereza del árbol es el

parámetro libre para esta técnica de aprendizaje automático, que se mide en términos de la información realmente contenida por los nodos secundarios.

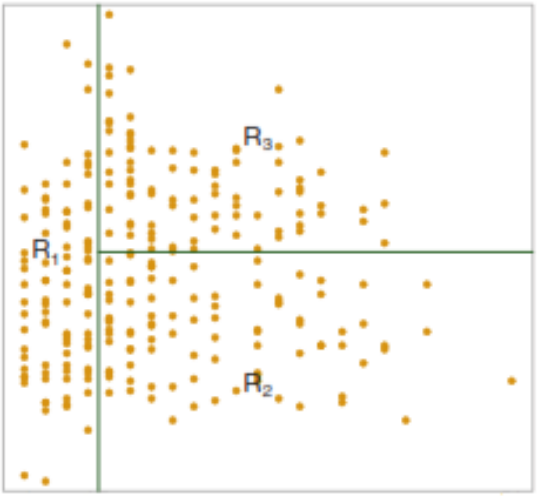
Ahora el proceso de construir un árbol de decisión en términos generales son dos pasos, y un ejemplo del resultado de estos pasos se ve en la Figura 13.

1. Dividimos el espacio predictor, es decir, el conjunto de valores posibles para $X_1$, $X_2, \ldots, X_n$, que es el conjunto de datos, en *J* regiones distintas y no superpuestas, por $R_1$, $R_2, \ldots, R_J$.

2. Por cada observación que cae en la región $R_J$ se realiza la misma clasificación que es la media de los valores de respuesta para las observaciones de entrenamiento en $R_J$

**Figura 13**

*Representación de datos divididos en regiones para DT*




Por ejemplo, dadas las regiones $R_1$ $y$ $R_2$ y la respuesta media de los datos de entrenamientos para $R_1$ es 25 y $R_2$ es 50. Entonces para un nuevo dato donde $X_1 = c$, si $c \epsilon R_1$ su clasificación será de 25, y lo mismo si $X_2 = c$, si $c \epsilon R_2$ su clasificación será 50.

Las regiones o cajas se construyen por el resultado del modelo de clasificación, con el objetivo de reducir la tasa de error de clasificación. La tasa de error de clasificación *E* es simplemente la fracción de

los datos de entrenamiento en esa región que no pertenecen a la clase más común, esto es dado por la fórmula (4)

$$E = 1 - \max_{k}(p_{Rk}) \quad (4)$$

Donde se obtiene el resultado de restar uno menos el máximo de $p_{Rk}$ y $p_{Rk}$ representa la porción de datos de entrenamiento en la *R* región de la clase *k*. Sin embargo, el error de clasificación no es lo suficientemente sensible para el crecimiento de árboles, por lo que son preferibles otras medidas como el índice de Gini. Aquí, utilizamos dos índices para este propósito, es decir, con la cantidad promedio de información contenida en cada evento

El índice de Gini se define por (5) es una medida de la varianza total entre las clases *k*. El índice de Gini adquiere su valor si todas las $p_{Rk}$ están cerca de cero o uno. Por esta razón, el índice de Gini se conoce como una medida de la pureza del nodo; un valor cercano a uno indica que un nodo contiene predominantemente observaciones de una sola clase.

$$G = \sum_{k=1}^{Rk} p_{Rk}(1 - p_{Rk}) \quad (5)$$

De esta forma el árbol va creciendo y de acuerdo a como funcione el modelo en la clasificación se podría podar el árbol o no, este es un proceso que se realiza si el modelo se encuentra en un estado de sobre entrenamiento.

#### 2.3.4.3. Naive Bayes

Este algoritmo se basa en el teorema de Bayes sobre la independencia entre los predictores. El clasificador NB supone que la presencia de una característica particular en una clase no está relacionada con la presencia de otra característica. (Asiri, 2018). El clasificador NB es un modelo probabilístico de ML, su algoritmo se basa en (6), este algoritmo es muy útil para conjuntos de datos muy grandes. Además, se sabe que NB incluso se compara con métodos de clasificación muy sofisticados como el Gradient Boost.

(Ray, 2017). En ML lo que deseamos es clasificar los datos *B* seleccionado la clase *A*, lo que se desea es hallar la clase más probable dado un conjunto de datos que usan como conocimiento previo sobre las clases. El teorema de Bayes muestra la forma de calcular la probabilidad de la clase con los datos previos, matemáticamente NB usa la formula (6).

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (6)$$

Donde $P(A|B)$ es la probabilidad de las clases *A* dada los datos *B*, se lo conoce como probabilidad posterior, $P(A)$ es la probabilidad que la clase *A* sea cierta, se lo denomina probabilidad previa, $P(B|A)$ es la probabilidad de los datos *B* dada la clase *A* y $P(B)$ es la probabilidad de los datos sean ciertos. Esta es la fórmula reiterativa del algoritmo. (Ray, 2017)

El teorema de Bayes proporcióna una forma de clasificar los datos a partir de la probabilidad posterior $P(A|B)$ a partir de $P(A)$, $P(B)$ y $P(A|B)$. La clasificación se realiza derivando el máximo posterior, que es el $P\ max(A_i|B)$ con la suposición anterior aplicada al teorema de Bayes. Esta suposición reduce en gran medida el costo computacional al contar solo la distribución de la clase. A pesar de que la suposición no es válida en la mayoría de los casos, ya que los atributos son dependientes, NB ha podido desempeñarse de manera impresionante como clasificador. (Asiri, 2018)

#### 2.3.4.4. Support Vector Machine

Para entender sobre SVM se debe comprender que es un hiperplano, el margen, y los vectores de soporte, ya que son parte esencial en el clasificador. Supongamos un espacio $p - dimensional$, un hiperplano es un subespacio plano de una dimensión $p - 1$. (James, Witten, Hastie, & Tibshirani, 2015) Por ejemplo, en dos dimensiones, un hiperplano es un subespacio plano unidimensional, es decir, una

línea. En tres dimensiones, un hiperplano es un subespacio plano bidimensional, en otras palabras, un plano.

La definición matemática para un hiperplano de $p - dimensiones$ se observa en (7) donde cualquier $Y = \left(Y_1, \ldots\ , Y_p\right)^T$ es un punto en el hiperplano.

$$\beta_0 + \beta_1 Y_1 + \beta_2 Y_2 + \ldots + \beta_p Y_p = 0 \quad (7)$$

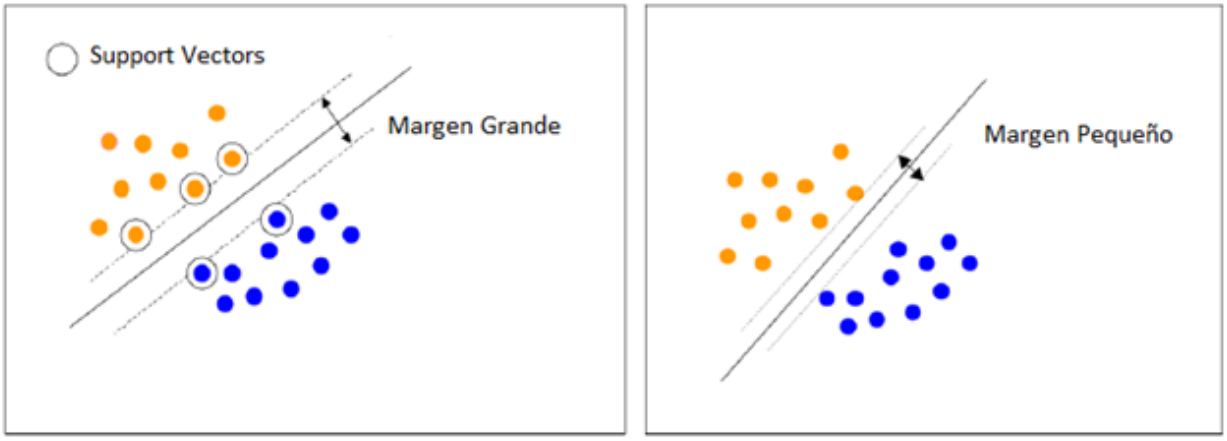
Suponiendo que *Y* no satisfaga (7) más bien si es un número mayor o menor que 0 esto nos indicaría en qué lado del hiperplano se encuentra ubicado un punto. (James, Witten, Hastie, & Tibshirani, 2015)

Un ejemplo para visualizar al hiperlano se ve en la Figura 14, de igual forma se una este ejemplo para la definición de margen y los vectores de soporte. Los vectores de soporte son los puntos de datos más cercanos al hiperplano e influyen en la posición y orientación del mismo, con estos puntos de datos se define el hiperplano, aparte usando los vectores de soporte, maximizamos el margen del clasificador, eliminarlos cambiará la posición del hiperplano. El margen es la brecha entre dos líneas en los puntos más cercanos de los datos, pero de diferentes clases. Se puede calcular como la distancia perpendicular desde la línea a los vectores de soporte. El margen grande se considera un buen margen y el margen pequeño se considera un margen malo. (Tutorialspoint, 2016) Esto se calcula dependiendo cuantas características vayan a aportar para la clasificación, como en la mayoría de los casos supera los 4 planos es imposible visualizarlas debido a esto se usa las denominadas funciones de Kernel explicadas más adelante.

SVM es un algoritmo de entrenamiento que genera un modelo que asigna los nuevos datos en una región del hiperplano, haciendo que este modelo sea un clasificador binario no probabilístico. (Portilla, 2019) El modelo de SVM es una representación de los datos de entrenamiento mapeados en el espacio, estos son divididos por el hiperplano.

**Figura 14**

*Hiperplano y margen de SVM*




Fuente: (Caragea, Cook, & Honavar, 2005)

De esta forma nuevos datos al ser mapeados en el mismo espacio son clasificados de acuerdo a lado del hierplano que se encuentren localizados. Ahora se puede ver el objetivo principal de SVM, el cual es dividir los conjuntos de datos en clases para encontrar un hiperplano con un margen grande, esto para que los datos nuevos puedan ser clasificados sin error. (Tutorialspoint, 2016) Al tener un margen pequeño los datos pueden pasar de un lado del hiperplano al otro, de esta forma se dan los fallos en la clasificación. Al tener una distancia grande en el margen se reduce la posibilidad de errores, ya que los datos se encuentran más separados del hiperplano y reduce los fallos en la clasificación.

Para poder generar un hiperplano con cualquier conjunto de datos que no sean separables se usa Kernel. El método kernel toma un espacio de entrada dimensional bajo y lo transforma en un espacio dimensional superior. En otras palabras, Kernel convierte los problemas no separables en problemas separables al agregarle más dimensiones. Hace que SVM sea más potente, flexible y preciso. Existen algunos tipos de Kernel usados por SVM, se analizará los dos casos más usados.

Kernel Polinomial, es una forma más generalizada del kernel lineal y distingue el espacio de entrada curvo o no lineal. La fórmula para el kernel polinomial se observa en (8), donde *d* es el grado de polinomio, que debemos especificar manualmente en el algoritmo de aprendizaje.

$$K(X, X_i) = 1 + \sum (X * X_i)^d \quad (8)$$

Kernel Radial Basis Function (RBF), utilizado principalmente en la clasificación SVM, asigna espacio de entrada en espacio dimensional indefinido. La fórmula (9) lo explica matemáticamente, el rango gamma es de 0 a 1. Necesitamos especificarlo manualmente en el algoritmo de aprendizaje. Un buen valor predeterminado de gamma es 0.1. (Tutorialspoint, 2016)

$$K(X, X_i) = \exp(-\gamma * \sum (X * X_i)^2) \quad (9)$$

Una vez que se tiene el hiperplano correcto para los datos de entrenamiento SVM como ya se dijo busca maximizar el margen entre los vectores de soporte y el hiperplano, la función que permite esta maximización es Hinge Loss (10).

$$l(y) = \begin{cases} \frac{1}{2\gamma} \max(0, 1 - ty)\text{^}2 & ty \geq 1 - \gamma \\ 1 - \frac{\gamma}{2} - ty & otro\ caso \end{cases} \quad (10)$$

El resultado es 0 si el valor predicho y el valor real son del mismo signo. Si no lo son, calculamos el valor de la pérdida. También agregamos un parámetro de regularización a la función de costo. El objetivo del parámetro de regularización es equilibrar la maximización del margen y la pérdida. Resolviendo la ecuación (10) al derivarla parcialmente e igualar los pesos, se obtiene dos escenarios del gradiente, cuando se clasifica correctamente (11) y cuando falla (12). (Gandhi, 2018), este parámetro es conocido como *C*.

$$\omega = \omega - \alpha(2\gamma\omega) \quad (11)$$

$$\omega = \omega + \alpha(y_i x_i - 2\gamma\omega) \quad (12)$$

Por lo que controlar este parámetro en el clasificador es fundamental, ya que controla hasta dónde llega la influencia de un solo conjunto de entrenamiento. Un valor grande de *C* hace que los puntos de datos más cercanos tengan un peso elevado. Y un valor pequeño de *C*, será una solución más generalizada. (Ahmed, 2019)

### 2.3.5. Métricas en Clasificación

Típicamente en la mayoría de modelos de clasificación se puede lograr dos objetivos, que el modelo este correcto en la clasificación o este incorrecto en la clasificación. Este sistema de evaluación se expande a múltiples clases afortunadamente, pero para explicar de forma más sencilla los parámetros de medición se explica la clasificación binaria.

Como ya se ha comentado e igual como explica (Portilla, 2019), al tratarse de un aprendizaje supervisado, primero se debe entrenar al modelo con los datos de entrenamiento, luego se realiza el test con los datos de prueba. Una vez obtenidos los datos de test usando el modelo de clasificación, se compara las etiquetas correctas con los resultados de la clasificación. Se repite este proceso para todos los datos de prueba que ingresan al modelo, al finalizar se obtiene un conteo de todas las clasificación correctas e incorrectas, lo importante es tener en cuenta que solo estos dos parámetros no explicaran todo lo que el trabajo de investigación lleva a cabo, por este motivo se usan cuatro métricas a partir de las dos anteriores. Organizando los resultados de los datos de prueba con los datos clasificados se genera la denominada matriz de confusión con las cuatro métricas que se mostraran más adelante, explicado por (Ghoneim, 2019).

Las clasificaciones correctas e incorrectas se las detalla mejor en la matriz de confusión ver ***Tabla 1***, donde se ubican los verdaderos positivos (TP), verdaderos negativos (TN), los falsos negativos (FN) y los falsos positivos (FP). Los TP son el número total de positivos que fueron correctamente clasificadas como positivos, TN es el total de negativos que han sido correctamente clasificadas como negativos, FN es número total de positivos que fueron clasificadas de forma incorrecta como negativos por el modelo y FP es número total de negativos que fueron clasificadas de forma incorrecta como positivos por el modelo (Zelada, 2017). Con estos datos se obtienen las métricas de clasificación para evaluar al modelo.

**Tabla 1**

*Matriz de Confusión*

| | | Condiciones Predichas | |
|---|---|---|---|
| | Población Total | Predicciones Positivas | Predicciones Negativas |
| Condiciones Verdaderas | Condición Positiva | Verdaderos Positivos (TP) | Falsos Negativos (FN) |
| | Condición Negativa | Falsos Positivos (FP) | Verdaderos Negativos (TN) |

Exactitud o *A* (del inglés *Accuracy*) es la métrica más intuitiva que se obtiene es el número de predicciones realizadas correctamente hechas por el modelo dividido para el número total de predicciones. Su fórmula es (13), en el numerador se encuentran todos los datos con etiquetas correctas y en el denominador todos los datos. *A* es útil cuando las clases a clasificar están balanceadas.

$$A = \frac{TP + TN}{TP + FP + FN + TN} \quad (13)$$

Sensibilidad o R (del inglés *recall*) es la cualidad del modelo de encontrar todos los casos más relevantes de la base de datos, o su definición más teórica es el TP dividido para la suma de TN más FN, su fórmula se puede ver en (14). Para comprender mejor el concepto de RE se formula una pregunta de ejemplo: De todos los LP clasificados, ¿Cuantos están correctamente clasificados?

$$R = \frac{TP}{TP + FN} \quad (14)$$

Precisión o *P* es la habilidad del modelo de clasificación de identificar solo los datos relevantes. Numéricamente se define como los TP divididos para la suma de TP mas los falsos negativos (FP), su fórmula se ve en (15). Para comprender mejor el concepto de P se formula una pregunta de ejemplo: ¿Cuántos de los LP clasificados en realidad son LP?

$$P = \frac{TP}{TP + FP} \quad (15)$$

Especificidad o *S* (del inglés *sensibility*) es la clasificación correcta de todos los TN, matemáticamente se define como TN dividido para TN mas FP, su fórmula es (16). Para comprender mejor el concepto de S se formula una pregunta de ejemplo: De todos los ESV menos LP ¿Cuantos ESV están correctamente predichos?

$$S = \frac{TN}{TN + FP} \quad (16)$$

Como se mencionó anteriormente si se organizan estas métricas se forma la matriz de confusión que se puede observar en la ***Tabla 1***, esta se puede extender hasta N clases y llegar a ser una matriz NxN, donde las columnas son las condiciones o clases predichas por el modelo y las filas son las condiciones reales. Esto sirve para mostrar cuando las clases están confundidas entre otras, dando la facilidad de trabajar con distintos errores de forma separada. (Santos, 2018)

En este trabajo de investigación lo más importante es detectar un aumento inusual de los eventos LP por lo que el enfoque de la clasificación radica en este evento. Para tener una idea de cuál parámetro es el más importante en un clasificador cabe recordar la de los parámetros de desempeño, vistas anteriormente, de donde se deduce que los parámetros más importantes son *R* y *P*. Al querer analizar dos parámetros lo primero que se nos podría ocurrir seria sacar un promedio de ambas y así analizar el

rendimiento de los clasificadores, pero ML ya genero su propio parámetro de rendimiento a partir de estudiar la relación de *P* y *R*, este es conocido como *F1-score*.

El parámetro *F1-score* o también denominado como el balance F-medida del inglés (*F-measure*) es un promedio ponderado entre *P* y *R* donde la contribución relativa de cada uno de ellos es igual. (ScikitLearn, sklearn metrics f1_score, 2016) Su fórmula matemática se ve en (17) y para calcularlo en Python hacemos es aumentar esta fórmula en el módulo de *desempeno* con la cual ya hemos calculado previamente los parámetros *A, S, P* y *R*. Este nuevo parámetro se mide entre 1 y 0 donde 1 es el mejor valor que se podrá alcanzar.

$$f1score = \frac{2\text{x}P\text{x}R}{P + R} \quad (17)$$

La tasa de error balanceado o BER (del inglés *Balanced Error Rate*) es otro parámetro que mide la relación entre R y P es el cual igual será analizado. BER es el promedio de la proporción de clasificaciones incorrectas en cada clase, su fórmula se ve en (18).

$$BER = 1 - \frac{(S + R)}{2} \quad (18)$$

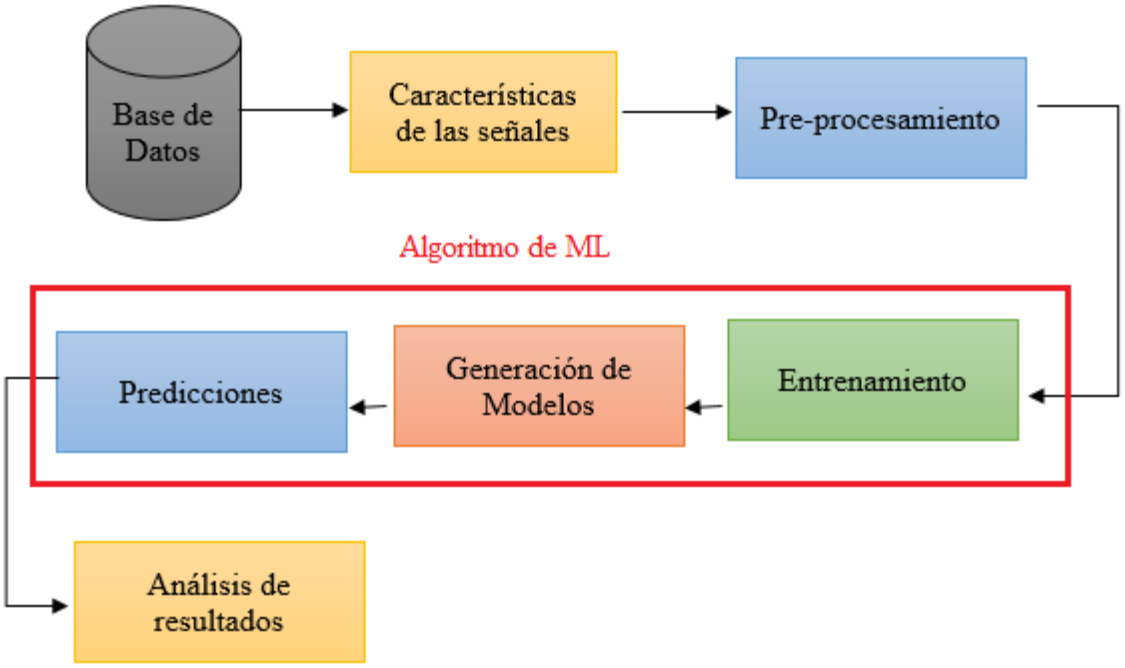
# CAPÍTULO III

## 3. Metodología

En este proyecto se usará la metodología de investigación experimental. Una vez realizado el estudio del estado del arte sobre ML en el que se pudo definir cuáles son los algoritmos de ML más usados y sus características, en este capítulo describiremos como se desarrolló el trabajo de investigación, el cual se lo explicará por etapas como se ve en la Figura 15.

**Figura 15**

*Diagrama de bloques del trabajo de investigación.*




El primer y segundo bloque describen cómo se obtuvieron los datos de los ESV del volcán Cotopaxi con los cuales se genera una base de datos con ESV, luego gracias a trabajos previos se obtienen 84 características de cada ESV, las cuales forman la base de datos inicial, que se usarán en el presente trabajo de investigación. El tercer bloque es el denominado preprocesamiento, consta de tres sube etapas que se

aplican a la base de datos inicial, primero se analiza la base de datos inicial para determinar una nueva base de datos balanceada, seguido se aplica una técnica denominada normalización y por último, se aplica un método de selección de características. Las etapas de preprocesamiento son de vital importancia debido a que luego de aplicar las tres subetapas se produce la nueva base de datos modificada consiguiendo que esta base sea balanceada, normalizada y con menos características. En los dos siguientes bloques de entrenamiento y generación de modelos se explicará el código de Python de cómo se entrenan y generan los modelos de los clasificadores de ML. Continuando con la etapa de clasificación, aquí se usa los modelos generados para clasificar los datos de prueba y obtener el rendimiento de dichos modelos. Finalmente, en la etapa de análisis de resultados se comparan los parámetros de *A, P, R, S*, *f1-score, y BER* de los clasificadores de esta forma se podrá determinar que algoritmo es más eficiente de ML en la clasificación de ESV usando Python. Cada una de las etapas descritas se explicarán a detalle en el resto de este capítulo.

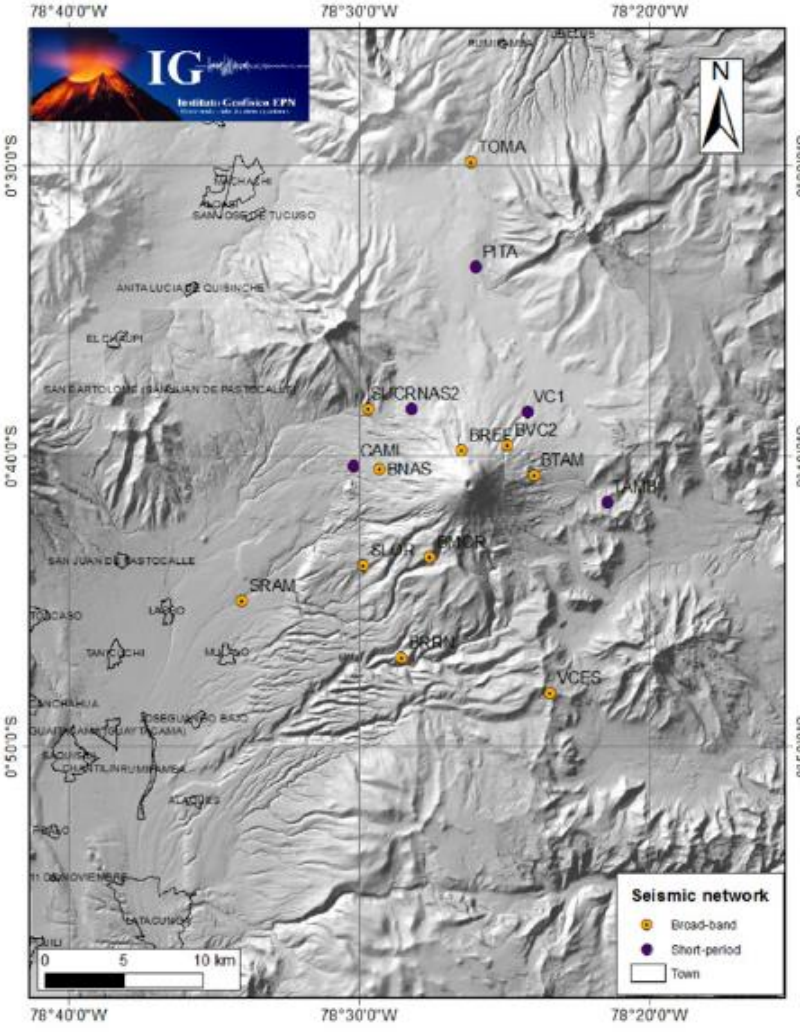
### 3.1. Bases de Datos

Se denomina a la primera base de datos como base de datos inicial la cual es usada en este trabajo de investigación y consta de características de ESV, las cuales se obtienen gracias a trabajos previos realizados sobre una base de datos de ESV proporciónada por el IGEPN. Los ESV fueron recolectados por el sistema de Monitorización y la red de sensores de alta calidad colocados en el Cotopaxi desde el año 2009 hasta el 2010. Estos datos son transmitidos al IGEPN usando señales de radio en la banda UHF. La red contiene dos tipos de estaciones sísmicas, las cuales se muestran en Figura 16, posee cinco estaciones de corto periodo (SP) cuatro de ellas con sensores de componentes verticales y dos de con sensores de tres componentes y todos con un rango de frecuencia de respuesta de 1–50 Hz y once estaciones de banda ancha (BB) (BREF, BNAS, SRAM, BRRN, BTAM, BMOR, SLOR,TOMA, SUCR, BVC2, y VCES), con un rango de frecuencia de respuesta de 0.1–50 Hz. (Lara, Benítez, Carrera, Ruiz, & Rojo, 2016). Cada evento detectado

por los sensores es digitalizado con una frecuencia de muestreo de 100 Hz por un convertidor análogo digital de 12 bits. (Benitez, Paillacho, Rojo, & Lara, 2017). La base de datos cuenta con 1187 ESV cada uno con 84 características, el total de ESV se reparte en 1044 eventos LP, 101 eventos VT, 8 eventos HB, 27 eventos regionales (RE) y 7 eventos de rompimiento de glaciales o también conocidos como Icequakes (ICE).

**Figura 16**

*Localización y despliegue de las estaciones sismológicas del Cotopaxi.*




Fuente: *(IGEPN, Cotopaxi, 2017)*

### 3.2. Características de las señales

Las características de las señales son un conjunto de métricas obtenidas de cada uno de los ESV, estas características representan a cada evento por separado y permiten clasificarlos.

**Tabla 2**

*Sumario de las 84 características*

| Dominio del Tiempo | | Dominio de Frecuencia | | Dominio de Escala | |
|---|---|---|---|---|---|
| ID | Características | ID | Características | ID | Características |
| f1 | Media | f29 | Potencia | f57 | Porcentaje de energía para D1 |
| f2 | Desviación Estandar | f30 | Densidad de picos en RMS[d] | f58 | Porcentaje de energía para D2 |
| f3 | Varianza | f31 | 2do pico con valor más alto | f59 | Porcentaje de energía para D3 |
| f4 | Entropia | f32 | Frec. de 2nd pico más alto | f60 | Porcentaje de energía para D4 |
| f5 | Kurasis | f33 | 3er pico con valor más alto | f61 | Porcentaje de energía para D5 |
| f6 | Entropia Multiescala (MSE)[b] | f34 | Frec. de 3rd pico más alto | f62 | Porcentaje de energía para D6 |
| f7 | Tiempo hasta máximo pico | **Dominio de Escala** | | f63 | A6 RMS en tiempo-dominio |
| f8 | Valor RMS | f35 | A6 Max. pico en frec.-dominio | f64 | D1 RMS en tiempo-dominio |
| f9 | Valor Pico-a-pico | f36 | D1 Max. pico en frec.-dominio | f65 | D2 RMS en tiempo-dominio |
| f10 | Pico-a-RMS porción[c] | f37 | D2 Max. pico en frec.-dominio | f66 | D3 RMS en tiempo-dominio |
| f11 | Energía | f38 | D3 Max. pico en frec.-dominio | f67 | D4 RMS en tiempo-dominio |
| f12 | Ratio del cero | f39 | D4 Max. pico en frec.-dominio | f68 | D5 RMS en tiempo-dominio |
| f13 | Density de picos above RMS[d] | f40 | D5 Max. pico en frec.-dominio | f69 | D6 RMS en tiempo-dominio |
| | **Dominio de Frecuencia** | f41 | D6 Max. pico en frec.-dominio | f70 | A6 Pico-a-pico en tiempo-dominio |
| f14 | Frecuencia de maximo pico | f42 | A6 Frec. de max. pico | f71 | D1 Pico-a-pico en tiempo-dominio |
| f15 | Bandawidth de 90% energía[e] | f43 | D2 Frec. de max. pico | f72 | D2 Pico-a-pico en tiempo-dominio |
| f16 | Entropia[a] | f44 | D3 Frec. de max. pico | f73 | D3 Pico-a-pico en tiempo-dominio |
| f17 | Media | f45 | D4 Frec. de max. pico | f74 | D4 Pico-a-pico en tiempo-dominio |
| f18 | Desviacion Estandar | f46 | D5 Frec. de max. pico | f75 | D5 Pico-a-pico en tiempo-dominio |
| f19 | Variance | f47 | D6 Frec. de max. pico | f76 | D6 Pico-a-pico en tiempo-dominio |
| f20 | Energía | f48 | A6 Media en frec.-dominio | f77 | A6 Pico-a-RMS porción en tiempo-dominio[c] |
| f21 | Kurtosis | f49 | D1 Media en frec.-dominio | f78 | D1 Pico-a-RMS porción en tiempo-dominio |
| f22 | Entropia multiescala | f50 | D2 Media en frec.-dominio | f79 | D2 Pico-a-RMS porción en tiempo-dominio |
| f23 | Maximo pico en 10–20 Hz banda | f51 | D3 Media en frec.-dominio | f80 | D3 Pico-a-RMS porción en tiempo-dominio |
| f24 | Frec. de max. pico en 10–20 Hz Banda | f52 | D4 Media en frec.-dominio | f81 | D4 Pico-a-RMS porción en tiempo-dominio |
| f25 | Maximo pico en 20–30 Hz banda | f53 | D5 Media en frec.-dominio | f82 | D5 Pico-a-RMS porción en tiempo-dominio |
| f26 | Frec. de max. Pico en 20–30 Hz Banda | f54 | D6 Media en frec.-dominio | f83 | D6 Pico-a-RMS porción en tiempo-dominio |
| f | Valor RMS | f55 | Media energía de componentes[f] | f84 | Media energía de wavelet coeficientes |
| f | Pico-a-RMS porción[c] | f56 | Porcentaje de energía para A6[g] | | |

[a] Entropía normalizada de Shannon se calcula con $E(x) = -\sum x^2 \log(x^2)$.

[b] MSE mide el grado de complejidad de las señales en el tiempo con una muestra de la entropía.

[c] Relación entre el valor absoluto y el valor cuadrático medio.

[d] Relación entre el número de picos cuyas amplitudes son más altas que el valor RMS con respecto a la longitud de la señal.

[e] Frecuencia mínima donde la energía acumulada es el 90% o más de la energía.

[f] Energía media de los componentes de señal de wavelet de A6, D1, D2, D3, D4, D5 y D6.

[g] Fracción de energía del componente de señal de wavelet con respecto a la energía total. (Perez, y otros, 2020)

Las 84 características de cada ESV ya fueron obtenidas por investigaciones y trabajos previos aplicados a la base de datos inicial de ESV proporciónada por el IGEPN. El software SIPASS es usado para obtener algunas de estas características, su arquitectura comienza con el uso de un código de Análisis Sísmico (SAC) para crear y administrar archivos de salida, la cual permite leer los archivos sísmicos que tienen el formato SEED (Del inglés *The Standard for the Exchange of Earthquake Data*). Seguido por el uso de la FFT (Del inglés *Fast Fourier Transform*), algoritmo que usa la transformada rápida de Fourier para calcular el periodo y obtener el espectro de la señal sísmica. Gracias a este software se pueden extraer varias características de las señales. (Viracucha & De la Bastida, 2014) Al convertir los archivos SEED en formatos .mat de Matlab se usan varias herramientas de este software para ver el comportamiento de cada uno de los eventos, de esta forma se obtienen 84 características de cada ESV. Algunas de estas características son desviación estándar, varianza, entropía, kurtosis, entropía en múltiples escalas, tiempo al que se genera el pico más alto, el valor eficaz, la media, entre otras, todas ellas determinadas en tiempo, frecuencia y escaladas para cada uno de los 1187 ESV de la base de datos. Las 84 características se las observa en la *Tabla 2*.

### 3.3. Pre procesamiento

Se denomina a esta etapa preprocesamiento por el análisis y los procesos que se deben realizar en la base de datos inicial para que sea adecuada y así entrenar a los modelos de ML y poder obtener un mayor desempeño en los parámetros de rendimiento. Las tres subetapas del preprocesamiento son el

balanceo de la base de datos, la normalización, y la selección de características, las cuales se aplican para generar la base de datos modificada. El balanceo de datos permite reducir el número de ESV con los que se entrena a los modelos de clasificación. La normalización permite limitar el rango numérico con el que se van a trabajar los datos, siendo el rango de operación de 0 a 1. La selección de características, como su nombre lo indica reduce el número de características seleccionando las más esenciales para el entrenamiento y generación de los modelos de clasificación. Para la implementación de todas las etapas de preprocesamiento, se usa JN y Python, su código se muestra en la sección de Anexos. A continuación, se explicará cada subetapa de forma teórica y su codificación.

### 3.3.1. Balancear la base de datos

Al analizar la base de datos inicial y contabilizar los ESV se determinó que se cuenta con 1044 eventos LP, 101 eventos VT, 8 eventos HB, 27 eventos RE y 7 ICE, realza de manera significativa que el 88% de la base de datos son solo eventos LP. Como se explica en (Santacruz, 2016) el desbalance en los datos se da cuando el número de elementos no es el mismo para todas las clases en una base de datos en temas de clasificación.

Los algoritmos de ML para la clasificación tienen problemas con bases de datos desbalanceadas ya que son muy sensibles al tamaño de las clases. Consecuentemente, los algoritmos favorecerán siempre a la clase mayoritaria, por lo que las métricas de *A* y *P* sean sesgadas, esto es un gran problema al momento de clasificar el resto de clases. (Santacruz, 2016).

Usualmente uno se confunde al observar que la clasificación posee valores de exactitud altos y se piensa que el clasificador funciona de manera correcta, pero estos son debido a la correcta clasificación de la clase mayoritaria que son un reflejo de la distribución de las clases minoritarias. Los algoritmos de clasificación fallan al clasificar la clase minoritaria ya que no se posee suficientes datos, esto muestra una tasa de error global alta por lo que es de suma importancia minimizarla, esto se realiza al dar más

importancia en el muestreo de la clase minoritaria. (Wei & Dunbrack, 2013) Para que el trabajo de investigación mejore los resultados se trabajará con una nueva base de datos balanceada, la base de datos modificada, se eliminaran los ESV que no posean como mínimo 100 eventos para el entrenamiento, por lo que no se toman en cuenta eventos HB, ICEQUAKE y RE. La nueva base de datos balanceada posee 102 eventos LP y 101 eventos VT.

En el capítulo IV se muestra la importancia del balanceo de datos, ya que se presentan modelos de clasificación con datos de entrenamiento sin balanceo y luego de balancear la base de datos, se aprecia claramente como mejoran los parámetros de rendimiento en el desempeño de clasificar ESV.

### 3.3.2. Normalización

Al analizar la base de datos inicial lo primero que se puede detectar en los datos es su gran diferencia en rangos numéricos en las características de los ESV, por ejemplo, en cualquier evento se observa que el valor de la varianza de la señal ronda los 0.01 hasta los 0.4, y en el caso de la entropía los valores van desde los 100 en adelante. Es debido a este gran rango numérico que abarca la base de datos que se busca una solución, haciendo un estudio se encuentra que la mejor solución es escalar los datos y los métodos más usados son la normalización y la estandarización.

La normalización es el proceso cuyo fin es la transformación de datos complejos a un conjunto de datos más simples, estables y pequeños. (MySQL, 2003). Se realiza este proceso para evitar la redundancia y anomalías de los datos, esto se ve reflejado en una mejora en el desempeño del clasificador, ya que las anomalías afectan significativamente en la generación de modelos. El uso de esta técnica evita la creación de dependencias y relaciones lo que nos deja con una base de datos limpia, de tamaño reducido y simple, lo que agilita los tiempos de procesamiento en ML. (Brownlee, 2019)

En otras palabras, la normalización usada en este trabajo de investigación es básicamente el escalar a todos los datos de las características dentro de un rango de 0 a 1. La fórmula para realizar la

normalización se ve en (19), este proceso es iterativo para todas las características de todos los ESV en la base de datos.

$$z = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (19)$$

Donde $z$ es el resultado normalizado de los datos, $x$ son los datos a normalizar, $\min(x)$ es el mínimo valor de los datos, y $\max(x)$ es el valor máximo de los datos. Esta normalización es lineal lo que transforma todos los datos de las características entre un rango de 0 y 1.

Para usar esta fórmula en Python se definirá una función llamada scale, como se observa en Segmento de código 1 una vez que se tiene la función se la puede llamar para transformar la base de datos en una normalizada. Cabe recalcar que el proceso de estandarización no se la realiza debido a que el rango de los datos va de -1 a 1 y ciertos algoritmos de clasificación solo permiten valores positivos.

**Segmento de código** 1, Función para Normalización

```
# Función de normalización
def scale(data): # data son los datos que se desean normalizar
    range_train=(data-data.min()).max()
    escala=(data-data.min())/range_train
    return escala
```

En el capítulo IV se muestra la importancia de normalizar los datos, se comprueba su importancia generando modelos de clasificación con datos sin normalizar y con los mismos datos normalizados. De esta forma se observa como mejoran los parámetros de rendimiento en el desempeño de clasificar ESV.

### 3.3.3. Selección de Características

Luego de las dos subetapas previas se reduce y normaliza la base de datos inicial a la base de datos modificada, la cual posee 84 características por cada ESV. Para realizar la clasificación con los algoritmos

de ML la selección de características es impórtate ya que de las 84 características algunas pueden ser irrelevantes para determinada variable e incluirla en el entrenamiento puede causar que el modelo incremente su complejidad y sea más difícil interpretarlo, el modelo puede resultar 'tonto' con clasificaciones inexactas e incrementa el tiempo de entrenamiento del modelo. (Luhaniwal, 2019)

La selección de características se trata de encontrar las características más relevantes e importantes de la base de datos modificada, esto ayudará a que el entrenamiento del algoritmo sea más veloz, crea un modelo con mejores clasificaciones, reduce la complejidad del modelo y reduce el sobreentrenamiento. Para esto existen tres métodos de selección de características, método de filtros, método embebido, y método de envoltura (Sayak, 2020), los cuales serán detallados a continuación.

Cabe recalcar que el método implementado en este trabajo ha sido el de envoltura por lo que su código será explicado igualmente. El capítulo IV muestra el desempeño de los modelos de clasificación con cada una de las tres técnicas del método de envoltura. De esta forma se observa como mejoran los parámetros de rendimiento en el desempeño de clasificar ESV.

#### 3.3.3.1. Método de filtro

El método de filtro se basa en generar un rango de importancia de las características, donde se eligen las características que más alto en el rango se encuentren. Algunas de las métricas para calcular el rango son la distribución chi-cuadrado usada para calcular la independencia de dos variables, Varianza donde las características constantes son eliminadas, Correlación para eliminar duplicados e información mutua para calcular la capacidad de clasificación de las variables objetivo. (Khandelwal, 2019)

#### 3.3.3.2. Método embebido

Este es un método iterativo que analiza cada iteración realizada en el proceso de entrenar al modelo y de manera minuciosa extrae las características que aporten más a cada una de las iteraciones en el entrenamiento. La manera de realizar ese proceso es mediante una penalización, se sanciona una

característica dado un rango de coeficientes, estos se eligen de acuerdo al método, el más común es el de regularización. Estos métodos también se denominan de penalización debido a las restricciones adicionales en la optimización de un algoritmo de clasificación que sesga el modelo hacia uno con menos características o complejidad. (Sayak, 2020)

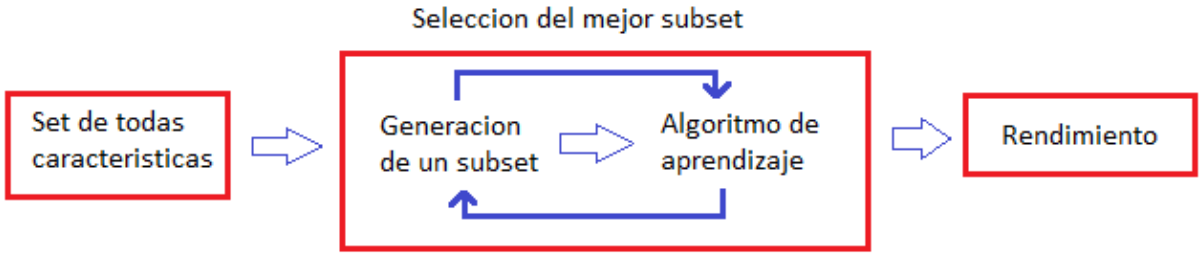
#### 3.3.3.3. Método de envoltura

Sigue un enfoque de búsqueda minucioso realizando una evaluación de todas las combinaciones posibles de características con el criterio de evaluación. El criterio de evaluación es la medida del rendimiento que depende del parámetro elegido, por ejemplo, se puede seleccionar el criterio de evaluación puede ser *P*, *A*, *f1-score*, *R*, *S*, *BER*, entre otros. El último paso es elegir la combinación de características que muestre los mejores resultados para el algoritmo de ML que se usara. (Luhaniwal, 2019)

La Figura 17 muestra un esquema del método de envoltura, este método busca un conjunto de características que mejor se acoplen al algoritmo de ML que se desee usar para mejorar su rendimiento. El proceso es iterativo ya que se divide en grupos denominados *subset* a las características para irlas probando en el algoritmo de ML, de esta forma se revisan todas las combinaciones posibles y se seleccionan las que mejor rendimiento presenten. Algunas técnicas de este método son, selección hacia adelante, eliminación hacia atrás y eliminación bidireccional. (Sayak, 2020)

**Figura 17**

*Método de envoltura*




- **Selección hacia adelante**

Esta técnica para empezar a codificarla necesita un algoritmo con el cual el modelo va seleccionando cada una de las características y filtrando de acuerdo al parámetro escogido, se requiere especificar un numero de características con las que se desea trabajar denominado *nivel de selección*. Segundo, se entrena todo el modelo reiteradas veces con una característica a la vez, seleccionando la que posea mejor desempeño en base a un parámetro especifico el cual es *f1-score*. A continuación, se vuelve a entrenar cada modelo previo aumentando una característica más. Finalmente, se seleccionan las características que cumplan que $C <$ *nivel de selección* sino se cumple esto se vuelve al paso anterior, siendo $C$ el número de características que se usan. (Khandelwal, 2019)

Para implementar esta técnica en Python se requiere importar la librería mlxtend la cual posee varias técnicas del método de envoltura. Para seleccionar la técnica a usar se importa la función SequentialFeatureSelector(), de esta función se selecciona la técnica a usar dependiendo la combinación que se ingrese, en el Segmento de código 2 se visualizan las librerías. La librería mlxtend.plotting posee el modulo plot_sequential_feature_selection que está diseñado para

graficar las características seleccionadas por las técnicas del método de envoltura. (Luhaniwal, 2019)

**Segmento de código 2**, Librería para el método de envoltura

```
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs
```

La función importada tiene los siguientes parámetros de entrada y su código se observa en el Segmento de código 3. Primero, se selecciona el algoritmo base para aplicarlo como estimador de todo el proceso, se selección el algoritmo SVM. Segundo los k_features que indican el número de características para ser seleccionadas, se usan las 84 características para ver su rendimiento. Los parámetros forward y floating definen que técnica usar, para selección hacia adelante los valores son foward = True y floating = False.

**Segmento de código 3**, Técnica selección hacia adelante

```
sffs = SFS(SVM(), #Algoritmo que se usara para selección de características
        k_features=84, #Número de características a usar
 #Foward y Floating para escoger la técnica de selección de características
        forward=True,
        floating=False,
        scoring = 'f1', #Parámetro de desempeño seleccionado
        cv = 0)
```

Ya seleccionada la técnica es momento de entrenarla con la función .fit() como se ve en el Segmento de código 4, este proceso se demora unos segundos y cuando finaliza el modelo ya tiene las características más relevantes. Con el código .k_feature_names_ se imprimen los nombres de las características finales y para poder visualizarlas se utiliza la librería del Segmento

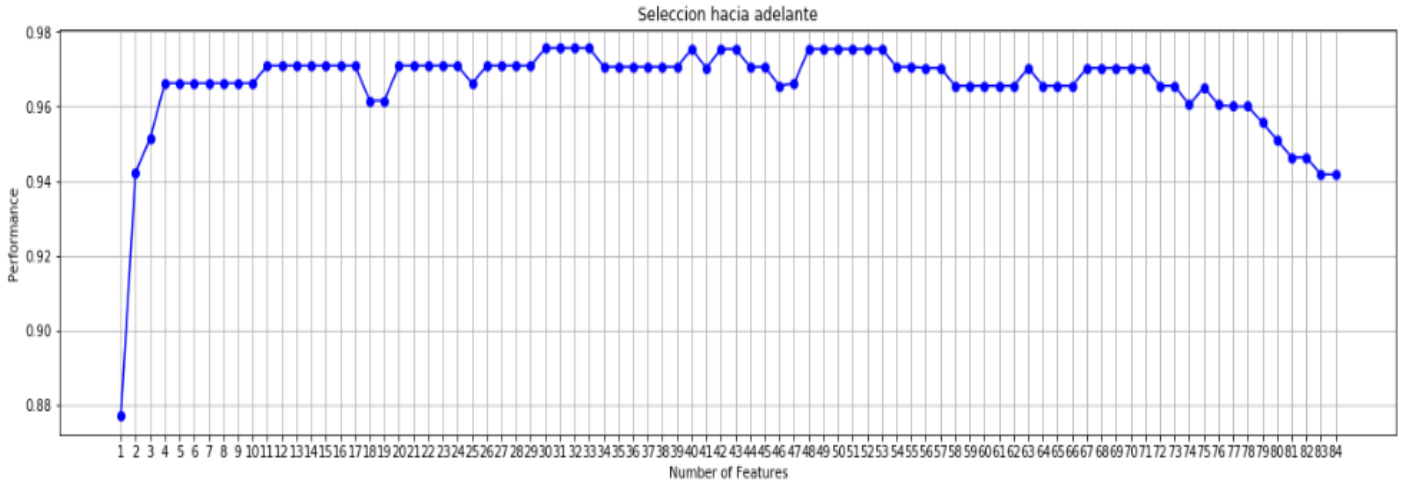
de código 2 y el parámetro que requiere de ingreso es el modelo que posee las características seleccionadas, la Figura 18 muestra el rendimiento del algoritmo de clasificación dependiendo el número de características usadas, en esta ocasión se usan las 84. Lo descrito anteriormente se encuentra en Segmento de código 4.

**Segmento de código 4**, Visualización de las características seleccionadas por el Método de Envoltura

```
sffs.fit(X, y) #Entrenando al modelo
#Impresión de la grafica de desempeño de selección de características
fig1 = plot_sfs(sffs.get_metric_dict(), figsize=(20,5),kind='std_dev')
plt.title('Sequential Forward Selection (w. StdErr)')
plt.grid()
plt.show()
sffs.k_feature_names_
```

**Figura 18**

*Rendimiento de la técnica de selección para adelante.*



La Figura 18 muestra el rendimiento del método de selección de características con las 84 esto se lo realiza debido a que no se sabe de primera instancia con qué número de características el algoritmo de clasificación es estable, observando la Figura 18 se deduce que con 30 características el algoritmo presenta el mejor desempeño, por lo que se vuelve a correr el Segmento de código 3 y Segmento de código 4 pero ahora con solo 30 características, de esta forma se obtiene un vector con las características finales, las características encontradas son 'f1', 'f2','f3', 'f4', 'f5', 'f6', 'f7', 'f8', 'f9', 'f10', 'f11', 'f12', 'f13', 'f15', 'f18', 'f19', 'f21','f26', 'f28', 'f30', 'f34', 'f38', 'f39', 'f44', 'f51', 'f59', 'f71','f72', 'f73','f82'.

Las siguientes técnicas presentan el mismo proceso al momento de su codificación solo varia en la selección de la técnica varían en los parámetros forward y floating por lo que solo se mostrara el valor necesario en cada parámetro, y su resultado gráfico.

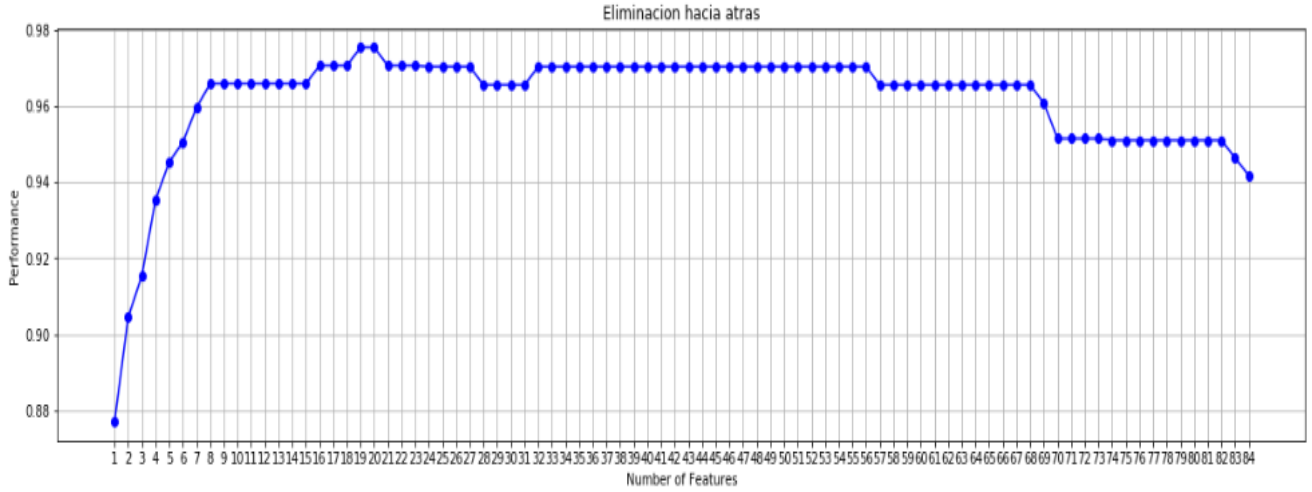
- **Eliminación hacia atrás**

Al contrario de la técnica anterior esta requiere un modelo lleno con todas las características para luego irlas removiendo dependiendo del *nivel de selección* y *C*. Una vez entrenado el modelo con las características se seleccionan las características que cumplan que *C* > nivel de selección, si esto no ocurre seguimos al siguiente paso, si esta condición se cumple se termina esta técnica. A continuación, se remueven características hasta que el proceso anterior se cumpla. (Khandelwal, 2019)

Para implementarlo en código los valores a cambiar son los parámetros forward y floating, para la eliminación hacia adelante los valores son foward = False y floating = False, vistos en el Segmento de código 3. Acto seguido se repite el Segmento de código 4 para visualizar la gráfica de desempeño del algoritmo con la técnica de eliminación hacia atrás. La Figura 19 muestra el rendimiento de esta técnica, donde indica que usando 20 características se logra el mejor desempeño, las características que se encontraron son 'f4', 'f9', 'f11', 'f12','f23', 'f28', 'f35','f43', 'f44', 'f45', 'f50', 'f51', 'f52', 'f53', 'f59', 'f60', 'f74', 'f77', 'f81','f82'.

**Figura 19**

*Rendimiento con la técnica de eliminación hacia atrás*




- **Eliminación bidireccional**

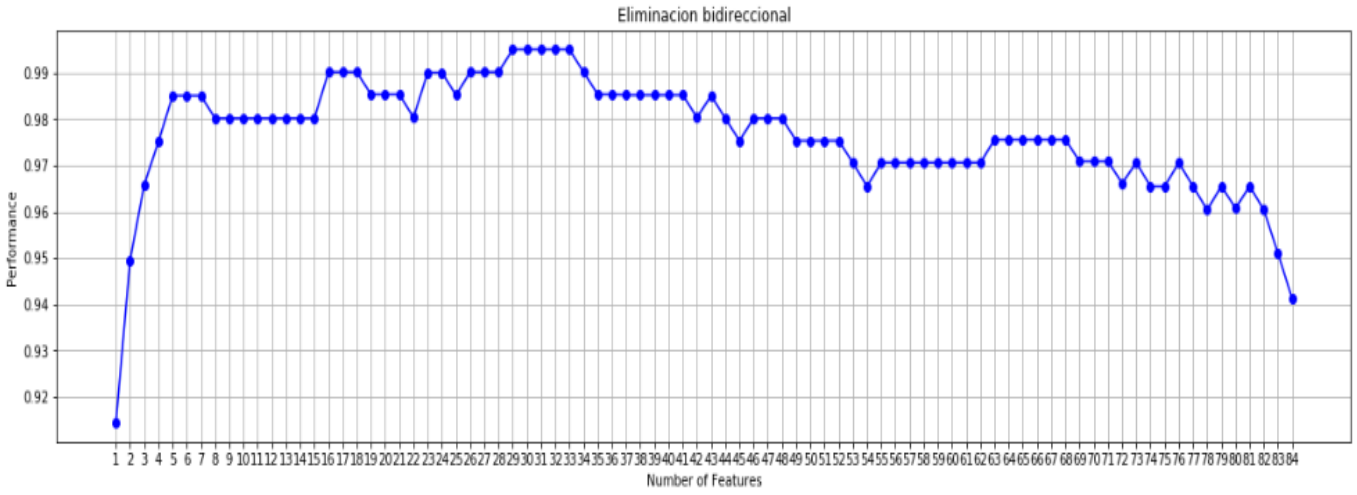
Ocupa ambas técnicas vistas anteriormente, requiere de dos niveles de selección primero como en selección hacia adelante al agregar una nueva característica verifica en base un nivel de selección también verifica la importancia de las características ya agregadas y si alguna de las características ya seleccionadas encontradas es insignificante respecto al otro nivel de selección, se usa la eliminación hacia atrás para eliminar esa característica insignificante. (Luhaniwal, 2019)

Para implementarlo en código los valores a cambiar son los parámetros forward y floating, para la eliminación bidireccional los valores son foward = True y floating = True, vistos en el Segmento de código 3. Acto seguido se repite el Segmento de código 4 para visualizar la gráfica de desempeño del algoritmo con la técnica de eliminación bidireccional. La Figura 19 muestra el rendimiento de esta técnica, donde indica que usando 30 características se logra el mejor desempeño, las características encontradas son 'f3', 'f4', 'f5', 'f7', 'f9', 'f11', 'f12', 'f13', 'f15', 'f20','f21', 'f24', 'f26', 'f28','f31', 'f32', 'f33', 'f34', 'f38', 'f42', 'f43', 'f55', 'f56','f57', 'f58', 'f61', 'f62', 'f65', 'f72', 'f73'.

El desempeño de las técnicas del método de envoltura que se han implementado en esta sección se analizaran en el Capítulo IV donde se compara el rendimiento de estos en los algoritmos de clasificación, de esta forma se selecciona el de mejor desempeño para continuar con el trabajo de investigación.

**Figura 20**

*Rendimiento de la técnica de eliminación bidireccional*




### 3.3.4. Algoritmo de Machine Learning

El objetivo de usar ML es generar modelos de clasificación, estos modelos son programas creados a partir de patrones numéricos identificados en los datos de entrenamiento, estos patrones numéricos se hallan al aplicar los diferentes algoritmos de clasificación en los datos. Los algoritmos de clasificación DT, RF, KNN, SVM, NB y XG como ya se explicó en el capítulo 2 en la sección de Machine Learning siguen el mismo proceso de entrenamiento y generación de un modelo de clasificación. Para comenzar a entender el código se explicará paso a paso el entrenamiento y la generación de un modelo de clasificación en JN, en Python existen librerías que permitirán usar los seis algoritmos de clasificación y de igual forma

entrenar sus modelos, las cuales serán explicadas más adelante, pero antes se merecen una mención dos librerías fundamentales para manipular de forma adecuada y rápida los datos, estas son *Numpy* y *Pandas*. *Numpy* es la librería para analizar datos estadísticos, también se puede usar como un contenedor multidimensional eficiente de datos genéricos y datos arbitrarios permitiendo que se integre sin problemas y rápidamente con una amplia variedad de bases de datos. (NumFOCUS, Numpy, 2020). *Pandas* es una herramienta de código abierto para la manipulación y análisis de datos especialmente para los *dataframes* es rápida, potente, flexible y fácil de usar. (NumFOCUS, pandas, 2020).

Para empezar a trabajar en la codificación de ML lo primero que se debe hacer es leer la base de datos modificada en JN. La base de datos proporciónada es un archivo con extensión .mat el cual se lo debe convertir a .csv para usarlo en JN. Una vez que se convierte este archivo en .csv se usa la función pd.read_csv(“ ”), la cual permite leer la base de datos, esto se ve en el Segmento de código 5. Para poder manipular la base de datos se la asigna a una variable, en este caso df, ahora para comprobar la base de datos se hace uso de las funciones de *Panda.* Se usa la función head() que muestra la cabecera de los 5 primeros ESV de la base de datos asignada a la variable df, así comprobamos que este correcta, su resultado se muestra en la Figura 21. Cabe recordar que la base de datos posee 84 características por lo que al usar la función head() se formaría una matriz de 5x85, para comodidad se muestran las 10 primeras columnas seguido de puntos suspensivos que muestran el resto de las 84 características.

**Segmento de código** 5, Importar librerías y Lectura de Base de Datos modificada

```
#Importar Librerias
import pandas as pd # Importar de Pandas para usar datos con dataframes
import numpy as np # Importar Numpy para analisis estadistico de los datos

#Leer Base de Datos
df=pd.read_csv("LPVT.csv") #Lectura de la base de datos
```

```
df.head() #Cabecera de la base de datos
```

**Figura 21**

*Resultado de mostrar la cabecera de la base de datos en JN*

| | Evento | f1 | f2 | f3 | f4 | f5 | f6 | f7 | f8 | f9 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | LP | -0.000008 | 0.134823 | 0.018177 | 119.586715 | 12.708361 | 1.166685 | 8.48 | 0.134802 | 1.814585 | ... |
| 1 | LP | -0.000034 | 0.205026 | 0.042036 | 126.565770 | 7.289889 | 1.786243 | 3.62 | 0.204966 | 1.960645 | ... |
| 2 | LP | -0.000102 | 0.159451 | 0.025425 | 124.766277 | 8.681597 | 1.771865 | 2.89 | 0.159415 | 1.943892 | ... |
| 3 | LP | 0.000103 | 0.214117 | 0.045846 | 145.765922 | 6.771051 | 1.671625 | 1.52 | 0.214061 | 1.971644 | ... |
| 4 | LP | -0.000019 | 0.168879 | 0.028520 | 232.074001 | 6.035482 | 1.853018 | 15.68 | 0.168855 | 1.759395 | ... |

5 rows × 85 columns

Acto seguido, se procede a separar las características de los datos y sus etiquetas como se observa en Segmento de código 6, las etiquetas son los nombres de los ESV que se tiene en la base de datos inicial. Se puede eliminar las columnas de la base de datos que se desee con la función .drop(), en este caso se elimina la columna que posee las etiquetas de los ESV y se asigna a una variable denominada X y las etiquetas de los ESV se asigna a una variable y. Por último se normaliza los datos usando la función scale() definida en Segmento de código 1, y el resultado de normalizar la base de datos se muestra en la Figura 22.

**Segmento de código 6,** Normalización de la base de datos

```
X = df.drop('Evento',axis=1) #Remover una columna de la matriz
y = df['Evento'] #Crear matriz de una columna
X=scale(X) #Normalizar la base de datos
X.head() #Cabecera
```

**Figura 22**

*Cabecera de datos normalizada*

| | f1 | f2 | f3 | f4 | f5 | f6 | f7 | f8 | f9 | f10 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.349342 | 0.133358 | 0.066495 | 0.079949 | 0.527794 | 0.142715 | 0.320357 | 0.133355 | 0.546658 | 0.634372 | ... |
| 1 | 0.328046 | 0.392472 | 0.254529 | 0.090190 | 0.220678 | 0.703506 | 0.123277 | 0.392338 | 0.906649 | 0.292536 | ... |
| 2 | 0.272339 | 0.224258 | 0.123613 | 0.087550 | 0.299559 | 0.690492 | 0.093674 | 0.224203 | 0.865359 | 0.480196 | ... |
| 3 | 0.440626 | 0.426027 | 0.284560 | 0.118365 | 0.191271 | 0.599760 | 0.038118 | 0.425909 | 0.933760 | 0.264632 | ... |
| 4 | 0.340183 | 0.259057 | 0.148011 | 0.245014 | 0.149579 | 0.763946 | 0.612328 | 0.259049 | 0.410633 | 0.432986 | ... |

5 rows × 84 columns

A continuación, se exportan las librerías necesarias para entrenar al modelo las cuales se observa en Segmento de código 7. La librería sklearn.model_selection permite llamar al objeto train_test_split, su función es realizar *cross validation* a la base de datos, seleccionando nosotros el porcentaje de datos para entrenamiento y el porcentaje para pruebas. Para poder acceder a los algoritmos de clasificación se tiene que importar cada uno de ellos con su respectiva librería, para una mejor explicación se usara solo un algoritmo, SVM, el resto del código se puede observar en la sección de Anexos. La librería sklearn.svm permite importar el algoritmo SVC que es el mismo que SVM con diferente nomenclatura. Finalmente la librería sklearn.metrics importa las métricas classification_report la cual nos entrega las métricas de *P, R, f1-score* y soporte, por otro lado, confusion_matrix permite imprimir la matriz de confusión. Finalmente,

**Segmento de código 7,** Librerías para métricas, división y modelo

```
#Importar Librerías
from sklearn.model_selection import train_test_split #Libreria para cross validation
from sklearn.metrics import classification_report,confusion_matrix #libreria para metricas de desempeno
from sklearn.svm import SVC #Libreria de algoritmo SVM
```

El Segmento de código 8, presenta la técnica de *cross validation* la cual reserva una parte de la base de datos para usarla como datos de prueba y el resto como datos de entrenamiento, la función que permite hacer esto es train_test_split donde se ingresa el porcentaje que se desea usar como datos de prueba, para este trabajo se esa un 40% en test_size y en random_state se coloca un número entero, en este caso 10, esto no es más que el inicio de los números aleatorios que decidirá la división de datos de prueba y entrenamiento. (Ray, 2017). La función train_test_split devuelve cuatro vectores, dos de ellos son los datos de entrenamiento (X_train_svm) y las etiquetas de estos datos (y_train_svm) y los otros dos son las etiquetas de los datos de prueba (y_test_svm) y sus datos (X_test_svm). El siguiente paso es instanciar al modelo como un objeto, se lo hace de la siguiente forma svm=SVC(), con el fin de usar sus funciones. La función que permite entrenar al modelo es .fit(), esta requiere dos parámetros, los datos de las características de los ESV que se encuentran en X_train_svm y las etiquetas de cada ESV que está en la variable y_train_svm, estos los datos de entrenamiento que se separaron usando train_test_split.

**Segmento de código 8,** División y Entrenamiento

```
#Uso de crossvalidation para obtener datos de prueba y datos de entrenamiento
X_train_svm, X_test_svm, y_train_svm, y_test_svm = train_test_split(X, y, test_size=0.4, random_state= 10)

svm = SVC() #Instanciación de función svm
svm.fit(X_train_svm,y_train_svm) #Entrenamiento de modelo svm
```

### 3.3.5. Análisis de Resultados

Para evaluar el desempeño del modelo generado se debe usar nuevos datos, no aplicados en el entrenamiento del modelo, en esta sección para evaluar al modelo se utiliza los datos de prueba

X_test_svm y y_test_svm que se separaron con *cross validation* en el Segmento de código 8. Para usar el modelo svm como un clasificador se le aplica la función .predict() que como parámetro de entrada requiere los datos de prueba X_test_svm con el objetivo de realizar la clasificación de los datos de prueba, lo que devuelve el modelo usando la función .predict() es un vector con las etiquetas, LP o VT, este vector se iguala a la variable svmpred, en el Segmento de código 9 se observa su código. El vector svmpred con las etiquetas es el resultado de clasificar los datos de prueba X_test_svm. Para poder analizar el desempeño del clasificador se usa la función classification_report como parámetros requiere el vector con las etiquetas reales de los ESV y_test_svm y el vector resultante de la clasificación del modelo svmpred. El resultado que muestra esta función es un indicador importante del desempeño del modelo, en este caso se obtiene un 93% de *P* al momento de clasificar eventos LP y un 100% de *P* en la clasificación de eventos VT.

**Segmento de código 9,** Desempeño del modelo

```
svmpred = svm.predict(X_test_svm) #Clasificacion de datos de prueba
etiqueta=['LP','VT']
#Impresion de resultados P y R
print(classification_report(y_test_svm,svmpred,labels= etiqueta))
              precision   recall   f1-score   support

      LP      0.93      1.00      0.97       43
      VT      1.00      0.92      0.96       39
```

La matriz de confusión resulta de utilidad para analizar los TP, TN, FP y FN obtenidos en la clasificación, su función es confusion_matrix() su código se implementa como se muestra en Segmento de código 10, los parámetros que se ingresan son los mismos que en la función classification_report. Para visualizar la matriz de confusión se puede usar la función print() pero para esta explicación se imprimirá

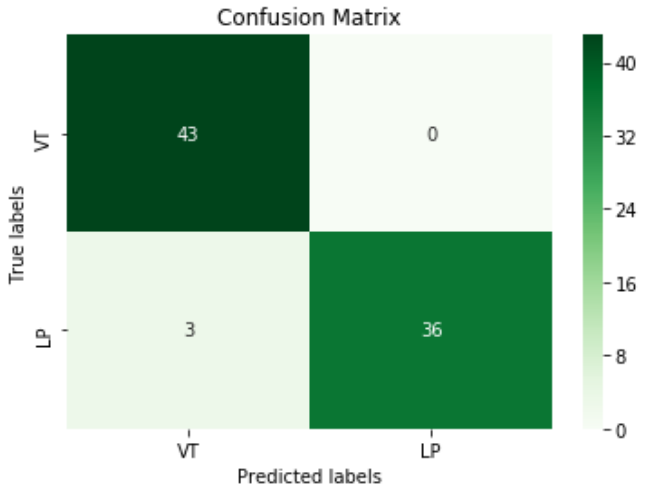
la matriz de forma que el lector entienda su funcionamiento. El resto del código simplemente es la forma en la cual será visualizada la matriz, colocando títulos e identificando las etiquetas verdaderas y_test_svm y las etiquetas clasificadas svmpred, esta matriz se observa en la Figura 23.

**Segmento de código 10,** Matriz de Confusión

```
cm=confusion_matrix(y_test_svm,svmpred,labels=label) #Generación de matriz de confusión
#Forma visual de impresión de una matriz de confusión
ax= plt.subplot()
sns.heatmap(cm, annot=True, ax = ax,cmap="Greens");
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
ax.set_xticklabels(label)
ax.set_yticklabels(label)
```

**Figura 23**

*Resultado Segmento de código 7 visualización de la Matriz de confusión*




Con la matriz de confusión se pueden distinguir de manera rápida las variables de los TP, TN, FP y FN y obtener sus valores, como se explicó en el capítulo II en la sección 2.3.5 Métricas en Clasificación,

con estos datos se puede calcular los parámetros de medición *A, P, R, S , f1-score, y BER,* el código de cada formula se ve en el Segmento de código 11. Para visualizar las respuestas de cada parámetro se usa la función print() como se muestra en el Segmento de código 12.

**Segmento de código 11,** Formulas de los Parámetros de Medición

```
tp, fn, fp, tn = cm.ravel() #asignación de elementos de la matriz de confusión
c= tp + tn
t=c+fn+fp
A=c/t       #Formula de Exactitud
S=tn/(tn+fp) #Formula de Especificidad
R=tp/(tp+fn) #Formula de Sensibilidad
P=tp/(tp+fp) #Formula de Precisión
f1=(2*P*R)/(P+R) #Formula de f1-score
BER=1-((S+R)/2) #Formula de BER
```

**Segmento de código 12,** Impresión de los Parámetros de Medición

```
#Impresión de A,S,R,P
print(f'Exactitud     -- A = {A*100:.0f} %') #Impresión de Exactitud
print(f'Especificidad  -- S = {S*100:.0f} %') #Impresión de Especificidad
print(f'Sensibilidad   -- R = {R*100:.0f} %') #Impresión de Sensibilidad
print(f'Precision      -- P = {P*100:.0f} %') #Impresión de Precisión
print(f'F1-score       -- f1 ={f1:.3f}') #Impresión de F1-score
print(f'BER          -- BER ={f1:.3f}') #Impresión de BER

#Resultados
Exactitud     -- A = 95.08 %
Especificidad  -- S = 100.00 %
Sensibilidad   -- R = 91.18 %
Precision      -- P = 100.00 %
F1-score       -- f1 =0.954
BER          -- BER =0.055
```

Para comparar la efectividad de cada algoritmo es necesario repetir todo lo que se ha explicado en este capítulo para cada algoritmo, para organizar de mejor forma el código de este capítulo se lo dividirá en dos partes, se creará una librería en Python y un código en JN, esto se explicará en la siguiente sección.

### 3.4. Creación de librería en Python y código de JN

Este trabajo de investigación busca como objetivo analizar el desempeño de los mejores clasificadores por lo que organizar y reducir el código facilitara clasificar ESV con varios algoritmos. Como se vio en la sección anterior todo el código escrito fue solo para el algoritmo de SVM, si se busca utilizar varios algoritmos de clasificación el código terminaría muy extenso, por este motivo se busca organizar el código para utilizar los seis algoritmos de clasificación con menos líneas de código.

La forma más efectiva que se encontró para reducir y organizar el código es la creación de una librería en Python con múltiples módulos de los algoritmos de clasificación y un archivo con el código de JN donde se usara la librería de Python. Una librería se define como un conjunto de implementaciones funcionales y un módulo es una porción del programa. (Pythones, 2017) Para crear una librería esta se genera en cualquier editor de texto; el nombre del archivo no debe contener espacios y luego del nombre se escribe la extensión *.py* de esta forma se tiene una librería de Python dentro de la cual se codifican los módulos a usar. El primer paso para crear un módulo es definirla con def seguido por un nombre y entre paréntesis los parámetros que necesita de entrada, en este caso serán los datos a clasificar. Los paramentos de salida se definen con la palabra return seguido por las variables que devolverá la función.

#### 3.4.1. Librería con módulos de los algoritmos de clasificación

La librería creada se denominada *Classification* es la que se usará para obtener el desempeño de los algoritmos. Primero nuestra librería usara otras librerías de Python ya que todas ellas poseen módulos

y funciones necesarias para la generación de un modelo por cada algoritmo de clasificación, las librerías se encuentran en el Segmento de código 13. Ya se habló de muchas de estas librerías, el resto de ellas son las librerías donde se localizan los módulos de los seis algoritmos de clasificación a usar.

**Segmento de código 13,** Librerías necesarias en *Classification*

```
import pandas as pd # Importar de Pandas para usar datos con dataframes
import numpy as np # Importar Numpy para analisis estadistico de los datos
from sklearn.model_selection import train_test_split #Libreria para cross validation
from sklearn.preprocessing import LabelEncoder #Libreria para transformar string a int
from sklearn.svm import SVC #Libreria de algoritmo SVM
from sklearn.ensemble import RandomForestClassifier #Libreria de algoritmo RF
from sklearn.neighbors import KNeighborsClassifier #Libreria de algoritmo KNN
from sklearn.naive_bayes import MultinomialNB #Libreria de algoritmo NB
from xgboost import XGBClassifier #Libreria de algoritmo XG
from sklearn.tree import DecisionTreeClassifier #Libreria de algoritmo DT
```

A continuación se define la función de normalización vista en el Segmento de código 1, luego se prosigue a definir tres vectores los cuales poseen las características de cada técnica del método de envoltura, esto se muestra en el Segmento de código 14. Estos vectores sirven para definir las características que se usarán en el entrenamiento de los modelos de clasificación, pueden ser cambiadas entre las tres, pero el programa usara ahora la técnica de selección hacia adelante.

**Segmento de código 14,** Características de las técnicas del método de envoltura

```
#SELECCION DE CARACTERISITCAS
#HACIA ADELANTE
adelante=['f1', 'f2','f3', 'f4', 'f5', 'f6', 'f7', 'f8', 'f9', 'f10', 'f11', 'f12', 'f13', 'f15', 'f18', 'f19', 'f21','f26', 'f28', 'f30', 'f34', 'f38', 'f39', 'f44', 'f51', 'f59', 'f71','f72', 'f73','f82']
#HACIA ATRAS
atras=['f3', 'f4', 'f9', 'f11','f23', 'f28', 'f35','f43', 'f44', 'f45', 'f50', 'f51', 'f52', 'f53', 'f59', 'f60', 'f74', 'f77', 'f81','f82']
```

```
#BIDIRECCIONAL
bidi=['f3', 'f4', 'f5', 'f7', 'f9', 'f11', 'f12', 'f13', 'f15', 'f20','f21', 'f24', 'f26', 'f28','f31', 'f32', 'f33', 'f34', 'f38', 'f42', 'f43', 'f55', 'f56','f57', 'f58', 'f61', 'f62', 'f65', 'f72', 'f73']
```

Posteriormente se ingresa la base de datos modificada de los ESV y se asigna los datos con las características a la variable Xr y las etiquetas a la variable yr. Segundo se utilizan las características encontradas con la técnica de selección hacia adelante, para que la matriz de datos Xr trabaje solo con estas características se utiliza el código Xr=Xr[adelante], si se desea usar otra técnica solo se cambia el vector adelante por uno de los definidos en Segmento de código 14. A continuación, se normalizan los datos y se aplica *cross validation* con la función train_test_split donde se separa el 70% de la base de datos para entrenamiento, esto se encuentra descrito en el Segmento de código 15.

**Segmento de código 15,** Base de datos para entrenamiento

```
train=pd.read_csv("LPVT.csv") #Lectura de base de datos para entrenamiento
Xr = train.drop('Evento',axis=1) #Base de datos sin etiquetas
yr = train['Evento'] #Base de datos solo las etiquetas
#SELECCION de CARACTERISITCAS
Xr=Xr[adelante] #Tecninca hacia adelante
Xr=scale(Xr) #Base de datos normalizada
#Uso de cross validation
X, X_test, y, y_test = train_test_split(Xr, yr, test_size=0.3, random_state= 10)
```

Acto seguido, se crea una función para cada uno de los algoritmos de clasificación, se explicará una de las seis funciones y el resto de ellas se podrá observar en la sección de Anexos. El modulo del algoritmo KNN se muestra en el Segmento de código 16. Primero, se requiere transformar las etiquetas de LP y VT a valores numéricos ya que el algoritmo KNN no clasifica con etiquetas tipo *string*, para esto se necesita la función LabelEncoder(), para transformar las etiquetas a valores numéricos se coloca la matriz

de etiquetas en la función labelencoder_y.fit_transform(y) dando como resultado un nuevo vector con etiquetas numéricas.

A continuación, para crear el modelo del algoritmo KNN se define una la variable knn, esta será el modelo, se iguala a la función del algoritmo KNN que requiere de tres parámetros para su funcionamiento, su código es knn = KNeighborsClassifier(). Los parámetros del algoritmo son los siguientes n_neighbors=20 que es el número de vecinos que el algoritmo KNN usara para el entrenamiento, metric y p sirven para seleccionar la distancia entre los vecinos y la distancia estándar usada en este algoritmo es la distancia euclidiana, para seleccionar la distancia euclidiana se requieren los siguientes parámetros metric = 'minkowski' y p = 2. (ScikitLearn, sklearn neighbors KNeighborsClassifier, 2018). Una vez definido los parámetros del algoritmo, el siguiente paso es entrenar al modelo y se usa la función .fit() y se ingresan las etiquetas y los datos de entrenamiento, anteriormente se definió que se trabajaría con el 70% de la base de datos para entrenamiento. Antes de realizar la clasificación usamos la función del Segmento de código 1 para normalizar los datos de prueba o cualquier nuevo ESV que se desee clasificar. Con la función .predict() se realiza la clasificación de los nuevos datos y entrega un vector con los ESV clasificados con sus etiquetas. Por último, se devuelve el vector con las etiquetas clasificadas en la variable asignada knpred luego de la palabra return.

**Segmento de código 16,** Módulo de clasificación de KNN

```
# Modulo de clasificador KNN
def clasificacionKN(test):
    labelencoder_y = LabelEncoder() # Función transformar string a int
    yk = labelencoder_y.fit_transform(y)
    # Algoritmo KNN
    knn = KNeighborsClassifier(n_neighbors = 20, metric = 'minkowski', p = 2)
    knn.fit(X,yk) # Entrenamiento de modelo KNN
    Xtest=scale(test) # Normalizacion de los datos de prueba
```

```
    knpred = knn.predict(Xtest) # Clasificacion de datos de prueba
    return knpred # Retorno de ESV clasificados
```

El módulo visto en el Segmento de código 16 es el cual se replica para los algoritmos DT, RF, SVM, NB, y XG, su código se puede observar en la sección de Anexos. Todos estos módulos generan un modelo de clasificación con cada uno de sus algoritmos respectivamente, los seis clasificadores son entrenados bajo las mismas condiciones, es decir, con los mismos datos de entrenamiento y para ver su desempeño se usan los mismos datos de prueba en los seis clasificadores. El desempeño de los clasificadores se puede ver en el capítulo IV.

Para evaluar el rendimiento de los clasificadores se ha creado un módulo adicional denominada *desempeño* la cual permite evaluar las métricas *P, A, R, S*, *f1-score, y BER,* el código se observa en Segmento de código 17. Como parámetro de entrada para el módulo *desempeno* se requiere la matriz de confusión del clasificador. A continuación, se usa la función .ravel() la cual retorna los elementos de una matriz en forma de vector, cada uno de estos elementos del vector se los asigna a una variables. Finalmente, las variables obtenidas son *TP, TN, FP* y *FN* con estas se calcula las métricas *A, S, R* y *P* con las formulas explicadas en el capítulo II en la sección 2.3.5 Métricas en Clasificacións, los resultados de las métricas son los parámetros de salida de este módulo. Los parámetros de desempeño *A, S, R* y *P* muestran los resultados de los de la clasificación de cada algoritmo, esto permite escoger el algoritmo que mejor desempeño demuestre en la clasificación de ESV.

**Segmento de código 17,** Modulo *desempeño* con las fórmulas para evaluar las métricas de desempeño

```
def desempeno(cm):
    tp, fn, fp, tn = cm.ravel() #asignación de elementos de la matriz de confusión
    c= tp + tn
```

```
        t=c+fn+fp
        A=c/t       #Formula de Exactitud
        S=tn/(tn+fp) #Formula de Especificad
        R=tp/(tp+fn) #Formula de Sensibilidad
        P=tp/(tp+fp) #Formula de Precisión
        f1=(2*P*R)/(P+R) #Formula de f1-score
        BER=1-((S+R)/2) #Formula de BER
        return A,S,R,P,f1
```

Los módulos descritos hasta ahora forman la librería *Classification*, para usar esta librería en Python se guarda el archivo con extensión *.py* de esta forma se la puede utilizar en JN el código de la librería completa se ve en la sección de Anexos.

### 3.4.2. Código de programa principal

El código del programa principal se lo realiza en JN tiene como finalidad ser un código sencillo para clasificar ESV, solo se requiere insertar los datos con las características de los ESV en uno de los módulos de la librería *Classification* y de esta forma se obtendrán los ESV clasificados en forma de un vector. El usuario no tiene que preocuparse de tener que generar un modelo de clasificación, lo único que necesita es escoger con que algoritmo desea trabajar, ya que tiene seis opciones disponibles. En esta sección se explicará el código de JN y el uso de los módulos de la librería *Classification*.

Primero se importarán las librerías que se han usado a lo largo de este trabajo de investigación como se ve en el Segmento de código 18, la librería *Pandas* permite manipular los datos, de la librería *sklearn.metrics* se usa el módulo de matriz de confusión y *classification_report*, la librería *sklearn.preprocessing* con la que se puede transformar datos de tipo s*tring* a entero y por último de la librería que se creó *Classification* se importan todos sus módulos.

**Segmento de código 18,** Librerías para programa final

```
import pandas as pd # Importar de Pandas para usar datos con dataframes
import numpy as np # Importar Numpy para analisis estadistico de los datos
from sklearn.model_selection import train_test_split #Libreria para cross validation
from sklearn.preprocessing import LabelEncoder #Libreria para transformar string a int
from sklearn.metrics import classification_report,confusion_matrix #libreria para metricas de desempeno
from Clasificacion import desempeno,clasificacionNB, clasificacionDT, clasificacionKN, clasificacionRF, clasificacionSVM, clasificacionXG # Módulos de algoritmos de clasificación
```

La clasificación se realiza con los datos de prueba que se han extraído de la base de datos entregada por el IGEPN, a esta base de datos se la denomino *Test*, el código para poder trabajar con la base de datos es Data=pd.read_csv("Test.csv"). Luego se debe retirar la columna de etiquetas de los datos de prueba a clasificar esto se lo realiza con la función Data.drop('Evento',axis=1). Acto seguido la columna de etiquetas se coloca en otra variable ytest=Data['Evento'] y se transforma estas etiquetas a valores numéricos para algoritmos como NB y KNN con la función LabelEncoder(). Por último, se extraen solo las características con las que se va a trabajar de los datos de prueba, estas son las determinadas por las técnicas del método de envoltura y se las declara igual que en el Segmento de código 14 para realizar esto se escribe el código test=test[adelante]. Todo el código explicado en este párrafo se ve en el Segmento de código 19.

**Segmento de código 19,** Preprocesamiento de los datos a clasificar

```
Data=pd.read_csv("Test.csv") #Lectura de base de datos para entrenamiento
test = Data.drop('Evento',axis=1) #Base de datos sin etiquetas
ytest=Data['Evento'] #Base de datos solo las etiquetas
labelencoder_y = LabelEncoder() #Modulo para transformar string a int
yk = labelencoder_y.fit_transform(ytest) #Transformación de etiquetas LP, VT a 0,1
test=test[adelante] #Seleccion de características a usar
```

Para hacer uso de los módulos de clasificación lo que se requiere como parámetro de entrada son los datos de prueba y los datos de salida del módulo se igualan a una variable, esta variable posee los ESV clasificados en forma de vector. En el Segmento de código 20, se ejemplifica este proceso usando el módulo del algoritmo SVM, su código es svm=clasificacionSVM(test). Para usar los distintos módulos se repite este proceso para todos los algoritmos llamando a los distintos módulos. Con este código ya se obtienen ESV clasificados, por lo que para clasificar ESV futuros solo se requiere sus datos con las características y el código devuelve los datos clasificados entre eventos LP y VT.

**Segmento de código 20,** Uso del módulo SVM para clasificación de ESV

```
svm=clasificacionSVM(test) #Uso del módulo SVM para clasificación de ESV
```

Para analizar el desempeño de los clasificadores de este trabajo de investigación se agrega dos líneas de código las cuales seria la creación de la matriz de confusión la cual requiere los resultados del clasificador svm y las etiquetas reales de los ESV obtenidas en ytest como se ve en cm=confusion_matrix(ytest,svm,label) ya que el módulo *desempeño* requiere la matriz de confusión como parámetro de entrada y este módulo devuelve los parámetros *A, S, R, P, f1-score, y BER.* Para visualizar los parámetros de medición se usa la función print() con la que se imprime sus resultados.

**Segmento de código 21,** Uso del módulo desempeño para impresión de parámetros de medición

```
cm=confusion_matrix(ytest,svm,label) #Matriz de confusion
A,S,R,P=desempeno(cm) #Uso de modulo desempeño
print(f'Exactitud     -- A = {A*100:.2f} %') #Impresión de Exactitud
print(f'Especificidad  -- S = {S*100:.2f} %') #Impresión de Especificidad
print(f'Sensibilidad   -- R = {R*100:.2f} %') #Impresión de Sensibilidad
print(f'Precision     -- P = {P*100:.2f} %') #Impresión de Precisión
print(f'F1-score      -- f1 ={f1:.3f}') #Impresión de F1-score
print(f'BER          --  BER ={ber:.3f}') #Impresión de BER
```

```
#Resultados
Exactitud      -- A = 95.08 %
Especificidad  -- S = 100.00 %
Sensibilidad   -- R = 91.18 %
Precision      -- P = 100.00 %
F1-score       -- f1 =0.954
BER            --  BER =0.044
```

El proceso descrito en este capítulo se realizará con cada uno de los algoritmos de clasificación, para poder obtener los parámetros de rendimiento con los seis clasificadores, sus resultados se mostrarán en el capítulo IV y de esta forma se podrá comparar el rendimiento de los clasificadores.

# CAPÍTULO IV

## 4. Resultados

En este capítulo explicara paso a paso la importancia del preprocesamiento con sus subetapas balanceo de datos, normalización y selección de características. En cada etapa se generan modelos de clasificación para ver sus resultados, el entrenamiento de los modelos se realiza con datos antes y después de cada subetapa los datos se colocan en tablas que muestran los parámetros de rendimiento de los seis algoritmos donde se verifica como mejoran los modelos de clasificación.

Primero se entrena un modelo con la base de datos inicial, se analiza el desempeño de los parámetros de rendimiento y como sus resultados al clasificar los eventos son muy bajos se prosigue a la etapa de preprocesamiento. La etapa de preprocesamiento consta de tres subetapas, el balanceo de datos permite reducir la base de datos inicial generando una nueva base denominada base de datos modificada con la que se trabajara en las siguientes subetapas, de igual forma se entrena un modelo con esta la base de datos modificada para verificar si el desempeño mejora. Se continua con la subetapa de normalización de datos, aquí se entrena al modelo con una base de datos balanceada y normalizada que es la modificada, y se ve claramente como mejoran los resultados de la clasificación. Por último, las técnicas selección de característica usaran la base de datos modificada, luego de esta nueva base se usarán solo las características seleccionadas por cada técnica y se generaran seis modelos con cada una, analizando de esta forma que algoritmo y con qué técnica es el mejor clasificador según los parámetros de rendimiento.

### 4.1. Etapa de preprocesamiento

Para comprender la etapa del preprocesamiento se mostrará las medidas de rendimiento de los diferentes clasificadores en tres etapas comenzamos el análisis de resultados con la generación de modelos usando las 84 características disponibles en la base de datos inicial, con el fin de analizar su rendimiento. Seguido por la base de datos modificada, luego sobre esto se aplica la normalización y por

último se aplica los diferentes métodos de selección de características de envoltura a la base de datos modificada. Para la generación de los modelos de clasificación se usa el código desde el Segmento de código 13 hasta el Segmento de código 20 variando ciertas líneas de código que serán explicadas.

### 4.2. Base de datos Inicial

Para analizar los resultados de emplear los modelos de clasificación con una base de datos sin preprocesamiento se sigue el proceso de *cross validation* dividiendo el 70% de los datos para entrenamiento y el 30% para pruebas. Luego se aplica la librería que se ha creado para clasificar los datos, con el cambio mostrado en train=pd.read_csv("Eventos.csv"), la base de datos inicial con todos los eventos se titula *Eventos* y contiene 1145 ESV. Esto genera modelos nuevos que serán usados en el programa de JN, en este programa de igual forma se cambia la base de datos y se aplica el mismo porcentaje de *cross validation* para obtener los mismos ESV de prueba.

Para usar las funciones de la librería se las debe llamar a los módulos de la librería como en Segmento de código 18, luego se utilizan los módulos con los parámetro de entrada de los datos de prueba, los cuales salieron de la función train_test_split(). Se usa el módulo de clasificaciónDT como se muestra en Segmento de código 22 el cual devuelve un vector con etiquetas de los datos clasificados, los cuales se utilizan para generar la matriz de confusión y un reporte de los parámetros *P* y *R*.

**Segmento de código 22,** Código de clasificación y desempeño usando DT

```
dt=clasificacionDT(Eventos) #Clasificación de ESV usando DT
cm=confusion_matrix(ytest,dt,label) #Matriz de confusión
print(cm) # Impresión de matriz de confusión
print(classification_report(ytest,dt,label)) #Impresión de reporte
```

El Segmento de código 23 muestra los resultados de la matriz de confusión, se puede observar como existen 317 aciertos para clasificar eventos LP y 0 para los eventos VT. El resultado del reporte de

clasificación muestra un alto porcentaje en *P* y *R* basados en LP con el 94% y 98% respectivamente y basados en VT un 0% en *P* y *R*, algo obvio dado que no se clasificó ningún evento VT.

**Segmento de código 23,** Resultados de la matriz de confusión y de los parámetros *P* y *R* usando el algoritmo DT

```
[[317   5]
 [ 22   0]]
              precision    recall  f1-score   support

          LP       0.94      0.98      0.96       322
          VT       0.00      0.00      0.00        22
```

Este resultado tan desalentador se debe a una base de datos desbalanceada donde se presenta un total de 1044 eventos LP y 101 eventos VT. No se ve necesario mostrar los resultados del resto de clasificadores ya que al usar uno solo se observa una pésima clasificación usando la base de datos inicial. La importancia de balancear la base de datos queda demostrada además su teoría se explicó en el capítulo 3 en la sección 3.3.1 Balancear la base de datos. Por este motivo la nueva base de datos presenta 203 ESV, los eventos LP son 102 y los eventos VT 101.

### 4.3. Base de datos balanceada

Para medir el rendimiento de una base de datos balanceada solo es necesario realizar un cambio al leer la base de datos con la que se entrena al modelo, esto se lo realiza en el código train=pd.read_csv("LPVT.csv") de la librería que se creó anteriormente. La base de datos LPVT contiene los ESV balanceados, y se la ha denominado base de datos modificada. Con una base de datos balanceada se generan nuevos modelos de clasificación al momento de importar la librería y usar sus módulos en el programa de JN. Se repite el Segmento de código 22 con la diferencia que dé inicio se usara otro clasificador llamando a un módulo distinto que es rf=clasificacionRF(test), se usa este módulo debido a es

una mejora al algoritmo de DT, así se obtendrán los resultados de la clasificación usando el algoritmo de clasificación RF.

La impresión de la matriz de confusión y el reporte de clasificación se ve en Segmento de código 24, observando la matriz de confusión se nota una mejora al clasificar los eventos VT ya que se logran clasificar 25 de 27 eventos VT de los datos de prueba, antes no se lograba ninguna clasificación de estos y se logra clasificar 33 de 34 eventos LP.

**Segmento de código 24,** Resultados de la matriz de confusión y de los parámetros P y R usando el algoritmo RF con la base de datos balanceada

```
[[33  1]
[ 2 25]]
              precision    recall  f1-score   support

          LP       0.94      0.97      0.96        34
          VT       0.96      0.93      0.94        27
```

Dado que las medidas de rendimiento *P* y *R* han sido buenas usando una función de Python para calcularlas, se usará el módulo de *desempeño* para calcular *A* y *S* en base a LP, esta función se encuentra en el Segmento de código 17 y para poder visualizarla en JN se usa el Segmento de código 20. Aplicando este módulo los resultados que se obtienen se visualizan en la *Tabla 3* con resultados sumamente mejores a los obtenidos anteriormente.

**Tabla 3**

*Parámetros de rendimiento usando el clasificador RF*

| Clasificador | A[%] | S[%] | R[%] | P[%] |
|---|---|---|---|---|
| RF | 95.08 | 92.59 | 97.06 | 94.29 |

Los resultados son bastante alentadores logrando una mejora notable, pero esto es solo con el algoritmo de RF por lo que se vuelve a ejecutar el código para los seis algoritmos con los que se va a trabajar, los resultados se ven en *Tabla 4*. Algunos clasificadores como DT, RF y XG funcionan con resultados arriba del 90% en los parámetros de rendimiento, pero se observa que clasificadores como KNN y SVM siguen presentando muchas fallas incluso NB no puede trabajar con valores negativos por lo que da error en su uso. Para mejorar el rendimiento de estos clasificadores se realiza la normalización de los datos explicada en el capítulo 3 en la sección 3.3.2 Normalización.

**Tabla 4**

*Parámetros de rendimiento con base de datos balanceada*

| Clasificador | A[%] | S[%] | R[%] | P[%] |
|---|---|---|---|---|
| DT | 95.08 | 92.59 | 97.06 | 94.29 |
| RF | 95.08 | 92.59 | 97.06 | 94.29 |
| KNN | 45.90 | 35.29 | 59.26 | 42.11 |
| SVM | 44.26 | 100 | 0 | 0 |
| NB | - | - | - | - |
| XG | 95.08 | 97.06 | 92.59 | 96.15 |

### 4.4. Base de datos Normalizada

Para aplicar la normalización a los datos se usa el modulo del Segmento de código 1 este código se usa dentro de la librería, en el Segmento de código 15 se muestra como se aplica la función de *scale* a la base de datos modificada con la que se entrenara a los modelos de los clasificadores.

A los datos que se usarán como prueba, los seleccionados anteriormente de la base de datos con 1187 ESV o para nuevos ESV provenientes del volcán, se les tiene que aplicar la función *scale*. El momento para aplicar *scale* en los datos de prueba es cuando se hace uso de algún módulo de clasificación como se observa en el Segmento de código 16 usando el clasificador KNN. Una vez que se tiene el modelo de clasificación entrenado con datos normalizados y los datos de prueba igualmente normalizados se puede clasificar de forma correcta los ESV, esto como ya se lo ha mencionado se realiza con la función .predict(datos de prueba).

Al usar cualquier módulo de clasificación el proceso será el mismo por lo que se puede analizar las medidas de rendimiento de los clasificadores usando sus módulos. Se usará el módulo de KNN para analizar la clasificación usando la matriz de confusión y el reporte de clasificación, como en el Segmento de código 22. La impresión de los resultados se muestra en el Segmento de código 25, una mejora notable se observa en los resultados, se logra clasificar 33 de 34 eventos LP y 26 de 27 eventos VT, tan altas clasificaciones no se habían logrado anteriormente. Con una base de datos balanceada y normalizada la clasificación incrementa notablemente, se puede ver en los parámetros P con un 97% y R con 97% calculados en base a LP.

**Segmento de código 25,** Resultados de la matriz de confusión y de los parámetros P y R usando el algoritmo RF con la base de datos balanceada y normalizada

```
[[33  1]
[ 1 26]]
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        34
           1       0.96      0.96      0.96        27
```

Vale la pena ver el rédito del resto de clasificadores ya que el clasificador KNN obtuvo buenas medidas de rendimiento. Ahora el entrenamiento, la generación del modelo y la clasificación de ESV se realizará con el resto de algoritmos usando los módulos de la librería *Classification*, usando los códigos desde el Segmento de código 17 hasta el Segmento de código 20 con cada algoritmo de clasificación, sus resultados se mostraran todos juntos en la *Tabla 5*.

**Tabla 5**

*Parámetros de rendimiento usando los clasificadores de ML*

| **Clasificador** | **A[%]** | **S[%]** | **R[%]** | **P[%]** | **F1-score** | **BER** |
|---|---|---|---|---|---|---|
| DT | 87 | 74 | 97 | 82 | 0.889 | 0.101 |
| RF | 97 | 93 | 97 | 94 | 0.957 | 0.072 |
| KNN | 98 | 96 | 97 | 96 | 0.963 | 0.059 |
| SVM | 95 | 97 | 91 | 97 | 0.954 | 0.084 |
| NB | 95 | 97 | 91 | 96 | 0.954 | 0.084 |
| XG | 93 | 94 | 93 | 93 | 0.926 | 0.067 |

El clasificador que muestra el mejor desempeño en *R* es RF con 97%, en *P* es SVM con 97%, en *BER* es KNN con 0.059 y en *f1-score* es KNN con 0.963, como *f1-score* es el balance entre ambos parámetros se puede concluir que KNN es el clasificador que muestra el mejor desempeño al clasificar eventos LP. A pesar de que estos resultados son alentadores existe la posibilidad de mejorar más aun el rendimiento de los clasificadores y este método es la selección de características que se discutirán a continuación.

### 4.5. Base de datos con selección de características

Para la selección de características el método a usar es el de envoltura como se discutió en el capítulo 3 en la sección 3.3.3 Selección de Características, cabe recordar que el método de envoltura posee tres técnicas distintas las cuales serán implementadas.

- **Selección hacia adelante**

Primero se usa la técnica de selección para adelante donde se escogen las 30 características que permitan un desempeño del 0.99 como se muestra en la Figura 18. Una vez que se obtiene el vector con las 30 características seleccionadas por la técnica hacia adelante como se ve el Segmento de código 14, se procede a cambiar el código en la librería y en el programa de JN para solo usar las 30 características y no las 84. Para usar las 30 características se coloca el vector 'adelante' luego de leer la matriz de datos de entrenamiento y de prueba como se muestra en el Segmento de código 26.

**Segmento de código 26,** Seleccion de características de la base de datos modificada

```
train=pd.read_csv("LPVT.csv")
train=train [adelante]
```

Una vez que se obtiene la base de datos modificada con solo 30 características se vuelve a generar los modelos de clasificación al llamar a los módulos de la librería y se realiza las pruebas con la base de datos de prueba, esto es repetir desde el Segmento de código 13 hasta el Segmento de código 20 con los cambios ya mencionados. Así se pueden obtener las medidas de rendimiento de los algoritmos de clasificación que se visualizan en la *Tabla 6*.

**Tabla 6**

*Parámetros de rendimiento 30 características, técnica hacia adelante*

| Clasificador | A[%] | S[%] | R[%] | P[%] | F1-score | BER |
|---|---|---|---|---|---|---|
| DT | 92 | 85 | 97 | 89 | 0.93 | 0.089 |
| RF | 96 | 95 | 98 | 96 | 0.957 | 0.035 |
| KNN | 98 | 96 | 100 | 97 | 0.986 | 0.019 |
| SVM | 95 | 100 | 91 | 100 | 0.954 | 0.044 |
| NB | 95 | 100 | 91 | 100 | 0.954 | 0.044 |
| XG | 98 | 96 | 100 | 97 | 0.986 | 0.019 |

- **Eliminación hacia atrás**

Como muestra la Figura 19 se requieren 16 características para lograr un desempeño del 0.9 para clasificar los ESV. El vector 'atrás' definido en el Segmento de código 14 contiene estas características y se lo usa para cambiar la base de datos modificada en la librería de Python y en el programa de JN para usar las 16 características. El vector 'atrás' se coloca luego de leer la matriz de datos de entrenamiento y de prueba como se muestra en el Segmento de código 26.

Una vez que se tiene la base de datos modificada con solo las características de la técnica de eliminación hacia atrás se repite el código desde el Segmento de código 13 hasta el Segmento de código 20, estos códigos muestran el entrenamiento, generación de los modelos y la clasificación que desempeñan, de esta forma se obtienen sus parámetros de desempeño que se puede ver en la *Tabla 7*.

**Tabla 7**

*Parámetros de rendimiento 20 características, técnica de eliminación hacia atrás*

| Clasificador | A[%] | S[%] | R[%] | P[%] | F1-score | BER |
|---|---|---|---|---|---|---|
| DT | 84 | 93 | 76 | 93 | 0.839 | 0.155 |
| RF | 97 | 96 | 97 | 97 | 0.971 | 0.033 |
| KNN | 93 | 93 | 94 | 94 | 0.941 | 0.066 |
| SVM | 93 | 93 | 94 | 94 | 0.941 | 0.066 |
| NB | 90 | 89 | 91 | 91 | 0.912 | 0.1 |
| XG | 95 | 93 | 97 | 94 | 0.957 | 0.045 |

- **Eliminación Bidireccional**

En la Figura 20 se muestra el número de características necesarias para la técnica de eliminación bidireccional y son 30 características de las 84. El vector 'bidi' definido en el Segmento de código 14 contiene estas características y se lo usa para cambiar la base de datos modificada en la librería de Python y en el programa de JN para usar las 30 características seleccionadas. El vector 'bidi' se coloca luego de leer la matriz de datos de entrenamiento y de prueba como se muestra en el Segmento de código 26.

Una vez que se tiene la base de datos modificada con solo las características de la técnica de eliminación hacia atrás se repite el código desde el Segmento de código 13 hasta el Segmento de código 20, estos códigos muestran el entrenamiento, generación de los modelos y la clasificación que desempeñan, de esta forma se obtienen sus parámetros de desempeño que se puede ver en la *Tabla 8*.

**Tabla 8**

*Parámetros de rendimiento 30 características, técnica de eliminación bidireccional*

| Clasificador | A[%] | S[%] | R[%] | P[%] | F1-score | BER |
|---|---|---|---|---|---|---|
| DT | 93 | 89 | 97 | 92 | 0.943 | 0.07 |
| RF | 97 | 93 | 100 | 94 | 0.971 | 0.037 |
| KNN | 95 | 93 | 97 | 94 | 0.957 | 0.052 |
| SVM | 95 | 96 | 94 | 97 | 0.955 | 0.048 |
| NB | 95 | 93 | 97 | 94 | 0.957 | 0.052 |
| XG | 98 | 96 | 100 | 97 | 0.986 | 0.019 |

### 4.6. Resultado de los clasificadores

El proceso de generar modelos de clasificación y evaluar sus parámetros de rendimiento se lo realizo en el software JN y Python 3 en una computadora con procesador Core™ i7 @2.40GHz y 8GB de RAM, con un sistema operativo de 64 bits. Los parámetros más importantes para evaluar pueden variar dependiendo la necesidad de los ESV a clasificar, según el IGEPN en (IGEPN, 2016) un aumento previo de los eventos LP han estado presentes en casi todas las erupciones del volcán Tungurahua, por este motivo la clasificación correcta de este evento será analizada, se tomará en cuenta los parámetros *f1-score* y *BER* debido a que son utilizan *P* y *R* para ser calculados. Otro enfoque para evaluar el parámetro más importante se lo obtiene de estudios previos sobre clasificación de ESV, en (Benitez, Paillacho, Rojo, & Lara, 2017), (Lara, Carrera, Benítez, Ruiz, & Rojo, 2016) se enfocan más en determinar el mejor desempeño en *A* y *BER*, por otro lado en (Malfante, Mura, Métaxian, & Mars, 2018) (Witsil & Jhonnson, 2019) (Ren, Peltier, Ferrazzini, Rouet-Leduc, & Jhonns, 2018) muestran mayor interés en *P*, *A, y BER*. Estas investigaciones muestran que parámetros serían los más importantes para estudiar en el rendimiento de

los clasificadores generados en este trabajo de investigación, las cuales se analizaran en el siguiente capítulo en la sección 5.1 Comparación con trabajos similares. En la etapa de selección de características cada una de las técnicas usa los mismos datos de prueba en los seis modelos de clasificación, de esta forma se puede comparar correctamente los resultados de la clasificación y sus medidas de rendimiento.

La *Tabla 9* muestra los resultados del parámetro *f1-score* de las tres técnicas de selección de características, se observa que los mejores resultados se obtienen con la técnica de selección hacia adelante, usando los algoritmos de KNN y XG, ambos obtienen el mismo puntaje de 0.986.

**Tabla 9**

*Comparación de resultados entre las técnicas en F1-score*

| Clasificador | Selección hacia adelante | Eliminación hacia atrás | Eliminación bidireccional |
|---|---|---|---|
| DT | 0.93 | 0.839 | 0.943 |
| RF | 0.957 | 0.971 | 0.971 |
| KNN | **0.986** | 0.941 | 0.957 |
| SVM | 0.954 | 0.941 | 0.955 |
| NB | 0.954 | 0.912 | 0.957 |
| XG | **0.986** | 0.957 | 0.986 |

Para tener un mejor panorama del mejor algoritmo de clasificación se analizará el parámetro de *BER*. La *Tabla 10* muestra el resultado del parámetro *BER* usando las tres técnicas de selección, podemos ver que los resultados más destacados se encuentran usando la técnica de selección hacia adelante con los algoritmos de KNN y XG logrando un 0.019 de BER al momento de clasificar los eventos LP.

**Tabla 10**

*Comparación de resultados de las técnicas del método de envoltura en BER*

| Clasificador | Selección hacia adelante[%] | Eliminación hacia atrás[%] | Eliminación bidireccional[%] |
|---|---|---|---|
| DT | 0.089 | 0.155 | 0.07 |
| RF | 0.052 | 0.033 | 0.037 |
| KNN | **0.019** | 0.066 | 0.052 |
| SVM | 0.044 | 0.066 | 0.048 |
| NB | 0.044 | 0.1 | 0.052 |
| XG | **0.019** | 0.045 | 0.019 |

La *Tabla 11* muestra los resultados del parámetro *A* usando las tres técnicas de selección de características, aquí se puede observar que nuevamente la técnica destacada es la de selección hacia adelante, los algoritmos KNN y XG logran un 98% de *A* siendo estos los mejores resultados logrados.

**Tabla 11**

*Comparación de resultados de las técnicas del método de envoltura en A*

| Clasificador | Selección hacia adelante[%] | Eliminación hacia atrás[%] | Eliminación bidireccional[%] |
|---|---|---|---|
| DT | 92 | 84 | 93 |
| RF | 95 | 97 | 97 |
| KNN | **98** | 93 | 95 |
| SVM | 95 | 93 | 95 |

| | | | |
|---|---|---|---|
| NB | 95 | 90 | 95 |
| XG | **98** | 95 | 98 |

Al terminar de analizar los parámetros más importantes *A, f1-score,* y *BER* se podría concluir que los mejores clasificadores son KNN y XG usando la técnica de selección hacia adelante debido a que presentaron los mejores resultados. Pero para realizar pruebas parecidas a la realidad se usará la base de datos completa con los 1187 eventos incluyendo los descartados anteriormente HB, RE, ICE para ver el comportamiento de los modelos. La nueva prueba dará otro punto de vista del desempeño de los modelos y nos permitirá tener un panorama más claro para escoger el mejor algoritmo de clasificación para ESV.

### 4.6.1. Pruebas con Base de datos completa

Al tener los modelos de clasificación ya generados se los puede usar para clasificar nuevos ESV y ver su desempeño. EL IGEPN al trabajar 24/7 detectando y clasificando ESV requiere modelos de clasificación que funcionen con miles de ESV en pocos segundos, la mejor forma de replicar esto es usando la base de datos inicial que consta de 1187 ESV, 1044 LP, 101 VT, 8 HB, 27 TRE y 7 ICEQUAKE.

Para esta prueba de clasificación se usaran los modelos generados con las tres técnicas del método de envoltura en el programa final escrito en JN que va desde el Segmento de código 18 hasta el Segmento de código 21. Este código se vuelve a ejecutar para las tres técnicas y usando todos los módulos creados de los algoritmos de clasificación, pero esta vez los datos de prueba será la base de datos completa como se explicó anteriormente. Primero se ejecutara el código para la técnica de selección hacia adelante, obteniendo los resultados de la Tabla 12. Segundo se repite el código para la técnica de eliminación hacia atrás, sus resultados se muestran en la

Tabla ***13***. Por último se vuelve a ejecutar el código para la técnica de eliminación bidireccional, los resultados de esta técnica se muestran en la Tabla 14.

**Tabla 12**

*Parámetros de rendimiento, técnica de selección hacia adelante*

| Clasificador | A[%] | S[%] | R[%] | P[%] | F1-score | BER |
|---|---|---|---|---|---|---|
| DT | 85 | 99 | 84 | 100 | 0.912 | 0.086 |
| **RF** | **89** | **99** | **88** | **100** | **0.937** | **0.064** |
| KNN | 90 | 96 | 89 | 100 | 0.942 | 0.073 |
| SVM | 86 | 99 | 85 | 100 | 0.92 | 0.079 |
| NB | 77 | 100 | 75 | 100 | 0.856 | 0.125 |
| XG | 88 | 100 | 87 | 100 | 0.931 | 0.065 |

**Tabla 13**

*Parámetros de rendimiento, técnica de eliminación hacia atrás*

| Clasificador | A[%] | S[%] | R[%] | P[%] | F1-score | BER |
|---|---|---|---|---|---|---|
| DT | 75 | 89 | 73 | 99 | 0.701 | 0.14 |
| RF | 64 | 100 | 61 | 100 | 0.756 | 0.196 |
| KNN | 64 | 400 | 61 | 100 | 0.757 | 0.195 |
| SVM | 68 | 100 | 65 | 100 | 0.786 | 0.176 |
| **NB** | **70** | **98** | **67** | **100** | **0.804** | **0.173** |
| XG | 63 | 99 | 60 | 100 | 0.749 | 0.206 |

**Tabla 14**

*Parámetros de rendimiento, técnica de eliminación bidireccional*

| Clasificador | A[%] | S[%] | R[%] | P[%] | F1-score | BER |
|---|---|---|---|---|---|---|
| DT | 85 | 100 | 84 | 100 | 0.911 | 0.082 |
| RF | 89 | 97 | 88 | 100 | 0.933 | 0.076 |
| KNN | 85 | 96 | 84 | 100 | 0.908 | 0.103 |
| SVM | 85 | 99 | 84 | 100 | 0.913 | 0.085 |
| NB | 81 | 100 | 79 | 100 | 0.883 | 0.104 |
| **XG** | **86** | **100** | **85** | **100** | **0.92** | **0.074** |

Al revisar los datos lo primero que se observa es que todos los algoritmos en las tres técnicas obtienen 100% en *P,* cabe recordar que se colocan aproximaciones por lo tanto cada clasificador a obtenido por encima del 99.5%, esto se debe a la cantidad de eventos LP que posee la base de datos original es 1044 de 1187 ESV, y la formula de *P* calcula el número total de LP clasificados correctamente, por lo que fallar al clasificar de 1-5 eventos LP da como resultado la aproximación de 100% en *P.* Para analizar los resultados de *A* y *BER* se generan dos tablas adicionales comparando las tres técnicas de selección de características, la Tabla 15 muestra los resultados de *A* y la Tabla 16 muestra los resultados de *BER.*

**Tabla 15**

*Comparación de resultados usando en el parámetro A*

| Clasificador | Selección hacia adelante[%] | Eliminación hacia atrás[%] | Eliminación bidireccional[%] |
|---|---|---|---|

| | | | |
|---|---|---|---|
| DT | 85 | 75 | 85 |
| **RF** | **89** | 64 | 89 |
| KNN | 89 | 64 | 85 |
| SVM | 86 | 68 | 85 |
| NB | 77 | 70 | 81 |
| XG | 88 | 63 | 86 |

**Tabla 16**

*Comparación de resultados en el parámetro BER*

| **Clasificador** | **Selección hacia adelante[%]** | **Eliminación hacia atrás[%]** | **Eliminación bidireccional** |
|---|---|---|---|
| DT | 0.086 | 0.14 | 0.082 |
| **RF** | **0.064** | 0.196 | 0.076 |
| KNN | 0.073 | 0.195 | 0.103 |
| SVM | 0.079 | 0.176 | 0.085 |
| NB | 0.125 | 0.173 | 0.104 |
| XG | 0.065 | 0.206 | 0.074 |

Estos resultados muestran una perspectiva distinta de algunos modelos de clasificación, KNN que fue el mejor algoritmo clasificando la base de datos de prueba y aquí decae su rendimiento al clasificar la base de datos completa. KNN lograba un BER del 0.019 con los datos de prueba, pero al analizar a base de datos completa su BER aumento a 0.074 demostrando que su desempeño ante datos desconocidos como son los eventos HB, TRE, y ICEQUEAKE junto con bases de datos más grandes no es tan efectivo. El

algoritmo de RF por otro lado demostró ser el mejor algoritmo al clasificar la base de datos completa, su *BER* fue el más bajo con 0.064 y con un 89% en *A* con la técnica de selección hacia adelante en la etapa de selección de características.

Otro punto a mencionar es que de antemano se sabe que los modelos generados solo poseen dos posibles salidas eventos LP o VT, por lo que los eventos HB, TRE, y ICEQUAKE serán clasificados como LP o VT. Esto puede será de ayuda para ver como los modelos interpretan eventos desconocidos y como los clasifican, ya que este fenómeno es común en el IGEPN donde sus modelos presentan dificultad al clasificar eventos HB y TRE. Los resultados en *P* al ser de 100% en todos los clasificadores muestran que todos los eventos desconocidos por el modelo los clasifica como eventos VT, esto genera una idea clara del comportamiento de los algoritmos de clasificación. Los algoritmos de clasifican a todos los eventos desconocidos como VT, con este comportamiento de los algoritmos se podría usar los eventos desconocidos por los modelos HB, TRE e ICEQUEAKE y los eventos VT para entrenar nuevos modelos sin los eventos LP para ver si se logra clasificar correctamente los eventos HB, TRE e ICEQUEAKE.

# CAPITULO V

## 5. Discusión

En el siguiente capítulo se discutirá sobre el programa clasificador de ESV usando ML, el funcionamiento del programa, se explicará las etapas de desarrollo del código, y como se muestran los resultados. Se analizará el desempeño de los modelos de clasificación generados y se identificará al que mejor desempeño muestra, además se compara los resultados del presente trabajo de investigación con otras investigaciones sobre la clasificación de ESV del volcán Cotopaxi. Se redactan las conclusiones sobre la investigación y se redacta una lista de posibles trabajos futuros en base al presente proyecto de investigación.

El presente trabajo de investigación se realizó un programa el cual usa seis algoritmos de ML distintos para la clasificación de ESV, los algoritmos usados son DT, RF, KNN, SVM, NB y XG. El primer paso en el desarrollo de la investigación fue analizar la base de datos inicial con la que se trabajó, dicha base es entregada por el IGEPN y consta de 1187 ESV, 1044 eventos LP, 101 eventos VT, 8 eventos HB, 27 eventos RE y 7 eventos ICE, estos ESV poseen 84 características cada uno y se encuentran etiquetados, por lo que se usan los algoritmos DT, RF, KNN, SVM, NB y XG. El proceso general que los algoritmos de ML siguen es la localizan de patrones en los datos proporcionados como entrenamiento, para poder realizar la clasificación. Los algoritmos analizan uno por uno los datos para encontrar similitudes que permitan al modelo aprender de estos patrones. (Smola & Vishwanathan, 2008).

La retroalimentación está presente en todo el desarrollo del código, usando la base de datos inicial el primer paso es la división de los datos en dos grupos de entrenamiento y de prueba, para lo cual se usa *cross validation* del 70/30, en otras palabras, de los datos se usa el 70% para datos de entrenamiento y el 30% para prueba, esta proporción en *cross validation* se usará durante todos los modelos. Generando seis

modelos con algoritmos distintos, con los que se clasifican los datos de prueba y se obtiene el resultado de las clasificaciones, analizando el desempeño de estos resultados con los parámetros de rendimiento se logra deducir la necesidad del balanceo de la base de datos inicial, ya que de los seis modelos ninguno lograba clasificar eventos VT.

Una vez que se tiene la base de datos se prosigue a la etapa del preprocesamiento que consta de tres subetapas, el balanceo de la base de datos, la normalización de los datos y la selección de características. Primero se balancea la base de datos del IGEPN para trabajar con una base de datos modificada de 204 eventos de los cuales 103 son LP y 101 son VT. Se prosigue con el entrenamiento de nuevos modelos de clasificación, para evaluarlos con los datos de prueba obteniendo nuevas medidas de rendimiento, estas medidas mejoran notablemente solo con balancear los datos, pero los resultados siguen siendo desalentadores al momento de la clasificación, como se muestra en la *Tabla 4*. La siguiente subetapa es la normalización de los datos, los valores de las características de los ESV varían ampliamente, van desde un rango de 0.0001 hasta valores mayores de 300, la solución a esto es la normalización, cuyo fin es obtener un conjunto de datos más simples, estables y pequeños a partir de la transformación de datos complejos. (MySQL, 2003). La normalización se aplica a todos los datos de la base de datos balanceada, dando como resultado nuevos datos de entrenamiento y de prueba que van en un rango de 0 a 1. Se usa estos datos de entrenamiento para generar seis nuevos modelos, los cuales se usan para clasificar los datos de prueba, una vez clasificados estos datos se analiza su desempeño con los parámetros de rendimiento, que se muestran en la *Tabla 5*, estos resultados son favorables y se percibe una mejora notable. Por último, algo muy útil para optimizar los modelos en ML es la selección de características, esto se basa en encontrar las características más relevantes de la base de datos modificada, ayudando a que el entrenamiento del algoritmo sea más veloz, incrementa las clasificaciones correctas, reduce el sobreentrenamiento y reduce la complejidad del modelo. (Sayak, 2020). El método usado es el de

envoltura el cual posee tres técnicas, en este trabajo de investigación se utilizan las tres técnicas. Obteniendo las características de cada técnica se entrenan modelos separados, para comparar los resultados de los parámetros de rendimiento entre las tres técnicas como se ve en la *Tabla 11*, siendo la técnica de selección para adelante la que mejores resultados presenta. La *Tabla 17* muestra los resultados de como es el proceso de cada etapa del preprocesamiento en el parámetro de *P*, se nota como los modelos van mejorando al momento de clasificar cada que se pasa de una etapa a otra. Como nota el algoritmo NB no trabaja con valores negativos por lo al generar un modelo con la base de datos sin normalizar (que incluye valores negativos) no se podrá realizar ninguna clasificación.

**Tabla 17**

*Resultados entre de cada etapa analizando el parámetro P.*

| Clasificador | Base de datos Balanceada [%] | Base de datos normalizada[%] | Selección de características[%] |
|---|---|---|---|
| DT | 90.29 | 82.05 | 88.24 |
| RF | 94.29 | 94.29 | 94.29 |
| KNN | 42.11 | 96.30 | 97.14 |
| SVM | 90.05 | 96.97 | 100 |
| NB | - | 96.77 | 100 |
| XG | 96.15 | 92.59 | 97.14 |

Se junta todas las mejoras hechas al clasificador en una sola etapa denominada de preprocesamiento que consta de tres sub etapas, la primera es el balanceo de los datos, continuando con la normalización y por último la selección de características. De esta forma el código queda como muestra la Figura 15, para ser utilizado en la clasificación de ESV, solo se requieren los datos de nuevos ESV y el programa se encarga

de la normalización y la selección de características, de este modo el usuario solo escoge el algoritmo de clasificación que prefiera y el programa devuelve los eventos clasificados, se realizó una prueba siguiendo este proceso en la sección 4.6.1 Pruebas con Base de datos completa.

Las pruebas realizadas a los algoritmos de clasificación con la base de datos completa muestran el desenvolvimiento cuasi real de los modelos de clasificación, debido a que clasifican 1188 ESV los resultados se muestran en la Tabla 12,

Tabla ***13*** y Tabla 14, la robustez de algunos modelos se hacen presente como en los algoritmos RF, SVM, y XG que presentaron los mejores parámetros de rendimiento y sus resultados al clasificar eventos LP fueron casi perfectos logrando clasificar en promedio 1040 de 1044 eventos. Lo que baja el porcentaje de los parámetros de rendimiento son las clasificaciones de los eventos VT, debido a que los modelos confunden los eventos HB, TRE, ICEQUAKE como eventos VT.

### 5.1. Comparación con trabajos similares

El trabajo de investigación realizado se comparara con el algoritmo propuesto en (Lara, Carrera, Benítez, Ruiz, & Rojo, 2016) titulado como *Automatic Recognition of Long Period Events From Vulcano Tectonic Earthquakes at Cotopaxi Vulcano*. En esta investigación se realiza un algoritmo que detecta y clasifica ESV, para la etapa de clasificación se realiza un balanceo de datos y se usaron dos tipos de técnicas de selección de características, el método de envoltura con la técnica de extracción recursiva de características y el método embebido con las técnicas de validación cruzada y podamiento, cada técnica se utilizó dentro de un algoritmo de clasificación adecuado y apropiado, ya sea SVM o DT. Obteniendo los mejores resultados con el método de envoltura con la técnica de extracción recursiva con el algoritmo de SVM y el vector con las características seleccionadas es el siguiente, 'f11', 'f13','f20', 'f21', 'f24', 'f27', 'f39', 'f41', 'f53', 'f56', 'f59', 'f68', 'f76', 'f77', 'f80'. El algoritmo clasificador presentado en (Lara, Carrera, Benítez, Ruiz, & Rojo, 2016) fue evaluado con una base de datos entregada por el IGEPN y que posee ciertos ESV iguales a la base de datos inicial que se usó en esta investigación. La *Tabla 18* muestra las

medidas de rendimiento obtenidas en la investigación de (Lara, Carrera, Benítez, Ruiz, & Rojo, 2016) comparándolos con los modelos de RF y SVM generados en el presente trabajo de investigación.

**Tabla 18**

*Comparación de resultados de algoritmos RF y SVM con un trabajo previo.*

| Medida de Rendimiento | SVM con 15 características [%] | RF con 30 características [%] | SVM con 30 características [%] |
|---|---|---|---|
| **A** | 97 | 96 | 95 |
| **S** | 98 | 95 | 100 |
| **R** | 96 | 98 | 91 |
| **P** | 97 | 96 | 100 |
| **BER** | 0.03 | 0.035 | 0.044 |

Se compara con los resultados del algoritmo que se realizó en este trabajo de investigación con el clasificador RF y SVM. Se usan dos algoritmos para comparar los resultados ya que RF presento los mejores resultados de los seis algoritmos de clasificación usados y SVM debido a que es el algoritmo usado en el trabajo de investigación de (Lara, Carrera, Benítez, Ruiz, & Rojo, 2016). En la *Tabla 18* se muestra una comparación de los parámetros de rendimiento, con el algoritmo de RF los parámetros son muy similares y la diferencia en el BER es de 0.0005. Por otro lado, el modelo usando el algoritmo SVM presentado en este trabajo de investigación muestra un mejor rendimiento en *P* y *S,* pero decae en *A, R* y *BER*. Se puede asumir que gracias a las etapas de preprocesamiento y a las técnicas usadas el algoritmo desarrollado en esta investigación trabaja de manera adecuada.

## 5.2. Conclusiones

Se logró desarrollar un sistema de clasificación de ESV utilizando seis algoritmos de clasificación los cuales son DT, RF, SVM, NB, KNN y XG en el software libre Python. De los seis algoritmos los que mejores resultados al clasificar ESV se obtuvieron de KNN y Utilizando el algoritmo de KNN y XG se obtiene un 98.36% en *A* y un 0.986 en *f1-score*.

Existen diversos tipos de algoritmos de ML con los que se puede trabajar para la clasificación de ESV, el estudio del arte realizado en este tema permitió identificar cuáles son los algoritmos más usados en la actualidad y basándose en estas investigaciones se seleccionó los algoritmos DT, RF, SVM, NB, KNN y XG que muestran el mejor desempeño al momento de clasificar ESV. De esta forma se evitó usar algoritmos de clasificación que disminuyeran los parámetros de rendimiento.

Trabajar con un software libre como Python ayudo mucho al desarrollo del trabajo de investigación, se utiliza la distribución de Anaconda que posee varias librerías y aplicaciones referentes a DS y ML. Las librerías NumPy y Pandas, son ideales para el procesamiento científico y para la manipulación de datos, de igual forma la librería *scikit-learn* es la más completa para temas de ML, de aquí se usaron varios módulos de algoritmos de clasificación los cuales permitieron generar los seis clasificadores del presente trabajo de investigación.

La etapa de preprocesamiento es la que permiten al presente trabajo de investigación lograr clasificaciones correctas dando como resultados en cada parámetro de rendimiento un porcentaje mayor al 95%. El balanceo de datos elimina que los modelos favorezcan solo a la clase mayoritaria, en este caso los eventos LP. La normalización de datos evita la redundancia, anomalías y rangos numéricos muy variables, esto permite trabajar con una base de datos limpia, simple y de tamaño reducido.

Uno de los mejores métodos para optimizar el desempeño de un modelo de clasificación es la selección de características. En esta etapa se logró mejorar los parámetros de rendimiento en un 3%

usando cualquier algoritmo de clasificación. El método de envoltura con la técnica de selección hacia adelante es la que mejor resultados entrego, seleccionando 30 de 84 características que fueron las más relevantes, reduciendo de esta forma la complejidad de los modelos y el sobreentrenamiento. El vector con las características finales son 'f1', 'f2','f3', 'f4', 'f5', 'f6', 'f7', 'f8', 'f9', 'f10', 'f11', 'f12', 'f13', 'f15', 'f18', 'f19', 'f21','f26', 'f28', 'f30', 'f34', 'f38', 'f39', 'f44', 'f51', 'f59', 'f71','f72', 'f73','f82'.

La etapa de preprocesamiento con sus tres optimizaciones, balanceo, normalización y selección de características de la base de datos inicial permiten obtener el mejor desempeño en los parámetros de rendimiento del presente trabajo de investigación. Al analizar el desempeño de los modelos de clasificación usando los datos de prueba obtenidos del *cross-validation* los resultados que sobresalen son un 100% en *P* con SVM y NB, un 100% en *R* con NB y XG, un 98.36% en *A* con KNN y XG, un 100% en *S* con SVM y NB y un 0.986 en *F1-score* en KNN y XG.

Para obtener el desempeño real de los modelos se aplicó una etapa adicional de prueba, se usó la base de datos completa para que los algoritmos clasifiquen los 1187 eventos. Los resultados de la los modelos de clasificación se muestran en la Tabla 15 y la Tabla 16 que analizan los parámetros *A* y *BER* respectivamente, el algoritmo que resalta en ambas tablas es RF obteniendo el mejor desempeño en todos los parámetros de rendimiento con 89% en *A* y 0.064 en *BER.* El algoritmo RF muestra ser el más robusto y que mejores clasificaciones logra.

Los resultados obtenidos tras clasificar la base de datos completa tienen porcentajes bajos de 90% en las medidas de rendimiento por que se clasifican eventos HB, TRE e ICEQUAKE. Los modelos al ser de clasificación binaria por defecto al detectar nuevos tipos de eventos incrementaran el error en las clasificaciones.

Al incluir nuevos eventos en las pruebas realizadas a los modelos de clasificación se busca ver su comportamiento ante datos nunca antes vistos y que no pertenezcan a las clases con las que se

entrenaron, en este caso LP y VT. Los mejores algoritmos RF, SVM y XG presentan un comportamiento similar al clasificar eventos desconocidos, los tres algoritmos clasifican los eventos HB, TRE e ICEQUAKE como VT.

La clasificación de eventos LP con los algoritmos RF, XG y SVM logran un 100% *P*, y un *BER* por debajo del 0.065 demuestra casi una perfección del modelo al clasificar este evento, lo cual es de suma importancia debido a que en la naturaleza los eventos que más produce un volcán son LP, y tener un modelo que los clasifique casi sin errores es fundamental.

Los eventos LP y VT han demostrado ser elementos claves para el Monitorización de un volcán activo como el Cotopaxi ya que muestran información importante sobre su estado. La correcta clasificación de estos eventos es vital, en esta investigación se trabajó con seis algoritmos de clasificación distintos, siendo RF el más sobresaliente, este algoritmo logro 89% en *A* y 0.937 *f1-score* y 0.064 *BER.* Estos parámetros son los más relevantes ya que se logra clasificar casi sin fallos los eventos LP como lo muestra *f1-score* pero sin dejar de lado los eventos VT como muestra *A*.

La base de datos entregada por el IGEPN contiene en su mayoría eventos LP, como esta investigación estaba interesada en su clasificación se seleccionaron solo 100 de los 900 eventos para entrenar a los modelos, sin embargo, para optimizar el entrenamiento de los algoritmos se requiere una base de datos con más eventos VT. Se concluye que se podría mejorar los parámetros de rendimiento como el 89% en *A* usando RF si se tuviera una base de datos con más eventos VT y así poder clasificar de mejor manera ambos eventos y no centrarnos solo en los eventos LP. Otra forma de aumentar los parámetros de rendimiento seria medir el desempeño de los clasificadores con una base de datos más extensa con solo eventos LP y VT sin tomar en cuenta a los eventos HB, TRE e ICEQUEAKE.

La presente investigación presenta resultados altos en los parámetros de rendimiento, como 98.36% en *A* usando el algoritmo KNN, todo esto realizado con librerías, módulos, y algoritmos basados

en código abierto en un software libre Python. En comparación a una investigación previa de (Lara, Carrera, Benítez, Ruiz, & Rojo, 2016) donde logran un 97% en *A* al clasificar los eventos LP y VT usando la misma base de datos pero en un software con licencia llamado MatLab®. El código del presente trabajo de investigación al ser escrito en Python permite libertad de ejecutar, distribuir, modificar y mejorar el software, estas ventajas permiten concluir que el uso de un software libre debe ser promovido para este tipo de investigaciones.

### 5.3. Trabajos Futuros

El presente trabajo de investigación ha sido implementado con Python 3 en su distribución Anaconda utilizando seis algoritmos de clasificación, DT, RF, SVM, Nb, KNN y XG, se puede mejorar el sistema al aplicar nuevos algoritmos, combinaciones de algoritmos o al usar mejoras matemáticas a los seis algoritmos usados de ML.

Un nuevo algoritmo que se podría implementar como trabajo futuro es el algoritmo conocido como Aprendizaje de maquina Extremo (ELM, del inglés *Extreme learning machine*) ha sido uno de los algoritmos que mejor rendimiento presenta en el campo de clasificación. La investigación de (Huang, Ding, & Zhou, 2010) compara su uso con SVM, el algoritmo ELM requiere un bajo costo computacional para el entrenamiento del clasificador, debido a que los nodos ocultos y exteriores son escogidos aleatoriamente y se analizan por separado. La investigación prueba que la teoría es correcta con resultados del uso de ambos clasificadores, ELM para clasificación tiende a lograr un mejor rendimiento que SVM, ELM para la clasificación es menos sensible a los parámetros especificados por el usuario y se puede implementar fácilmente.

El presente trabajo de investigación mostro los mejores parámetros de rendimiento usando el algoritmo KNN por lo tanto se podrían buscar optimizaciones o combinaciones que mejoren más la clasificación de ESV. En (Góra & Wojna, 2012) se combinan los algoritmos de KNN, Inducción de reglas

(del inglés *rule induction*) y aprendizaje basado en instancias (del inglés Instance Based learning), la clasificación se realiza sin usar todo el conjunto de datos de entrenamiento, por el contrario, se selecciona solo una vecindad de datos. El tamaño óptimo de la nueva base de datos se calcula automáticamente en el entrenamiento. La combinación de un algoritmo basado en reglas y KNN acelera de forma significativa el algoritmo utilizando todas las reglas mínimas. El clasificador presentado tiene una alta *P* en comparación al uso individual de clasificadores KNN y más adecuado para clasificadores basados en reglas.

Una optimización del algoritmo KNN se presenta en (Yuan, Zeng-Hui, Hong-Xia, & Chun-Ru, 2014) muestran una versión mejorada de KNN seleccionando mejores centroides iniciales con los que comienza el algoritmo. Primero se analiza las distancias entre cada par de puntos de datos, a continuación, se averigua los datos que presenten características parecidas y finalmente se construye centroides iniciales basados en los puntos de datos encontrados. Si se varia los centroides iniciales variaran los resultados. Si se encuentra centroides iniciales que sean consistentes con la distribución de datos, se puede obtener una mejor agrupación. Los resultados experimentales muestran que el algoritmo de agrupamiento de KNN mejorado tiene una precisión más alta que la original. Este método podría ser implementado en el trabajo de investigación para optimizar aún más la clasificación de ESV.

De igual forma para un trabajo futuro se puede analizar más métodos de selección de características y realizar un análisis comparativo de ellas, dos métodos conocidos son el método de filtro o el método embebido de los que se habló en este trabajo de investigación, pero no se implementaron. Incluso se puede implementar métodos integrados, estos se basan en sacar las cualidades de otros métodos y combinarlos en uno solo, como la combinación del método de filtro y envoltura.

Los resultados obtenidos al probar los modelos con la base de datos completa abre las puertas a un trabajo futuro muy interesante debido a la efectividad de los modelos generados al clasificar eventos LP y asociar todos los eventos desconocidos a los eventos VT. Con la agrupación de todos los eventos a la

clase VT se pueden generar nuevos modelos de clasificación con los datos de entrenamiento solo de eventos VT, HB, RE, TRE, y ICEQUAKE, de esta forma se descartarían los eventos ya clasificados LP y se centraría en el resto de eventos, pero para lo cual se requiere una base de datos mas amplia con de los eventos VT, HB, RE, TRE, y ICEQUAKE.

Aplicar otras ramas de la inteligencia artificial como Deep Learning para la clasificación de ESV abre una gran puerta a un nuevo campo de investigaciones. Estudios de Deep Learning muestran resultados positivos al clasificar grandes cantidades de datos, pero para poder hacer uso de esta rama de la ciencia se requiere ampliar de manera significativa la base de datos de los ESV.

Un trabajo futuro muy interesante seria la clasificación de ESV de algunos volcanes activos del Ecuador como es el Tungurahua, se puede utilizar el trabajo desarrollado en esta investigación, pero con datos del Tungurahua o de otro volcán activo del Ecuador y ver el desempeño de los modelos.

Finalmente, para futuros trabajos en base al volcán Cotopaxi lo más recomendable y necesario es ampliar la base de datos existente, al momento no se posee suficientes ESV lo que siempre generara bases de datos no balanceadas, se requieren más eventos VT como mínimo para que la generación de los modelos mejore su rendimiento. De igual forma se necesitan más eventos de HB, ICEQUAKE, RE para lograr sistemas de detección y clasificación completos, solo de esta forma se logrará tener sistemas confiables para alertar sobre posibles erupciones o cambios de comportamiento repentinos del volcán Cotopaxi.

**BIBLIOGRAFÍA**


Ahmed, R. (2019). Machine Learning Bootcamp with Python. *Udemy.*

Aki, K., & Chouet, B. (1975). Origin of coda waves: Source, attenuation, and scattering effects. *Journal of Geophysical Research, 80*(23), 1896-1977. doi:https://doi.org/10.1029/JB080i023p03322

Alvarez, M., Henao, R., & Duque, E. (Agosto de 2007). Clasificación de eventos sísmicos empleando procesos Gaussianos. (U. T. Pereira, Ed.) *Scientia et Technica Año XIII, 35*( 0122-1701 ). Recuperado el 9 de Noviembre de 2019

Anaconda. (2015). *Anaconda Distribution*. Obtenido de https://www.anaconda.com/distribution/

Asiri, S. (2018). *Machine Learning Classifiers*. Obtenido de towardsdatascience: https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623

Astronoo. (2012). *Volcanes de Ecuador*. Obtenido de http://www.astronoo.com/es/articulos/volcanes-ecuador.html

Bautista, M., & Lara, R. (2018). *Mejora del sistema de reconocimiento de microsismos originado en el volcán Cotopaxi basado en técnicas de Machine Learning.*

Bekara, M., & Van Der Baan, M. (2007). Local Singular Value Decomposition for Signal Enhancement of Seismic Data. *Geophysics, 72*(2), 59-65.

Beniot, J., & Mcnutt, R. (1996). Global volcanic earthquake swarm database and preliminary analysis of volcanic earthquake swarm duration. *Annali di Geofisica, XXXIX*(2), 221-230.

Benitez, D., Paillacho, V., Rojo, J., & Lara, R. (2017). On the use of Multi-class Support Vector Machines for Classification of Seismic Signals at Cotopaxi Vulcano. *IEEE International Autumn Meeting on Power, Electronics and Computing - ROPEC*.

Blong, J. R. (1984). *Volcanic Hazards.* Estados Unidos: Academic Press.

Brownlee, J. (2019). *Rescaling Data for Machine Learning in Python with Scikit-Learn*. Obtenido de https://machinelearningmastery.com/rescaling-data-for-machine-learning-in-python-with-scikit-learn/

C. Jiang, H. Z.-C. (2017). Machine Learning Paradigms for Next-Generation Wireless Networks. *IEEE Wireless Communications*.

Cachupuendo, D., Vasquez, F., Toulkeridis, T., & Karl, H. (2017). Evaluación de la comunicación preventiva de las amenazas volcánicas del volcán Cotopaxi en niños de escuelas fiscales y privadas del Valle de los Chillos. *Revista de Ciencias de Seguridad y Defensa, II*(2).

Capó, M., Pérez, A., & Lozano, J. (2017). An efficient approximation to the K-means clustering for massive data. *Elsevier Knowledge-Based Systems*, 56-69.

Caragea, D., Cook, D., & Honavar, V. (2005). *Visual Methods for Examining Support Vector Machine Results, with Applications to Gene Expression Data Analysis.*

Chouet, B. A. (1996). Long-period vulcano seismicity: its source and use in eruption forecasting. *Nature*, 309-316.

Corominas, M. (Noviembre de 2011). ESTUDIO COMPARATIVO DE LOS PLANES DE ACTUACIÓN FRENTE AL RIESGO VOLCÁNICO (CHILE, COSTA RICA, EL SALVADOR, ECUADOR, ESPAÑA, MÉXICO Y NICARAGUA). *Revista Geológica de América Central*, 33-56. doi:: 10.15517/rgac.v0i52.18980

Coruna, U. d. (2017). *Ondas Sismicas*. Obtenido de https://www.udc.es/dep/dtcon/estructuras/ETSAC/Investigacion/Terremotos/ondas_s%EDsmicas.htm

Daniel Andrade, M. H. (2005). Peligros Volcanicos asociados con el Cotopaxi. *Corporacion Editora Nacional*.

Ecuador, R. (03 de 2020). *Sistema de Alerta Temprana*. Obtenido de https://www.gestionderiesgos.gob.ec/category/sat-sistema-de-alerta-temprana/

Ecuavisa. (17 de Julio de 2015). *Cuatro nuevos sismógrafos monitorean al volcán Cotopaxi*. Obtenido de https://www.ecuavisa.com/articulo/noticias/actualidad/114549-cuatro-nuevos-sismografos-monitorean-al-volcan-cotopaxi

EducarChile. (2012). *Centro de recursos digitales*. Obtenido de Ondas sísmicas : http://centroderecursos.educarchile.cl/handle/20.500.12246/52350

Esgozcue, J., & Canet, J. (2000). Revisión de métodos de análisis espectral. Aplicación al estudio y simulación de un registro sísmico. *Revista de obras públicas*, 429-446.

Firoozabadi, A., Seguel, F., Soto, I., Guevara, D., Huenupan, F., Millaray, C., & Franco, L. (2017). Evaluation of Llaima vulcano activities for localization and classification of LP, VT and TR events. *Journal of Electrical Engineering*, 325-338.

Gandhi, R. (2018). *Support Vector Machine — Introduction to Machine Learning Algorithms*. Obtenido de https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

GeoEnciclopedia. (2016). *Erupcion Volcanica.*

Gerkman, T., & Hendriks, R. (2012). Unbiased MMSE-Based Noise Power Estimation With Low Complexity and Low Tracking Delay. *IEEE Trans Audio, Speech, Language Processing, 20*, 1383-1393.

Ghoneim, S. (2019). *Accuracy, Recall, Precision, F-Score & Specificity, which to optimize on?* Obtenido de Towardsdatascience: https://towardsdatascience.com/accuracy-recall-precision-f-score-specificity-which-to-optimize-on-867d3f11124

Góra, G., & Wojna, A. (2012). RIONA: A New Classification System Combining Rule Induction and Instance-Based Learning. *Fundamenta Informaticae*, 369-390.

GoRaymi. (2016). *Volcanes en Ecuador*. Obtenido de https://www.goraymi.com/es-ec/ecuador/volcanes-ecuador-ai81v47yu

Granada, U. d. (2018). *Python y sus principales caracteristicas .* Granada.

Gubbins, D. (2004). The seismic wavefield: Introduction and theoretical development. *Cambridge University Press, 143*(3), 541-547. doi:https://doi.org/10.1016/j.pepi.2004.06.002

Guffanti, M., Brantley, S., Cervelli, P., Nye, C., Serafino, G., Siebert, L., . . . Wald, L. (2007). *Technical-Information Products for a National Vulcano Early Warning System.* U. S Geological Survey. Virginia: U. S. Department of the Interior.

Guy, L., & Mohamed, E.-G. (2018). *Computing with Data: An Introduction to the Data Industry.* Springer.

Hajian, A., Cannavò, F., Greco, F., & Nunnari, G. (2019). CLASSIFICATION OF MOUNT ETNA (ITALY) VOLCANIC ACTIVITY BY MACHINE LEARNING APPROACHES. *ANNALS OF GEOPHYSICS*.

Hall, M., & Mothes, P. (2008). The rhyolitic-adesitic eruptive history of Cotopaxi vulcano, Ecuador. *Springer Bulletin of Vulcanology, 70*, 675-702. doi:https://doi.org/10.1007/s00445-007-0161-2

Hall, M., Andrade, D., Mothes, P., Troncoso, P., Eissen, J.-P., Samaniego, P., . . . Yepes, H. (2005). Los peligros volcánicos asociados con el Cotopaxi. En Institut de Recherche Pour Le D´eveloppement (IRD), & IGEPN. Corporación Editorial Nacional.

Hernandez. (2017). Teoria de la Coda.

Huang, G.-B., Ding, X., & Zhou, H. (2010). Optimization method based extreme learning machine for classification. *Neurocomputing*, 155-163.

Ibáñez, J., & Carmona, E. (1997). *Sismicidad Volcánica.* Granada: Instituto Andaluz de Geofísica. Universidad de Granada.

IGEPN. (2014). *Glosario*. Obtenido de https://www.igepn.edu.ec/glosario

IGEPN. (16 de Agosto de 2016). Recuperado el 2 de 4 de 2018, de Crónicas de la Erupción del Volcán Cotopaxi 2015; http://www.igepn.edu.ec/servicios/noticias/1376-cronicas-de-la-erupcion-del-volcan-cotopaxi-2015

IGEPN. (18 de 09 de 2016). *Aumento de la actividad sísmica del volcán Tungurahua*. Obtenido de https://www.igepn.edu.ec/content/41-volcanes?start=80

IGEPN. (2017). *Cotopaxi.* Obtenido de http://www.igepn.edu.ec/

IGEPN. (06 de 2018). *Cotopaxi Red Monitorización*. Obtenido de https://www.igepn.edu.ec/cotopaxi-red-de-Monitorización/content/9-cotopaxi

IGEPN. (2019). *Clasificacion de Volcanes en el Ecuador*. Obtenido de https://www.igepn.edu.ec/red-de-observatorios-vulcanologicos-rovig

IGEPN. (2019). *Presentación*. Obtenido de Instituto Geofísico de la Escuela Politécnica Nacional: https://www.igepn.edu.ec/nosotros

INPRES, I. d. (2016). *Tipos de erupciones volcanicas.*

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2015). *An Introduction to Statistical Learning.* Springer.

Jaramillo Aranha, C. (2015). *Caracterización de señales sísmicas del volcán Cotopaxi utilizando estimadores espectrales clásicos y de máxima entropía.* Universidad de las Fuerzas Armadas ESPE, Departamento de Eléctrica y Electrónica, Sangolquí.

Jaramillo, C., León, R., Lara, R., Benitez, D., & Ruiz, M. (2014). *Caracterización de Señales Sísmicas del Volcán Cotopaxi Utilizando Estimadores Espectrales clásicos y Máxima Entropía.* Cuenca: Maskana.

Jupyter. (2014). *Jupyter*. Obtenido de jupyter.org

Khandelwal, R. (24 de 10 de 2019). *Feature selection in Python using the Filter method*. Obtenido de https://towardsdatascience.com/feature-selection-in-python-using-filter-method-7ae5cbc4ee05

Kharkovyna, O. (2019). *Medium*. Obtenido de Python vs R. Choosing the Best Tool for AI, ML & Data Science: https://medium.com/datadriveninvestor/python-vs-r-choosing-the-best-tool-for-ai-ml-data-science-7e0c2295e243

Lara, R., Benítez, D., Carrera, E., Ruiz, M., & Rojo, J. (2016). Feature Selection of Seismic Waveforms for Long Period Event Detection at Cotopaxi Vulcano. *Journal of Vulcanology and Geothermal Research*, 34-49.

Lara, R., Carrera, V., Benítez, D., Ruiz, M., & Rojo, J. (2016). Automatic Recognition of Long Period Events From Vulcano Tectonic Earthquakes at Cotopaxi Vulcano. *IEEE Transactions on Geoscience and Remote Sensing*.

Lara, R., Paillacho, V., & Villalva, M. (2016). *Detección de Eventos del Volcán Cotopaxi Empleando Clasificación Supervisada; Revista Iberoamericana de las Ciencias Computacionales e Informática*.

Lara-Cueva, R., Moreno, A., Larco, J., & Benitez, D. (2016). Real-Time Seismic Event Detection Using Voice Activity Detection Techniques. *IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING*, 1939-1404. Obtenido de http://www.ieee.org/publications standards/publications/rights/index.html

Lara-Cueva, R., Paillacho-Salazar, V., & Villalva-Chaluisa, M. (2016). Hacia un sistema de detección automática de señales del volcán Cotopaxi. *DYNA, 84*, 176-184. doi:10.15446/dyna.v84n200.54573

Lara-Cueva, R., Paillacho-Salazar, V., & Villalva-Chaluisa, M. (Marzo de 2017). Towards an automatic detection system of signals at cotopaxi vulcano. *DYNA*, 176-184. doi:http://dx.doi.org/10.15446/dyna.v84n200.54573

Londoño, I. (2011). *Implementación de un Sistema de Monitorización de señales sísmicas del volcán Cotopaxi empleando una red de sensores Inalámbricos.* Sangolquí.

Loyola, E. (2017). *Ondas Sismicas*. Obtenido de https://www.slideshare.net/

Luhaniwal, V. (4 de 10 de 2019). *Feature selection using Wrapper methods in Python*. Obtenido de https://towardsdatascience.com/feature-selection-using-wrapper-methods-in-python-f0d352b346f

Malfante, M., Mura, M. D., Métaxian, J.-P., & Mars, J. (2018). Machine Learning for Vulcano-seismic Signals: Challenges and Perspective. *HAL archives-ouvertes*.

Mario Ruiz, R. L. (2016). Automatic Recognition of Long Period Events From Vulcano Tectonic Earthquakes at Cotopaxi Vulcano. *IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING* .

Medina, A. (2015). El volcán Cotopaxi es monitoreado por una red de 59 estaciones. *El Comercio*.

Meza, I. V. (Marzo de 2013). MFCCs. (UNAM, Ed.) México. Obtenido de https://turing.iimas.unam.mx/~ivanvladimir/posts/mfcc/

Mora , M., & Alvarado , G. E. (2001). Primer taller en actualización de Sismología Volcánica. *Red Sismológica Nacional (UCR-ICE).* Costa Rica: Red Sismológica Nacional. Recuperado el 11 de

Noviembre de 2019, de https://rsn.ucr.ac.cr/documentos/educativos/vulcanologia/5099-que-es-un-tremor

MySQL. (2003). *Normalizacion de bases de datos.* Mallorca.

NationalGeographic. (2015). *Peligro Latente*. Obtenido de https://www.ngenespanol.com/

Notimerica. (05 de 06 de 2018). *Los 10 volcanes más peligrosos de Iberoamérica*. Obtenido de https://www.notimerica.com/sociedad/noticia-10-volcanes-mas-peligrosos-iberoamerica-20180605192152.html

NumFOCUS. (2020). *Numpy*. Obtenido de https://numpy.org/

NumFOCUS. (2020). *pandas*. Obtenido de https://pandas.pydata.org/

Perez, D. (2017). *Evolucion Espacio-Tiempo del parametro b en el volcan Cotopaxi.* Quito.

Perez, N., Benites, D., Grijalva, F., Lara, R., Ruiz, M., & Aguilar, J. (2020). ESeismic: Towards an Ecuadorian vulcano seismic repository. *Journal of Vulcanology and Geothermal Research*, 7.

Portilla, J. (2019). *Python for data science and machine learning .*

Python. (2012). *What is Python?* Obtenido de https://www.python.org/doc/essays/blurb/

Pythones. (2017). *Módulos y Librerías en Python : Importar, acceder, crear*. Obtenido de https://pythones.net/importar-modulos-en-python/

Ray, S. (2017). *6 Easy Steps to Learn Naive Bayes Algorithm with codes in Python and R*. Obtenido de www.analyticsvidhya.com/: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

ray, S. (3 de 5 de 2018). *Improve Your Model Performance using Cross Validation (in Python and R)*. Obtenido de https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r/

Ren, X., Peltier, A., Ferrazzini, V., Rouet-Leduc, B., & Jhonns, A. (2018). Machine Learning Reveals the Seismic Signature of Eruptive Behavior at Piton de la Fournaise Vulcano. *Geophysical Reserch Letters*.

Ríos, M. (2005). *Análisis Espectral Multivariable aplicado a Señales Cerebrales Reales (EGG).* Sevilla: Universidad de Sevilla.

Robledano, A. (2019). *OpenWebinars*. Obtenido de Que es Python.

Rocco, C. M., & Zio, E. (2005). Bootstrap-Based Techniques for Computing Confidence Intervals in Monte Carlo System Reliability Evaluation. *IEEE*, 303-307.

Rong, H.-J., Ong, Y.-S., Tan, A.-H., & ZexuanZhu. (2018). A fast pruned-extreme learning machine for classification problem. *Neurocomputing*, 359-366.

Rosenberg, M. (22 de Diciembre de 2018). *Ring of Fire*. Obtenido de ThoughtCo: https://www.thoughtco.com/ring-of-fire-1433460

Rubin, J., Abreu, R., Ganguli, A., Nelaturi, S., Matei, I., & Sricharan, K. (2016). Classifying heart sound recordings using deep convolutional neural networks and mel-frequency cepstral coefficients. *Computing in Cardiology Conference (CinC)*, (págs. 813-816). Vancouver. Obtenido de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7868867&isnumber=7868653

Salazar, D., & D'Ercole , R. (2009). Percepción del riesgo asociado al volcán Cotopaxi y vulnerabilidad en el Valle de Los Chillos (Ecuador). *Bulletin de l'Institut Français d'Études Andines, 38(3)*, 849-871. doi:10.4000/bifea.2522

Santacruz, A. (2016). *Por qué es importante trabajar con datos balanceados para clasificación*. Obtenido de http://amsantac.co/blog/es/2016/09/20/balanced-image-classification-r-es.html

Santos, P. R. (2018). *Machine Learning a tu alcance: La matriz de confusión*. Obtenido de empresas.blogthinkbig: https://empresas.blogthinkbig.com/ml-a-tu-alcance-matriz-confusion/

Sayak, P. (2 de 1 de 2020). *Beginner's Guide to Feature Selection in Python*. Obtenido de https://www.datacamp.com/community/tutorials/feature-selection-python

ScikitLearn. (2016). *sklearn metrics f1_score*. Obtenido de https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

ScikitLearn. (2018). *sklearn neighbors KNeighborsClassifier*. Obtenido de https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Sieron, K. (2015). *Vulcanismo.* Veracruzana.

Smola, A., & Vishwanathan, S. (2008). *Introduction to Machine Learning.* Cambridge University Press.

Solanas, A., & Sierra, V. (1999). Bootstrap: Fundamentos e Introducción a sus Aplicaciones. *Universidad de Barcelona*(55), 143-154.

Soto, R., Huenupan, F., Meza, P., Millaray, C., & Franco, L. (2018). Spectro-Temporal Features Applied to the Automatic Classification of Volcanic Seismic Events. *Journal of Vulcanology and Geothermal Research*.

Tilling, R., & Beate, B. (1993). Los peligros volcánicos. Apuntes breves sobre un curso breve. *Organización Mundial de Observatorios Vulcanológicos*, 2-3.

Toro, L. (2018). *Anaconda Distribution: La Suite más completa para la Ciencia de datos con Python*. Obtenido de https://blog.desdelinux.net/ciencia-de-datos-con-python/

Toulkeridis, T., & Aguilera, E. (2015). *El Volcán Cotopaxi, una amenaza que acecha.* Sotavento Ed.

Tutorialspoint. (2016). *ML - Support Vector Machine(SVM)*. Obtenido de https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_support_vector_machine.htm

UNESCO. (1998). EVALUACIÓN y PREDICCIÓN DE LA PELIGROSIDAD.

Universidad de Costa Rica. (31 de 05 de 2011). *Laboratorio de Ingeniería Sísmica del Instituto de Investigaciones en Ingeniería (INII).* Recuperado el 05 de 02 de 2018, de http://www.lis.ucr.ac.cr/7

Valeria Paillacho, R. L. (2016). Hacia un sistema de detección automática de señales del volcán Cotopaxi. *DYNA* .

Vallejo Vargas, S. (Febrero de 2011). Distribución de cenizas volcánicas holocénicas - tardias en la costa del Ecuador. Quito, Pichincha, Ecuador: Escuela Politécnica Nacional.

Viracucha , E., & De la Bastida, J. (2014). Sistema Informático para el Procesamiento y Análisis de Señales Sísmicas de Volcanes en Ecuador. *Revista EPN, 33*.

Wassermann, J. (2012). Vulcano Seismology.

Wei, Q., & Dunbrack, R. (2013). The Role of Balanced Training and Testing Data Sets for Binary Classifiers in Bioinformatics.

Witsil, A., & Jhonnson, J. (2019). Vulcano video data characterized and classified through computer vision and machine learning algorithms. *Reserch Gate*.

Yuan, F., Zeng-Hui, M., Hong-Xia, Z., & Chun-Ru, D. (2014). A new algorithm to get the initial centroids. *IEEE*.

Zelada, C. (2017). *Evaluación de modelos de clasificación*. Obtenido de Rpubs: https://rpubs.com/chzelada/275494

Zou, K. H., O'Malley, J., & MauriMD, L. (2007). Receiver-Operating Characteristic Analysis for Evaluating Diagnostic Tests and Predictive Models. (H. M. School, Ed.) *Harvard Clinical Research Institute*, 654-657. doi:https://doi.org/10.1161/CIRCULATIONAHA.105.594929

# ANEXOS