FACULTAD DE INGENIERÍA

TESIS PREVIA A LA OBTENCIÓN DEL TÍTULO DE MAGISTER EN SISTEMAS DE INFORMACIÓN CON MENCIÓN EN DATA SCIENCE

MINERÍA DE DATOS PARA EL MODELADO DE APRENDIZAJE AUTOMÁTICO EN LA RESISTENCIA DE MATERIALES POR LA INCLUSIÓN DE FIBRAS NATURALES.

Caso de estudio: Influencia de la inclusión de fibra de abacá en la resistencia a la compresión de limos arenosos, Laboratorio de, Mecánica de Suelos, Pavimentos y Geotécnica de la Pontificia Universidad Católica del Ecuador

NOMBRE:

ESCOBAR TERÁN CHARLES ÉDISSON

DIRECTOR:

MELGAREJO RAFAEL

QUITO, 2022

RESUMEN EJECUTIVO

El propósito del proyecto es buscar modelos de aprendizaje automático que permitan predecir la influencia de fibras naturales de abacá en la resistencia a la compresión de limos arenosos. Para el efecto, se aplicará el ciclo de vida de CRISP-DM[1] a la data registrada en el Laboratorio de Resistencia de Materiales, Mecánica de Suelos, Pavimentos y Geotécnica de la PUCE. De encontrarse modelos de aprendizaje automático para este conjunto de datos, partiendo de casos particulares se podrá generalizar y predecir valores futuros de resistencia de materiales.

Dentro de la metodología CRISP-DM, para seleccionar el modelo de aprendizaje de máquina será necesario experimentar con los datos disponibles y encontrar aquel en los que el error sea menor, de tal manera que la máquina pueda resolver situaciones con datos no conocidos. Los resultados serán evaluados y se seleccionará el que presente la mejor aproximación para entrenar una máquina.

Una vez entrenada, se utilizarán nuevos datos de casos particulares, y se evaluará su despliegue. Si el despliegue no es satisfactorio será necesario argumentar las posibles razones por la que los datos no pudieran converger en un modelo de aprendizaje automático en cada una de las pruebas.

Conforme a como se han planteado los objetivos, las etapas de Comprensión del negocio, Comprensión de datos y Preparación de datos se desarrollan en el acápite 4.1, mientras que el modelado se realiza en el 4.2, quedando la evaluación en el acápite 4.3.

Palabras clave: *Fibra de abacá, aprendizaje automático, aprendizaje de máquina, algoritmos de aprendizaje automático, minería de datos, resistencia de materiales.*

1 CRoss Industry Standard Process for Data Mining (CRISP-DM)

CAPÍTULO 1:

1.1. Dedicatoria

A mi padre, el Gran Jefe, cuya presencia se mantiene presente en nuestra memoria a través de su ejemplo de vida, a mi señora madre, un ángel en la tierra que con amor y paciencia amalgama la familia.

A mis hijos y compañera de vida quienes me acompañan y animan en las distintas aventuras de la vida.

1.2. Agradecimientos

A Dios, por su infinito amor y gracia y a todos quienes han contribuido de una u otra forma a ayudarme a crecer en todos los aspectos.

1.3. Tabla de contenidos

Tabla de Contenidos

Índice de figuras

Índice de tablas

CAPÍTULO 2

2. INTRODUCCIÓN

2.1 Justificación

De los múltiples ensayos con distintos tipos de materiales que realiza el Laboratorio de Resistencia de Materiales, Mecánica de Suelos, Pavimentos y Geotécnica de la PUCE está el de resistencia por la inclusión de fibras naturales autóctonas de la región, entre ellas la fibra de abacá en los limos arenosos. Hay que resaltar que esta data es inédita, levantada por los técnicos del laboratorio, y no ha sido usada previamente para minería de datos ni aprendizaje automático.

Por las características de CRISP-DM[2] es posible analizar y probar la data del mencionado Laboratorio dentro de un modelo de minería de datos y probar diferentes modelos de aprendizaje automático que permitan predecir la influencia de materiales como la fibra de abacá en la resistencia a la compresión de limos arenosos. Tomando en cuenta que los datos están etiquetados, el modelo de aprendizaje probablemente corresponda a aprendizaje automático y será necesario determinar cuáles serían las características de entrada y de salida (objetivo) y si se usan todas ellas.

Dentro del ciclo de vida de CRISP-DM, el entrenamiento partiría de casos particulares para luego generalizar su funcionamiento, de tal manera que permita predecir valores futuros con valores no conocidos por la máquina. A priori se propone utilizar aprendizaje supervisado, considerando que la data es etiquetada y se debe determinar si la salida es de naturaleza continua o discreta.

En la práctica será necesario experimentar con los datos disponibles y encontrar primero si esta data es convergente en alguno de los modelos, considerando el de mejor aproximación y o menor error. Sin embargo dentro del ciclo de vida de minería de datos seleccionado, esto debería demostrarse ya que se corren riesgos de que los modelos no converjan o que la data no sea consistente, así, el objetivo es determinar las razones y factores que impiden el aprendizaje, considerando principalmente que los experimentos con estos materiales se

2 CRoss Industry Standard Process for Data Mining (CRISP-DM)

realizaron en el pasado y no sería fácil repetirlos, implicando materiales y recursos con los que no se cuenta actualmente.

En todo caso y en el peor escenario que esta data inédita no tenga un modelo convergente de aprendizaje automático, si sería posible cumplir el ciclo de vida de CRISP-DM aunque no se pueda determinar por el momento un modelo al que se ajusten los datos.

2.2 Planteamiento del problema

"El Laboratorio de Resistencia de Materiales, Mecánica de Suelos, Pavimentos y Geotécnica de la Pontificia Universidad Católica del Ecuador, funciona desde 1965 y consta en el Registro de Consultoría como Entidad Universitaria Consultora. Las actividades que se desarrollan en este centro, también conocido como Laboratorio de Materiales de Construcción (LMC), están orientadas a la investigación, control de calidad de los materiales y consultoría en proyectos de ingeniería, así como la fiscalización de obras civiles." (PUCE,2022)[3]

Dentro de sus actividades de investigación y cobijado bajo el paraguas del dominio "Hábitat, infraestructura y movilidad" en el desarrollo de la línea de investigación "Diseño, infraestructura y sistemas sociales y ambientales para un hábitat sostenible"[4], se ha enfocado a analizar el impacto en la resistencia de materiales al introducir en su composición elementos autóctonos del Ecuador, como por ejemplo la fibra de abacá en los limos arenosos. Para ello se ha basado en la amplia experiencia y trayectoria del Laboratorio que cuenta tanto con el equipamiento necesario para realizar los diferentes tipos de ensayos como el personal capacitado.

Así, "El LMC[5] cuenta con personal altamente capacitado, experiencia laboral corporativa e infraestructura con tecnología en constante actualización. Como complemento básico y

3 Pontificia Universidad Católica del Ecuador, (2022). Laboratorio de Resistencia de Materiales, Mecánica de Suelos, Pavimentos y Geotécnica https://www.puce.edu.ec/servicios/laboratorio-de-suelos/

4 Dominios Académicos y Líneas de Investigación, PUCE, 2017

5 Laboratorio de Resistencia de Materiales, Mecánica de Suelos, Pavimentos y Geotécnica de la Pontificia Universidad Católica del Ecuador

fundamental de mejora continua, el LMC decidió trabajar bajo el Sistema de Gestión de Calidad basado en la Norma ISO/IEC 17025.” (PUCE,2022)[6]

La investigación desarrollada en el LMC por Albuja J. y otros[7] “determina la influencia de la inclusión de fibras de abacá en la resistencia a la compresión simple de especímenes reconstituidos de limos arenosos. Para ello se añade fibra en diferentes longitudes (5, 10 y 15 mm) y cantidades (0,5, 1,0, 1,5 y 2,0% del peso seco del suelo), evaluando el comportamiento físico y mecánico de las mezclas suelo-fibras mediante ensayos de compactación y compresión no confinada” (Albuja J. Y otros, 2021).

Como resultado de los ensayos, “Las pruebas realizadas mostraron un aumento en el esfuerzo máximo de compresión de hasta 1235.1% (en comparación con el suelo libre de fibra) cuando se incrementa el contenido de fibra, su longitud o ambos. La falla por compresión ocurrió con una deformación axial mayor cuando se agregó fibra de 10 y 15 mm en una dosis del 1% o en porcentajes iguales o superiores al 1,5%, independientemente de la longitud de la fibra. Una serie de modelos mixtos lineales identificaron efectos estadísticamente significativos de la longitud y el porcentaje de fibra sobre el nivel de esfuerzo y sobre la deformación unitaria.” (Albuja J. Y otros, 2021).

Se debe recalcar que los datos con los que se cuenta, obtenidos de las mediciones realizadas en el LMC de la PUCE, con el objeto de medir el efecto de introducir fibra de abacá en los limos arenosos y determinar cuánto afecta a su resistencia constituyen data inédita levantada por los técnicos del laboratorio.

Aun cuando, en comparación con conjuntos de datos usados para ciencias de datos en el área de ingeniería civil[8], la data existente es abundante ya que se cuenta con más de 1600 instancias, esta no ha sido estudiada o analizada para ser utilizada con metodologías de minería de datos como por ejemplo CRISP-DM[9] para aprendizaje automático. Es necesario analizar si la data requiere de una preparación o pre procesamiento para determinar si es

6 Pontificia Universidad Católica del Ecuador, (2022). Laboratorio de Resistencia de Materiales, Mecánica de Suelos, Pavimentos y Geotécnica https://www.puce.edu.ec/servicios/laboratorio-de-suelos/

7 Albuja-Sánchez, J.; Alcívar, E.; Escobar, D.; Montero, J.; Realpe, G.; Muñoz, A.; Peñaherrera-Aguirre, M. Influence of Abaca Fiber Inclusion on the Unconfined Compressive Keywords: simple compression test; unconfined compression test; abaca fiber; soil improvement; fiber-reinforced soil Strength of Reconstituted Sandy Silts. Fibers 2022, 10, 99. https://doi.org/10.3390/fib10110099

8 103 instancias, Yeh, I-Cheng, "Modeling slump flow of concrete using second-order regressions and artificial neural networks," Cement and Concrete Composites, Vol.29, No. 6, 474-480, 2007.

9 CRoss Industry Standard Process for Data Mining (CRISP-DM)

posible aplicar un modelo de aprendizaje automático que permita predecir la influencia de materiales como la fibra de abacá en la resistencia a la compresión de limos arenosos.

Tampoco se ha determinado si la data existente, en caso de que supere las etapas de entendimiento y preparación, puede ser usada para aprendizaje supervisado o no supervisado. Aunque se presume que la data pueda etiquetarse, no se han definido las variables características de entrada y objetivo de salida. Tampoco se ha definido que en caso de que se confirme que es etiquetada si se trata de un modelo de regresión o clasificación.

A priori y en consideración a que se cuenta con datos conocidos de mediciones previas se asume que se cumple con los requisitos para entrenar una máquina de aprendizaje automático supervisado, partiendo de casos particulares para generalizar su funcionamiento, de tal manera que permita predecir valores futuros con valores no conocidos por la máquina. Presuposición que debe demostrarse.

Si bien es cierto se presume que se podría aplicar aprendizaje supervisado para seleccionar el modelo de aprendizaje de máquina, considerando que la data es de naturaleza continua y categórica, con lo que se esperaría que los modelos respondan satisfactoriamente con esta data, sin embargo, en la práctica no se ha experimentado con los datos disponibles y no se conoce los modelos que arrojen el menor error, de tal manera que la máquina pueda resolver situaciones con datos no conocidos.

Como premisa se podría partir que una vez que se haya seleccionado y evaluado el mejor modelo de aprendizaje de máquina y este haya sido entrenado estará en una situación de generalidad con capacidad de resolver nuevos casos particulares con nuevos datos. Sin embargo, esto debe demostrarse para no correr riesgos de que los modelos no converjan o que la data no sea consistente. Si no existe convergencia, hay que procurar determinar lo que pudiese estar ocurriendo,

Problema:

No se ha determinado si es posible usar el modelo CRISP-DM para predecir la influencia de la inclusión de fibras de abacá en la resistencia a la compresión simple de especímenes reconstituidos de limos arenosos a través de aprendizaje automático.

Pregunta:

¿El modelo CRISP-DM puede ser usado para predecir la influencia de la inclusión de fibras de abacá en la resistencia a la compresión simple de especímenes reconstituidos de limos arenosos a través de aprendizaje automático?

2.3 Contextualización del tema u objeto

Contexto general/Objeto de estudio: Modelos de minería de datos y modelos de aprendizaje automático.

Contexto Particular/Campo de acción: Minería de datos y modelos de aprendizaje automático para predecir el impacto de las fibras de abacá en la resistencia a la compresión de limos arenosos.
(Fibra de abacá, aprendizaje de máquina, algoritmos de aprendizaje automático, minería de datos)

2.4 Objetivos

Objetivo General

Determinar si es posible usar el modelo CRISP-DM para predecir la influencia de la inclusión de fibras de abacá en la resistencia a la compresión simple de especímenes reconstituidos de limos arenosos a través de aprendizaje automático usando el conjunto de datos resultado de ensayos del LMC.

Objetivos Específicos

- Aplicar la metodología CRISP-DM para entender y preparar la data resultado de ensayos del LMC para aprendizaje automático.

- Determinar las técnicas de modelado de aprendizaje automático para predecir la deformación por resistencia a la compresión no confinada de limos arenosos reconstituidos debido a la influencia de la inclusión de fibra de abacá.
- Evaluar los modelos que mejor cumplen con los objetivos de la data

2.5 Metodología y Técnicas

Se aplicará el método inductivo con el cual se busca generalizar el funcionamiento de una máquina en base al entrenamiento de la misma, partiendo de valores conocidos de situaciones particulares, y que servirán para entrenarla de tal manera que pueda de manera general predecir el comportamiento de datos desconocidos.

Se partirá de un conjunto de datos de mediciones realizadas en el Laboratorio de Resistencia de Materiales, Mecánica de Suelos, Pavimentos y Geotécnica de la PUCE, el mismo que será analizado de manera cuantitativa y cualitativa para determinar la calidad de la misma y en caso de ser necesaria pre procesarla.

Se aplicará el método científico empírico experimental de minería de datos para determinar un modelo de aprendizaje de máquina que se ajuste a la data, probando varios modelos cuyas particularidades se ajusten a la data, para seleccionar el que mejor se ajuste a ella.

Como técnica instrumental se aplicará la metodología CRoss Industry Standard Process for Data Mining (CRISP-DM).

CAPÍTULO 3

3. MARCO TEÓRICO Y CONCEPTUAL

3.1 CRISP-DM

"Publicado en 1999, CRISP-DM (CRoss Industry Standard Process for Data Mining (CRISP-DM)) es el marco más popular para ejecutar proyectos de ciencia de datos. Proporciona una descripción natural de un ciclo de vida de ciencia de datos (el flujo de trabajo en proyectos centrados en datos). Sin embargo, este enfoque centrado en tareas para ejecutar proyectos no aborda los problemas de equipo y comunicación. Por lo tanto, CRISP-DM debe combinarse con otros marcos de coordinación de equipos." (Data Science Process Alliance, 2021).

En la siguiente figura se puede apreciar el resultado de la aplicación de una encuesta para determinar la preferencia del uso de frameworks en proyectos de ciencia de datos, en la que se puede apreciar una fuerte inclinación al uso de esta metodología.

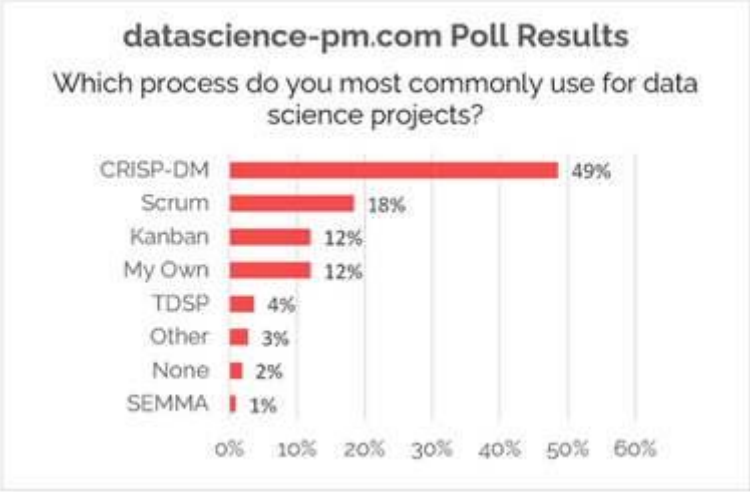


Figura 1: Encuesta de uso para frameworks para ciencia de datos (Data Science Process Alliance, 2021)

Fases de CRISP-DM[10]

10 La descripción y análisis de las fases de CRISP-DM ha sido orientada a definir un marco de trabajo, en el cual al contestar las preguntas se define también un marco de acción que servira de referencia en el desarrollo del producto propuesto. Las preguntas se basan en las propuestas por la Data Science Process Alliance (2021) en su exposición de las fases del modelo.

El modelo CRISP-DM cuenta con seis fases:

1. Comprensión del negocio

“En esta fase se espera contestar preguntas como; ¿a qué se dedica el negocio? ¿Qué necesita el negocio?, en base a las respuestas de esas preguntas se puede comenzar a dar sentido a los datos que se posea” (Data Science Process Alliance, 2021)

2. Comprensión de datos

“En esta fase se espera contestar las preguntas; ¿Qué datos se tiene?, ¿Qué datos se necesitan?, ¿Están limpios los datos?, ¿Los datos requieren ser procesados?, y desarrollar estrategias o actividades para responder estas preguntas” (Data Science Process Alliance, 2021)

3. Preparación de datos

Dado que este ciclo de vida está orientado a ser aplicado en aprendizaje “automático, en esta fase es importante contestar la pregunta ¿Cómo organizar los datos para el modelado?, de tal manera que se puedan realizar análisis como impacto de las características de entrada en el target de salida a través de correlaciones u otros análisis, así como también posibles relaciones o redundancias entre las características de entrada” (Data Science Process Alliance, 2021)

4. Modelado

“Considerando la naturaleza de los datos en esta fase se debe satisfacer la pregunta /Qué técnicas de modelado se pueden aplicar?, la cual tendrá relación con la naturaleza de los datos y el giro del negocio. En este punto se pueden considerar varios modelos en los que se puedan variar sus hiperparámetros para que se ajusten de mejor manera a los datos” (Data Science Process Alliance, 2021)

5. Evaluación

“En la fase de evaluación, una vez que se han encontrado los modelos que se ajusten a los datos y al giro del negocio, es necesario contestar la pregunta; ¿Qué cumple mejor con los objetivos de negocio?, y la respuesta a esto puede estar orientado tanto a los modelos de aprendizaje de máquina, como de sus hiperparámetros. De ser necesario se puede regresar a cualquiera de las fases previas para realizar los ajustes que correspondan” (Data Science Process Alliance, 2021)

6. Despliegue

“En la última fase y a través de la pregunta; ¿Cómo acceden las partes interesadas a los resultados?, se deben encontrar estrategias de implementación y despliegue de los resultados obtenidos” (Data Science Process Alliance, 2021)

3.2 Aprendizaje supervisado

El aprendizaje supervisado es aquel en el que se cuenta con un conjunto de datos en los que se pueden identificar las variables predictoras de entrada o características y por lo menos una variable de salida u objetivo que se la conoce como etiquetas.

Si la variable objetivo es numérica y continua se pueden aplicar modelos de aprendizaje de regresión, mientras que si es categórica de carácter ordinal o no, se pueden aplicar modelos de clasificación.

3.2.1 Modelos de Clasificación

“La clasificación de datos es un proceso de dos pasos, en el primer paso, se crea un clasificador que describe un conjunto predeterminado de clases de datos o conceptos. Este es el paso de aprendizaje (o fase de entrenamiento), donde un algoritmo de clasificación construye el clasificador analizando o -aprendiendo de- un conjunto de entrenamiento. En el segundo paso el modelo se utiliza para la clasificación. En primer lugar, se estima la precisión predictiva del clasificador. Si se el conjunto de entrenamiento para medir la precisión del clasificador, esta estimación probablemente sería optimista, porque el clasificador tiende a sobreajustar los datos. Por lo tanto, se utiliza un conjunto de prueba, formado por tuplas de prueba y sus etiquetas de clase asociadas. Estas tuplas se

seleccionan aleatoriamente del conjunto de datos generales. Son independientes de las tuplas de entrenamiento". (Jiawei Han,2006)

3.2.1.1 Clasificador de k-vecinos cercanos KNN

"El método del k-vecinos cercanos se describió por primera vez a principios de la década de 1950, los clasificadores de vecinos más cercanos se basan en el aprendizaje por analogía, es decir, comparando una tupla de prueba dada con tuplas de entrenamiento que son similares a ella. Las tuplas de entrenamiento están descritas por n atributos. Cada tupla representa un punto en un espacio n-dimensional. De esta forma, todas las tuplas de entrenamiento se almacenan en un espacio patrón de n dimensiones. Cuando se le da una tupla desconocida, un clasificador de k-vecinos cercanos busca en el espacio del patrón las k tuplas de entrenamiento que están más cerca de la tupla desconocida" (Jiawei Han,2006) y que servirán de referencia para asignar una clase al valor desconocido.

3.2.1.2 Clasificador de árboles de decisión

"Un árbol de decisión es una estructura de árbol similar a un diagrama de flujo, donde cada nodo denota una prueba en un valor de atributo, cada rama representa un resultado de la prueba y las hojas del árbol representan clases o distribuciones de clases. Los árboles de decisión se pueden convertir fácilmente en reglas de clasificación" (Jiawei Han,2006)

3.2.1.1 Clasificador Rain Forest

"Para mejorar aún más la escalabilidad de la inducción del árbol de decisión, se propuso un método llamado Rain Forest. Se adapta a la cantidad de memoria principal disponible y se aplica a cualquier algoritmo de inducción de árboles de decisión. El método mantiene un conjunto AVC (Valor de atributo, etiqueta de clase) para cada atributo, en cada nodo del árbol, que describe las tuplas de entrenamiento en el nodo. El conjunto AVC de un atributo A en el nodo N proporciona los recuentos de etiquetas de clase para cada valor de A para las tuplas en N. Por lo general, este tamaño debe caber en la memoria, incluso para datos del mundo real. Sin embargo, RainForest tiene técnicas para manejar el caso en el que el grupo AVC no cabe en la memoria" (Jiawei Han,2006)

3.2.1.4 Clasificador por Regresión Logística

“La regresión logística modela la probabilidad de que ocurra algún evento como una función lineal de un conjunto de variables predictoras. Los datos de conteo exhiben con frecuencia una distribución de Poisson y comúnmente se modelan usando la regresión de Poisson. Los modelos logarítmicos lineales aproximan distribuciones de probabilidad multidimensionales discretas. Pueden usarse para estimar el valor de probabilidad asociado con las celdas del cubo de datos... La técnica se amplía bien para permitir muchas dimensiones. Además de la predicción, el modelo logarítmico lineal es útil para la compresión de datos” (Jiawei Han,2006)

3.2.1.5 Matriz de Confusión

“La matriz de confusión es una herramienta útil para analizar qué tan bien un clasificador puede reconocer tuplas de diferentes clases. Dadas m clases, una matriz de confusión es una tabla de al menos tamaño m por m. Una entrada, CM i, j en las primeras m filas y m columnas indica el número de tuplas de clase i que fueron etiquetadas por el clasificador como clase j. Para que un clasificador tenga una buena precisión, idealmente la mayoría de las tuplas se representarían a lo largo de la diagonal de la matriz de confusión, desde la entrada CM 1 , 1 hasta la entrada CM m , m , con el resto de las entradas cerca de cero. La tabla puede tener filas o columnas adicionales para proporcionar totales o tasas de reconocimiento por clase.” (Jiawei Han,2006)

3.3 Aprendizaje no supervisado

Cuando no se cuenta con las variables de salida o etiquetas, los datos pueden ser agrupados bajo distintos criterios buscando analogías o disimilitudes entre ellos. Este agrupamiento se conoce como aprendizaje no supervisado, ya que la máquina no tiene experiencias previas para aprender.

CAPÍTULO 4

4. MINERÍA DE DATOS PARA EL MODELADO DE APRENDIZAJE AUTOMÁTICO EN LA RESISTENCIA DE MATERIALES POR LA INCLUSIÓN DE FIBRAS NATURALES.

En este acápite se estudian, usando la metodología CRISP-DM, los modelos de aprendizaje automático que podrían aplicarse en los datos recopilados en el LMC en el estudio de la influencia de la inclusión de fibra de abacá en la resistencia a la compresión de limos arenosos.

Conforme a como se han planteado los objetivos, las etapas de Comprensión del negocio, Comprensión de datos y Preparación de datos se desarrollan en el acápite 4.1, mientras que el modelado se realiza en el 4.2, quedando la evaluación en el acápite 4.3.

4.1 Uso de la la metodología CRISP-DM para entender y preparar la data para aprendizaje automático.

4.1.1 Comprensión del negocio

"El Laboratorio de Resistencia de Materiales, Mecánica de Suelos, Pavimentos y Geotécnica de la Pontificia Universidad Católica del Ecuador, funciona desde 1965 y consta en el Registro de Consultoría como Entidad Universitaria Consultora. Las actividades que se desarrollan en este centro, también conocido como Laboratorio de Materiales de Construcción (LMC), están orientadas a la investigación, control de calidad de los materiales y consultoría en proyectos de ingeniería, así como la fiscalización de obras civiles." (PUCE,2022)[11]

Dentro de sus actividades de investigación y cobijado bajo el paraguas del dominio "Hábitat, infraestructura y movilidad" en el desarrollo de la línea de investigación "Diseño, infraestructura y sistemas sociales y ambientales para un hábitat sostenible"[12], se ha enfocado a analizar el impacto en la resistencia de materiales al introducir en su composición elementos autóctonos del Ecuador, como por ejemplo la fibra de abacá en los limos arenosos. Para ello se ha basado en la amplia experiencia y trayectoria del Laboratorio que cuenta tanto con el equipamiento necesario para realizar los diferentes tipos de ensayos como el personal capacitado.

11 Pontificia Universidad Católica del Ecuador, (2022). Laboratorio de Resistencia de Materiales, Mecánica de Suelos, Pavimentos y Geotécnica https://www.puce.edu.ec/servicios/laboratorio-de-suelos/

12 Dominios Académicos y Líneas de Investigación, PUCE, 2017

La investigación desarrollada en el LMC por Albuja J. y otros[13] "determina la influencia de la inclusión de fibras de abacá en la resistencia a la compresión simple de especímenes reconstituidos de limos arenosos. Para ello se añade fibra en diferentes longitudes (5, 10 y 15 mm) y cantidades (0,5, 1,0, 1,5 y 2,0% del peso seco del suelo), evaluando el comportamiento físico y mecánico de las mezclas suelo-fibras mediante ensayos de compactación y compresión no confinada" (Albuja J. otros, 2022).

Como resultado de los ensayos, "Las pruebas realizadas mostraron un aumento en el esfuerzo máximo de compresión de hasta 1235.1% (en comparación con el suelo libre de fibra) cuando se incrementa el contenido de fibra, su longitud o ambos. La falla por compresión ocurrió con una deformación axial mayor cuando se agregó fibra de 10 y 15 mm en una dosis del 1% o en porcentajes iguales o superiores al 1,5%, independientemente de la longitud de la fibra. Una serie de modelos mixtos lineales identificaron efectos estadísticamente significativos de la longitud y el porcentaje de fibra sobre el nivel de esfuerzo y sobre la deformación unitaria." (Albuja J. Y otros, 2021).

4.1.2 Comprensión de datos

Para identificar la(s) entrada(s) y salida(s), este análisis se apega al concepto de considerar característica(s) a la(s) entradas y objetivo(s) a la salida(s)

Los datos con los que se cuenta, obtenidos de las mediciones realizadas en el LMC de la PUCE, con el objeto de medir el efecto de introducir fibra de abacá en los limos arenosos y determinar cuánto afecta a su resistencia es abundante ya que se cuenta con más de 1600 instancias, y es necesario analizar si la data requiere de una preparación o pre procesamiento para determinar si es posible aplicar un modelo de aprendizaje automático que permita predecir la influencia de materiales como la fibra de abacá en la resistencia a la compresión de limos arenosos.

En el experimento desarrollado por Albuja y otros "Se seleccionaron un total de cuatro concentraciones de fibra: 0.5%, 1.0%, 1.5% y 2.0% en relación al peso seco del suelo .. y

13 Albuja-Sánchez, J.; Alcívar, E.; Escobar, D.; Montero, J.; Realpe, G.; Muñoz, A.; Peñaherrera-Aguirre, M. Influence of Abaca Fiber Inclusion on the Unconfined Compressive Keywords: simple compression test; unconfined compression test; abaca fiber; soil improvement; fiber-reinforced soil Strength of Reconstituted Sandy Silts. Fibers 2022, 10, 99. https://doi.org/10.3390/fib10110099

se cortaron tres longitudes diferentes: 5 mm, 10 mm y 15 mm. Con los porcentajes y longitudes de fibra seleccionados, se prepararon 13 combinaciones diferentes de tenacidad como se muestra en la Tabla 1”

Tabla 1: Soil-fiber mixtures.(Influence of Abaca Fiber Inclusion on the Unconfined Compressive Strength of reconstituted sandy silts, Albuja, J y otros, 2022)

Mixture No.	Fiber Content (%)	Fiber Length (mm)	No. of specimens
1	0	0	2
2	0.5	5	2
3	1.0	5	2
4	1.5	5	2
5	2.0	5	2
6	0.5	10	2
7	1.0	10	2
8	1.5	10	2
9	2.0	10	2
10	0.5	15	2
11	1.0	15	2
12	1.5	15	2
13	2.0	15	2

En este punto se pueden identificar dos características:

- “Fiber Content (%)”
- “Fiber Length (mm)”

La prueba de compresión no confinada simple realizada por Albuja y su equipo, “aplicó una carga para producir una tasa de deformación axial del 1 % en la muestra por cada minuto. El esfuerzo máximo de compresión no confinada ocurre cuando la muestra indica una falla frágil o cuando su deformación unitaria alcanza el 15%”.

La figura 2 muestra el equipo usado para realizar la prueba y el estado de la muestra durante y después del experimento.

(a) (b) (c)

Figura 2: a) Marco de carga digital Tritest ELE modelo 25-3518/02 b) Muestra durante la prueba c) Muestra después de la prueba (Albuja y otros, 2022)

En este punto se puede identificar una tercera variable:

- “Fuerza de compresión no confinada (kPa)” o FC-NC

Sin embargo, se debe identificar si es una característica u objetivo.

La cuarta variable corresponde al porcentaje de la deformación del cilindro, resultado de la aplicación de velocidad de ensayo en las probetas de suelo-fibra que fue de 0.73 mm/min (1% de la altura de la muestra), información obtenida por Albuja:

- “Deformación” (%)

En las Figuras 3 4 y 5 se pueden apreciar las curvas de tensión-deformación con variaciones del contenido de fibra y la longitud de la fibra (a) 5 mm; (b) 10 mm; (c) 15 mm, en las que se puede observar que los cilindros de prueba se deforman una vez que se alcanza el pico máximo de compresión y luego de esto la tensión comienza a disminuir, así para un mismo valor de tensión existirían dos porcentajes de deformación, el uno previo a la ruptura del cilindro y el otro luego de la misma. Por tanto, se necesita de una variable de discriminación que permita identificar cuando ha ocurrido la ruptura.

La prueba concluye una vez que se consigue “el esfuerzo máximo de compresión no confinado, el mismo que ocurre cuando la muestra indica una falla frágil o cuando su deformación unitaria alcanza el 15%”, por tanto, la quinta variable sería:

- “Esfuerzo máximo de compresión no confinado (Booleano)” ó EMC-NC (Booleano)

La cual puede ser identificada como la variable objetivo de naturaleza booleana.

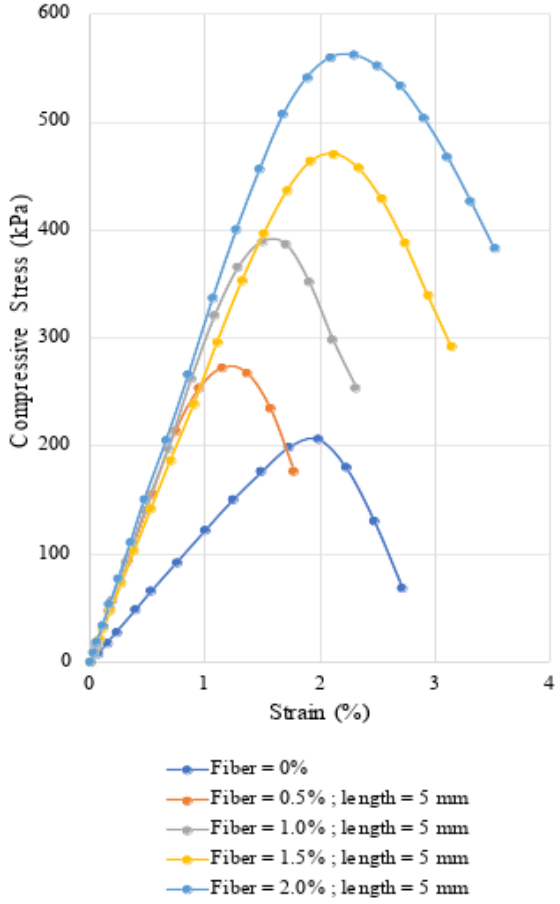


Figura 3: Curva tensión-deformación con variaciones del contenido de fibra y la longitud de fibra de 5 mm (Albuja y otros, 2022)

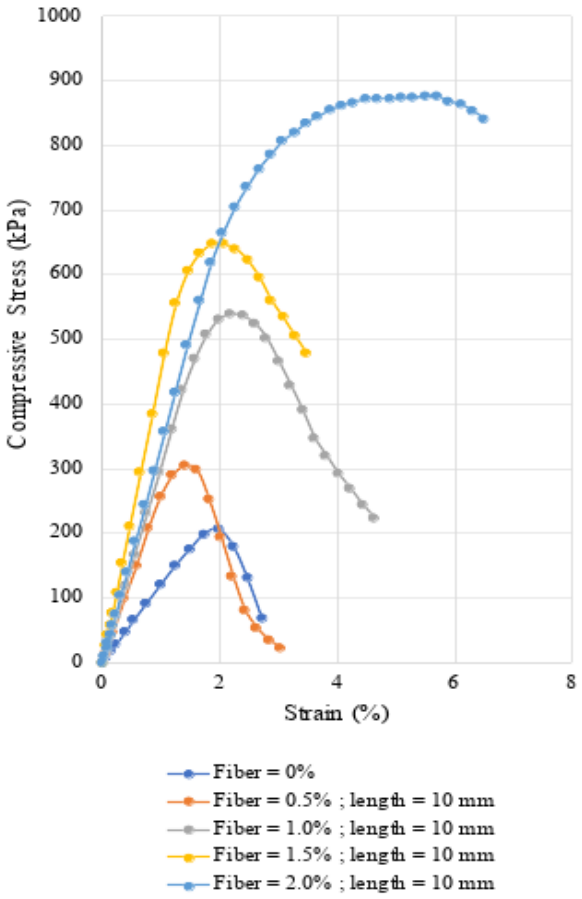


Figura 4: Curva tensión-deformación con variaciones del contenido de fibra y longitud de fibra de 10 mm (Albuja y otros, 2022)

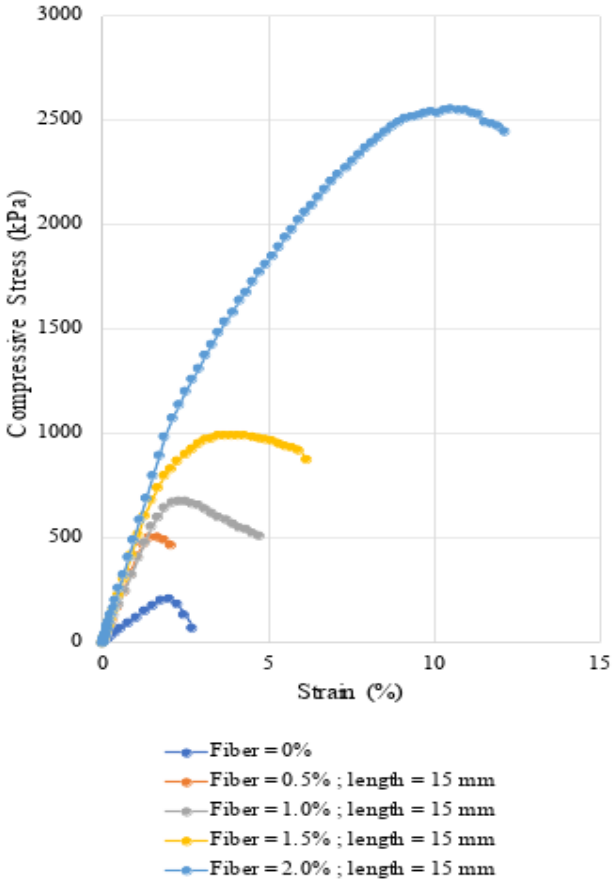


Figura 5: Curva tensión-deformación con variaciones del contenido de fibra y la longitud de fibra de 15 mm (Albuja y otros, 2022)

De esta manera las características y variable objetivo se muestran en la Tabla 2 y necesitan ser contrastadas con el conjunto de datos generados en el LMC.

Tabla 2: Variables identificadas en el artículo publicado por Albuja y otros

Tipo de variable:	Características				Objetivo
Variable:	Fiber Content	Fiber Length	FC-NC	Deformación	EMC-NC
Unidad:	%	mm	kPa	%	Si/No

En resumen:

Fiber Content: Porcentaje de contenido de fibra de abacá
Fiber Length: Longitud de la Fibra
FC-NC: Fuerza de compresión no confinada
Deformación: Deformación de la probeta
EMC-NC: Esfuerzo máximo de compresión no confinado

Por otro lado, se tienen los datos que se han extraído del estudio realizado en el Laboratorio de Suelos el mismo tiene que ser contrastado con las variables de la tabla 2.

Para el análisis del conjunto de datos se usó el lenguaje de programación Python y como IDE el cuaderno de Jupyter.

A continuación, se muestran algunas capturas de pantalla y el análisis respectivo.

```
In [4]: import pandas as pd
        lime=pd.read_csv("Lime.csv",sep=",")

In [9]: lime.shape
Out[9]: (1604, 13)

In [6]: lime.head()
Out[6]:
```

	ID	Tipo	Cm	Per	Cm*Percent	Def	Fu	DefU	Esf	LOGDef	LOGFu	LOGDefU	LOGEsf
0	Pb48	1.5cm 1.5%	1.5	1.5	2.25	0.000000	-3.864000	0.000000	-4.382997	0.000000	-0.866461	0.000000	-5.563360
1	Pb24	1.5cm 2%	1.5	2.0	3.00	0.073900	-3.477600	0.100349	-4.004228	0.030964	-0.281997	0.041530	-0.421622
2	Pb27	.5cm.5%	0.5	0.5	0.25	0.000000	-3.091200	0.000000	-3.528832	0.000000	-0.041532	0.000000	-0.068457
3	Pb26	.5cm.5%	0.5	0.5	0.25	3.747793	-3.091200	5.208645	-3.347032	0.676492	-0.041532	0.792997	0.015346
4	Pb1	Control	0.0	0.0	0.00	0.000000	-2.590673	0.000000	-2.945085	0.000000	0.149012	0.000000	0.157733

Figura 6: Cabecera del conjunto de datos

En el conjunto de datos de las pruebas realizadas en laboratorio se puede apreciar que existen 13 columnas y 1604 instancias.

La identificación de cada una de las columnas corresponde a:

ID: Identificación de la probeta de prueba
Tipo: Descripción de los parámetros de mezcla
Cm: Longitud de la fibra de abacá
Per: Porcentaje de contenido de fibra de abacá
Cm*Percent: Descripción largo de la fibra y porcentaje de la misma
Def: Deformación de la probeta
Fu: Fuerza aplicada
DefU: Deformación unitaria
Esf: Esfuerzo
LOGDef: Logaritmo de la deformación
LOGFu: Logaritmo de la fuerza
LOGDefU: Logaritmo de la deformación unitaria
LOGEsf: Logaritmo del esfuerzo

Adicionalmente se pueden apreciar los estadísticos de los datos:

```
In [6]: # Estadísticos
        lime.describe()
```

Out[6]:

	Cm	Per	Cm*Percent	Def	Fu	DefU	Esf	LOGDef	LOGFu	LOGDefU	LOGEsf
count	1604.000000	1604.000000	1604.000000	1604.000000	1604.000000	1604.000000	1604.000000	1604.000000	1604.000000	1604.000000	1604.000000
mean	1.120948	1.367830	1.646976	2.511850	585.129356	3.436644	624.443489	0.446589	2.415646	0.527143	2.453479
std	0.444370	0.604351	0.970771	2.531371	643.472190	3.447952	662.446248	0.290574	0.692672	0.325213	0.709262
min	0.000000	0.000000	0.000000	0.000000	-3.864000	0.000000	-4.382997	0.000000	-0.866461	0.000000	-5.563360
25%	1.000000	1.000000	0.750000	0.656645	103.845000	0.902304	116.516657	0.219229	2.032799	0.279280	2.082424
50%	1.500000	1.500000	1.500000	1.732904	372.296400	2.403156	416.946332	0.436623	2.575530	0.531882	2.624622
75%	1.500000	2.000000	2.250000	3.473300	796.660200	4.740938	868.834923	0.650628	2.903446	0.758983	2.941123
max	1.500000	2.000000	3.000000	12.341300	3106.656000	16.809943	3062.888148	1.125198	3.492852	1.250663	3.486752

Figura 7: Estadísticos del conjunto de datos

4.1.3 Preparación de datos

Análisis de características (Variables de entrada)

Una vez revisado el artículo publicado por Albuja y otros con el conjunto de datos de las pruebas realizadas en el Laboratorio se encuentra que existen diferencia de las denominaciones de las variables, así como un número mayor de ellas debido a nuevas columnas que resultan de las combinaciones u operaciones entre ellas.

Para extraer las características de interés y homologar las variables se construye la Tabla 3: Comparación de variables entre el conjunto de datos y el artículo, de donde se extraen como características de interés las que corresponden al artículo publicado por Albuja y otros.

Tabla 3: Comparación de variables entre el conjunto de datos y el artículo

Variable artículo	**Variable laboratorio**	**Descripción**
	ID	Identificación de la probeta de prueba
	Tipo	Descripción de los parámetros de mezcla
Fiber_Length	Cm	Longitud de la fibra de abacá
Fiber_Content	Per	Porcentaje de contenido de fibra de abacá
	Cm*Percent	Descripción largo de la fibra y porcentaje de la misma
Deformation	Def	Deformación de la probeta
FC-NC	Fu	Fuerza de compresión no confinada
	DefU	Deformación unitaria
	Esf	Esfuerzo
	LOGDef	Logaritmo de la deformación
	LOGFu	Logaritmo de la fuerza
	LOGDefU	Logaritmo de la deformación unitaria

	LOGEsf	Logaritmo del esfuerzo
EMC-NC		Esfuerzo máximo de compresión no confinado

Conforme lo mencionado en el párrafo anterior y una vez que han sido identificadas, es necesario construir un nuevo dataset que contenga las características de interés o variables de entrada:

Figura 8: Características seleccionadas del conjunto de datos

```
In [24]: lime_proc=lime[['ID','Cm','Per','Def','Fu']]

In [25]: lime_proc.head()
```

Out[25]:

	ID	Cm	Per	Def	Fu
0	Pb48	1.5	1.5	0.000000	-3.864000
1	Pb24	1.5	2.0	0.073900	-3.477600
2	Pb27	0.5	0.5	0.000000	-3.091200
3	Pb26	0.5	0.5	3.747793	-3.091200
4	Pb1	0.0	0.0	0.000000	-2.590673

Variable objetivo (Variable de salida o etiqueta)

Como variable objetivo conforme lo discutido en la sección **¡Error! No se encuentra el origen de la referencia.**, se puede identificar al Esfuerzo máximo de compresión no confinado “EMC-NC”, la misma que puede ser construida con el valor booleano de “0” cuando la probeta no se ha roto, o “1” luego de su rompimiento, considerando que el esfuerzo máximo de compresión no confinada ocurre cuando la muestra indica una falla frágil o cuando su deformación unitaria alcanza el 15%”.

Al analizar los datos con mayor detalle se observa que estos se encuentran desordenados

```
In [31]: lime_proc.head(10)
```

Out[31]:

	ID	Cm	Per	Def	Fu
0	Pb48	1.5	1.5	0.000000	-3.864000
1	Pb24	1.5	2.0	0.073900	-3.477600
2	Pb27	0.5	0.5	0.000000	-3.091200
3	Pb26	0.5	0.5	3.747793	-3.091200
4	Pb1	0.0	0.0	0.000000	-2.590673
5	Pb3	1.0	0.5	0.000000	-2.158895
6	Pb2	0.0	0.0	0.000000	-2.158895
7	Pb21	1.5	1.5	0.000000	-1.932000
8	Pb43	0.5	0.5	0.000000	-1.545600
9	Pb32	1.5	2.0	0.000000	-1.545600

como se puede apreciar en la Figura 9: Primeros 10 registros del dataset, en la cual las probetas se muestran mezcladas, por tanto será necesario ordenarlas por ’ID’, ‘Cm’ y ‘Def’ mientras que la variable ‘Fu’ debería llegar a un máximo y luego de eso disminuir, tal como se muestra en las curvas de la

```
In [30]: lime_proc.sort_values(['ID','Cm','Def'], ascending=True)
```

Out[30]:

	ID	Cm	Per	Def	Fu
4	Pb1	0.0	0.0	0.000000	-2.590673
71	Pb1	0.0	0.0	0.073900	6.908462
101	Pb1	0.0	0.0	0.221700	11.226251
131	Pb1	0.0	0.0	0.369500	16.839377
159	Pb1	0.0	0.0	0.517300	22.884282
195	Pb1	0.0	0.0	0.665100	30.224523
232	Pb1	0.0	0.0	0.812900	38.860101
256	Pb1	0.0	0.0	0.960700	47.927458
292	Pb1	0.0	0.0	1.108500	57.858373
328	Pb1	0.0	0.0	1.256300	72.538855
365	Pb1	0.0	0.0	1.404100	86.787559
404	Pb1	0.0	0.0	1.551900	105.785830
439	Pb1	0.0	0.0	1.699700	126.511218
469	Pb1	0.0	0.0	1.847500	141.623479
485	Pb1	0.0	0.0	1.995300	152.849731
482	Pb1	0.0	0.0	2.143100	151.554394
436	Pb1	0.0	0.0	2.290900	126.942997
361	Pb1	0.0	0.0	2.438700	87.651117
263	Pb1	0.0	0.0	2.586500	51.381689
202	Pb1	0.0	0.0	2.734300	32.383417
140	Pb1	0.0	0.0	2.882100	18.998272

Figura 10, lo que ocurre una vez que la probeta se ha roto.

```
In [31]: lime_proc.head(10)
```

Out[31]:

	ID	Cm	Per	Def	Fu
0	Pb48	1.5	1.5	0.000000	-3.864000
1	Pb24	1.5	2.0	0.073900	-3.477600
2	Pb27	0.5	0.5	0.000000	-3.091200
3	Pb26	0.5	0.5	3.747793	-3.091200
4	Pb1	0.0	0.0	0.000000	-2.590673
5	Pb3	1.0	0.5	0.000000	-2.158895
6	Pb2	0.0	0.0	0.000000	-2.158895
7	Pb21	1.5	1.5	0.000000	-1.932000
8	Pb43	0.5	0.5	0.000000	-1.545600
9	Pb32	1.5	2.0	0.000000	-1.545600

Figura 9: Primeros 10 registros del dataset

Si se observa la columna 'Def' de la probeta en la

```
In [30]: lime_proc.sort_values(['ID','Cm','Def'], ascending=True)
```

Out[30]:

	ID	Cm	Per	Def	Fu
4	Pb1	0.0	0.0	0.000000	-2.590673
71	Pb1	0.0	0.0	0.073900	6.908462
101	Pb1	0.0	0.0	0.221700	11.226251
131	Pb1	0.0	0.0	0.369500	16.839377
159	Pb1	0.0	0.0	0.517300	22.884282
195	Pb1	0.0	0.0	0.665100	30.224523
232	Pb1	0.0	0.0	0.812900	38.860101
256	Pb1	0.0	0.0	0.960700	47.927458
292	Pb1	0.0	0.0	1.108500	57.858373
328	Pb1	0.0	0.0	1.256300	72.538855
365	Pb1	0.0	0.0	1.404100	86.787559
404	Pb1	0.0	0.0	1.551900	105.785830
439	Pb1	0.0	0.0	1.699700	126.511218
469	Pb1	0.0	0.0	1.847500	141.623479
485	Pb1	0.0	0.0	1.995300	152.849731
482	Pb1	0.0	0.0	2.143100	151.554394
436	Pb1	0.0	0.0	2.290900	126.942997
361	Pb1	0.0	0.0	2.438700	87.651117
263	Pb1	0.0	0.0	2.586500	51.381689
202	Pb1	0.0	0.0	2.734300	32.383417
140	Pb1	0.0	0.0	2.882100	18.998272

Figura 10 se puede apreciar que esta es creciente, sin embargo, el momento en que la probeta se rompe ocurre cuando la fuerza 'Fu' llega al máximo, en este caso "152,84" y luego de esto empieza a disminuir.

```
In [30]: lime_proc.sort_values(['ID','Cm','Def'], ascending=True)
```

Out[30]:

	ID	Cm	Per	Def	Fu
4	Pb1	0.0	0.0	0.000000	-2.590673
71	Pb1	0.0	0.0	0.073900	6.908462
101	Pb1	0.0	0.0	0.221700	11.226251
131	Pb1	0.0	0.0	0.369500	16.839377
159	Pb1	0.0	0.0	0.517300	22.884282
195	Pb1	0.0	0.0	0.665100	30.224523
232	Pb1	0.0	0.0	0.812900	38.860101
256	Pb1	0.0	0.0	0.960700	47.927458
292	Pb1	0.0	0.0	1.108500	57.858373
328	Pb1	0.0	0.0	1.256300	72.538855
365	Pb1	0.0	0.0	1.404100	86.787559
404	Pb1	0.0	0.0	1.551900	105.785830
439	Pb1	0.0	0.0	1.699700	126.511218
469	Pb1	0.0	0.0	1.847500	141.623479
485	Pb1	0.0	0.0	1.995300	152.849731
482	Pb1	0.0	0.0	2.143100	151.554394
436	Pb1	0.0	0.0	2.290900	126.942997
361	Pb1	0.0	0.0	2.438700	87.651117
263	Pb1	0.0	0.0	2.586500	51.381689
202	Pb1	0.0	0.0	2.734300	32.383417
140	Pb1	0.0	0.0	2.882100	18.998272

Figura 10: Datos de la probeta 1 Pb1 ordenados

Con este criterio de puede construir la variable objetivo, de tal manera que cuando se llegue al máximo, la variable objetivo 'EMC-NC' cambie de "0" a "1", lo cual deberá realizarse para cada una de las probetas mostradas en la

```
lime_sorted['ID'].unique()
```

```
array(['Pb1', 'Pb10', 'Pb11', 'Pb12', 'Pb13', 'Pb14', 'Pb15', 'Pb16',
       'Pb17', 'Pb18', 'Pb19', 'Pb2', 'Pb20', 'Pb21', 'Pb22', 'Pb23',
       'Pb24', 'Pb25', 'Pb26', 'Pb27', 'Pb28', 'Pb3', 'Pb30', 'Pb31',
       'Pb32', 'Pb33', 'Pb34', 'Pb35', 'Pb36', 'Pb37', 'Pb38', 'Pb39',
       'Pb4', 'Pb40', 'Pb41', 'Pb42', 'Pb43', 'Pb44', 'Pb45', 'Pb46',
       'Pb47', 'Pb48', 'Pb49', 'Pb5', 'Pb50', 'Pb51', 'Pb52', 'Pb53',
       'Pb54', 'Pb6', 'Pb7', 'Pb8', 'Pb9'], dtype=object)
```

Figura 11


```
lime_sorted['ID'].unique()
```

```
array(['Pb1', 'Pb10', 'Pb11', 'Pb12', 'Pb13', 'Pb14', 'Pb15', 'Pb16',
       'Pb17', 'Pb18', 'Pb19', 'Pb2', 'Pb20', 'Pb21', 'Pb22', 'Pb23',
       'Pb24', 'Pb25', 'Pb26', 'Pb27', 'Pb28', 'Pb3', 'Pb30', 'Pb31',
       'Pb32', 'Pb33', 'Pb34', 'Pb35', 'Pb36', 'Pb37', 'Pb38', 'Pb39',
       'Pb4', 'Pb40', 'Pb41', 'Pb42', 'Pb43', 'Pb44', 'Pb45', 'Pb46',
       'Pb47', 'Pb48', 'Pb49', 'Pb5', 'Pb50', 'Pb51', 'Pb52', 'Pb53',
       'Pb54', 'Pb6', 'Pb7', 'Pb8', 'Pb9'], dtype=object)
```


Figura 11: Listado de probetas

El código para construir la variable objetivo y los primeros resultados se muestran en la

```
Probeta=lime_sorted.iloc[0]['ID']
for i in range(len(lime_sorted)):
    if Probeta==lime_sorted.iloc[i]['ID']:
        if (lime_sorted.iloc[i]['Fu'] > lime_sorted.iloc[i-1]['Fu']) or (i==0) :
            print(i," ",lime_sorted.iloc[i]['ID']," ",lime_sorted.iloc[i]['Fu']," ",0)
            lime_sorted.loc[i,'EMC-NC']=0
        else:
            print(i," ",lime_sorted.iloc[i]['ID']," ",lime_sorted.iloc[i]['Fu']," ",1)
            lime_sorted.loc[i,'EMC-NC']=1
    else:
        Probeta=lime_sorted.iloc[i]['ID']
        print(i," ",lime_sorted.iloc[i]['ID']," ",lime_sorted.iloc[i]['Fu']," ",0)
        lime_sorted.loc[i,'EMC-NC']=0
```

```
0   Pb1   -2.590673   0
1   Pb1   6.908462   0
2   Pb1   11.226251   0
3   Pb1   16.839377   0
4   Pb1   22.884282   0
5   Pb1   30.224523   0
6   Pb1   38.860101   0
7   Pb1   47.927458   0
8   Pb1   57.858373   0
9   Pb1   72.538855   0
10   Pb1   86.787559   0
11   Pb1   105.78583   0
12   Pb1   126.511218   0
13   Pb1   141.623479   0
14   Pb1   152.849731   0
15   Pb1   151.554394   1
16   Pb1   126.942997   1
17   Pb1   87.651117   1
18   Pb1   51.381689   1
19   Pb1   32.383417   1
20   Pb1   18.998272   1
21   Pb10   0.0   0
22   Pb10   16.19217   0
23   Pb10   30.5388005   0
24   Pb10   40.16153946   0
```

Figura 12, donde se puede apreciar el valor que se le está asignando a la variable ‘EMC-NC’.


```
Probeta=lime_sorted.iloc[0]['ID']
for i in range(len(lime_sorted)):
    if Probeta==lime_sorted.iloc[i]['ID']:
        if (lime_sorted.iloc[i]['Fu'] > lime_sorted.iloc[i-1]['Fu']) or (i==0) :
            print(i," ",lime_sorted.iloc[i]['ID']," ",lime_sorted.iloc[i]['Fu']," ",0)
            lime_sorted.loc[i,'EMC-NC']=0
        else:
            print(i," ",lime_sorted.iloc[i]['ID']," ",lime_sorted.iloc[i]['Fu']," ",1)
            lime_sorted.loc[i,'EMC-NC']=1
    else:
        Probeta=lime_sorted.iloc[i]['ID']
        print(i," ",lime_sorted.iloc[i]['ID']," ",lime_sorted.iloc[i]['Fu']," ",0)
        lime_sorted.loc[i,'EMC-NC']=0
```

```
0   Pb1   -2.590673   0
1   Pb1   6.908462   0
2   Pb1   11.226251   0
3   Pb1   16.839377   0
4   Pb1   22.884282   0
5   Pb1   30.224523   0
6   Pb1   38.860101   0
7   Pb1   47.927458   0
8   Pb1   57.858373   0
9   Pb1   72.538855   0
10   Pb1   86.787559   0
11   Pb1   105.78583   0
12   Pb1   126.511218   0
13   Pb1   141.623479   0
14   Pb1   152.849731   0
15   Pb1   151.554394   1
16   Pb1   126.942997   1
17   Pb1   87.651117   1
18   Pb1   51.381689   1
19   Pb1   32.383417   1
20   Pb1   18.998272   1
21   Pb10   0.0   0
22   Pb10   16.19217   0
23   Pb10   30.5388005   0
24   Pb10   40.16153946   0
```


Figura 12: Código para construir la variable objetivo y primeros resultados

	ID	Cm	Per	Def	Fu	EMC-NC
0	Pb1	0.0	0.0	0.0000	-2.590673	0.0
1	Pb1	0.0	0.0	0.0739	6.908462	0.0
2	Pb1	0.0	0.0	0.2217	11.226251	0.0
3	Pb1	0.0	0.0	0.3695	16.839377	0.0
4	Pb1	0.0	0.0	0.5173	22.884282	0.0
5	Pb1	0.0	0.0	0.6651	30.224523	0.0
6	Pb1	0.0	0.0	0.8129	38.860101	0.0
7	Pb1	0.0	0.0	0.9607	47.927458	0.0
8	Pb1	0.0	0.0	1.1085	57.858373	0.0
9	Pb1	0.0	0.0	1.2563	72.538855	0.0
10	Pb1	0.0	0.0	1.4041	86.787559	0.0
11	Pb1	0.0	0.0	1.5519	105.785830	0.0
12	Pb1	0.0	0.0	1.6997	126.511218	0.0
13	Pb1	0.0	0.0	1.8475	141.623479	0.0
14	Pb1	0.0	0.0	1.9953	152.849731	0.0
15	Pb1	0.0	0.0	2.1431	151.554394	1.0
16	Pb1	0.0	0.0	2.2909	126.942997	1.0
17	Pb1	0.0	0.0	2.4387	87.651117	1.0
18	Pb1	0.0	0.0	2.5865	51.381689	1.0
19	Pb1	0.0	0.0	2.7343	32.383417	1.0
20	Pb1	0.0	0.0	2.8821	18.998272	1.0

La Figura 13 muestra las características de entrada y variable objetivo de la “probeta 1” donde se puede notar el momento del rompimiento de la misma en el registro 15. Esta situación se repite para todas

las probetas listadas en la

```
lime_sorted['ID'].unique()
```

```
array(['Pb1', 'Pb10', 'Pb11', 'Pb12', 'Pb13', 'Pb14', 'Pb15', 'Pb16',
       'Pb17', 'Pb18', 'Pb19', 'Pb2', 'Pb20', 'Pb21', 'Pb22', 'Pb23',
       'Pb24', 'Pb25', 'Pb26', 'Pb27', 'Pb28', 'Pb3', 'Pb30', 'Pb31',
       'Pb32', 'Pb33', 'Pb34', 'Pb35', 'Pb36', 'Pb37', 'Pb38', 'Pb39',
       'Pb4', 'Pb40', 'Pb41', 'Pb42', 'Pb43', 'Pb44', 'Pb45', 'Pb46',
       'Pb47', 'Pb48', 'Pb49', 'Pb5', 'Pb50', 'Pb51', 'Pb52', 'Pb53',
       'Pb54', 'Pb6', 'Pb7', 'Pb8', 'Pb9'], dtype=object)
```

Figura 11.

	ID	Cm	Per	Def	Fu	EMC-NC
0	Pb1	0.0	0.0	0.0000	-2.590673	0.0
1	Pb1	0.0	0.0	0.0739	6.908462	0.0
2	Pb1	0.0	0.0	0.2217	11.226251	0.0
3	Pb1	0.0	0.0	0.3695	16.839377	0.0
4	Pb1	0.0	0.0	0.5173	22.884282	0.0
5	Pb1	0.0	0.0	0.6651	30.224523	0.0
6	Pb1	0.0	0.0	0.8129	38.860101	0.0
7	Pb1	0.0	0.0	0.9607	47.927458	0.0
8	Pb1	0.0	0.0	1.1085	57.858373	0.0
9	Pb1	0.0	0.0	1.2563	72.538855	0.0
10	Pb1	0.0	0.0	1.4041	86.787559	0.0
11	Pb1	0.0	0.0	1.5519	105.785830	0.0
12	Pb1	0.0	0.0	1.6997	126.511218	0.0
13	Pb1	0.0	0.0	1.8475	141.623479	0.0
14	Pb1	0.0	0.0	1.9953	152.849731	0.0
15	Pb1	0.0	0.0	2.1431	151.554394	1.0
16	Pb1	0.0	0.0	2.2909	126.942997	1.0
17	Pb1	0.0	0.0	2.4387	87.651117	1.0
18	Pb1	0.0	0.0	2.5865	51.381689	1.0
19	Pb1	0.0	0.0	2.7343	32.383417	1.0
20	Pb1	0.0	0.0	2.8821	18.998272	1.0

Figura 13: Características de Entrada y Variable Objetivo de la Probeta 1

Dado que la característica “Fu” es la fuerza que la probeta presenta cuando está siendo comprimida, tiene relación con la variable objetivo, ya que cuando esta empieza a disminuir se entiende que la probeta se rompió, por tanto no se la considerará como variable de entrada en la aplicación de los modelos.

4.2 Determinación de las técnicas de modelado de aprendizaje automático

4.2.1 Modelado

Una vez que la data existente ha superado las etapas de entendimiento y preparación, puede ser usada para modelado de aprendizaje supervisado, ya que fue posible etiquetar las variables de salida, y dado que la variable de salida es categórica y corresponde a las clases “0” y “1” se pueden probar modelos de clasificación para en las etapas de evaluación determinar cuál es el que mejor se ajusta a la data.

Como se mencionó previamente, dado que se cuenta con los datos conocidos de mediciones previas se asume que se cumple con los requisitos para entrenar una máquina de aprendizaje automático supervisado, partiendo de casos particulares para generalizar su funcionamiento, de tal manera que permita predecir valores futuros con valores no conocidos por la máquina.

Si bien es cierto se presume que se podría aplicar aprendizaje supervisado para seleccionar el modelo de aprendizaje de máquina, considerando que la data es de naturaleza categórica, se esperaría que los modelos respondan satisfactoriamente con esta data, y es lo que se va a experimentar con los datos disponibles y modelos de clasificación, de tal manera que la máquina pueda resolver situaciones con datos no conocidos.

Dado que no se requiere de cálculos complejos y con el fin de evitar el depender de infraestructura costosa y gran capacidad de procesamiento se eligieron modelos sencillos de clasificación, de tal manera que incluso pudieran ser implementado en computación de borde.

Se probarán los siguientes modelos de clasificación:

- Vecinos cercanos
- Árboles de decisión
- Random Forest

- Clasificación por Regresión Logística

Dado que se identificaron las características de entrada y la variable objetivo, estas serán asignadas a las variables “X” y “y” respectivamente, tal como se muestra en la

```
X=np.array(lime_sorted[['Cm','Per','Def']])
```

```
X
```

```
array([[0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.0739    ],
       [0.        , 0.        , 0.2217    ],
       ...,
       [1.        , 1.5       , 2.33623367],
       [1.        , 1.5       , 2.48403367],
       [1.        , 1.5       , 2.63183367]])
```

```
y=np.array(lime_sorted['EMC-NC'])
```

```
y
```

```
array([0., 0., 0., ..., 1., 1., 1.])
```

Figura 14.


```
X=np.array(lime_sorted[['Cm','Per','Def']])
```

```
X
```

```
array([[0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.0739    ],
       [0.        , 0.        , 0.2217    ],
       ...,
       [1.        , 1.5       , 2.33623367],
       [1.        , 1.5       , 2.48403367],
       [1.        , 1.5       , 2.63183367]])
```

```
y=np.array(lime_sorted['EMC-NC'])
```

```
y
```

```
array([0., 0., 0., ..., 1., 1., 1.])
```


Figura 14: Características y Variable Objetivo

Para la evaluación de los modelos se trabajará con validación cruzada, separando la data en data de entrenamiento y data de prueba, de tal manera que se pueda probar su eficiencia con un conjunto de datos no conocidos.

Esta separación de datos se mantendrá para todos los modelos seleccionados, de tal manera que los resultados obtenidos sean comparables.

El proceso de separación de la data se muestra en la

```
# Separar la data
from sklearn.model_selection import train_test_split
```

```
# Separar la data en datos de entrenamiento y datos de prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
print(X_train)
```

```
[[0.5        1.         0.6651    ]
 [1.         1.5        3.4733    ]
 [1.5        1.5        0.        ]
 ...
 [0.5        1.5        2.1643    ]
 [1.5        2.         0.09791574]
 [1.5        2.         0.0357586 ]]
```

Figura 15: Separación de la data de entrenamiento y prueba


```
# Separar la data
from sklearn.model_selection import train_test_split
```

```
# Separar la data en datos de entrenamiento y datos de prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
print(X_train)
```

```
[[0.5        1.         0.6651    ]
 [1.         1.5        3.4733    ]
 [1.5        1.5        0.        ]
 ...
 [0.5        1.5        2.1643    ]
 [1.5        2.         0.09791574]
 [1.5        2.         0.0357586 ]]
```


Figura 15: Separación de la data de entrenamiento y prueba

Una vez que se cuenta con el conjunto de datos de entrenamiento y prueba se procede a entrenar la máquina con los modelos seleccionados.

4.2.1.1 Modelado con vecinos cercanos

Para entrenar el modelo con vecinos cercanos se debe encontrar el hiperparámetro del número de vecinos cercanos que mejor se ajuste a los datos, en este caso se probó con 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,13, 14 vecinos cercanos para encontrar el que entregue la mejor

respuesta. Las siguientes capturas muestran los resultados entre 8 y 12 vecinos cercanos que es donde ocurre el punto de inflexión.

La creación y entrenamiento de la máquina con 6 vecinos cercanos se muestra en la

```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=8)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[347  21]
 [ 36  78]]
Accuracy: 88.17%
```

Figura 18, donde se puede apreciar una precisión del 86,31%

```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=6)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[343  25]
 [ 41  73]]
Accuracy: 86.31%
```

Figura 16: Modelado KNN con 6 vecinos cercanos

La creación y entrenamiento de la máquina con 7 vecinos cercanos se muestra en la Figura 17, donde se puede apreciar una precisión del 87,14%

```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=7)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[337  31]
 [ 31  83]]
Accuracy: 87.14%
```

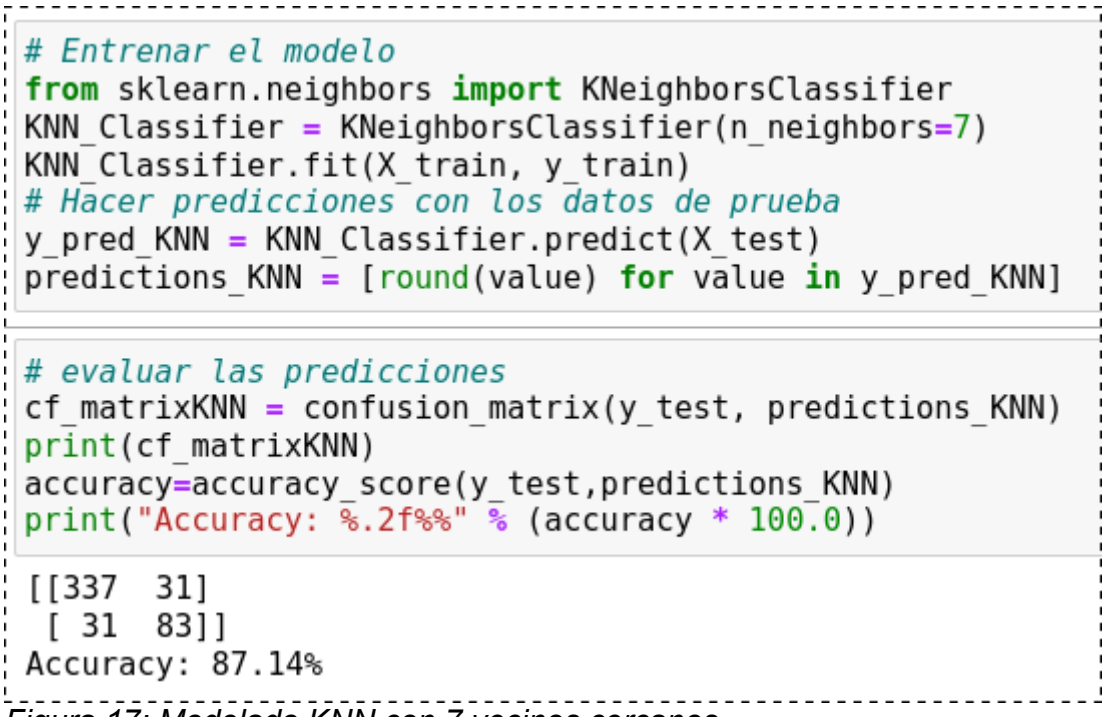


Figura 17: Modelado KNN con 7 vecinos cercanos

La creación y entrenamiento de la máquina con 8 vecinos cercanos se muestra en la

```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=8)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[347  21]
 [ 36  78]]
Accuracy: 88.17%
```

Figura 18, donde se puede apreciar una precisión del 88,17%


```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=8)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[347  21]
 [ 36  78]]
Accuracy: 88.17%
```


Figura 18: Modelado KNN con 8 vecinos cercanos

La creación y entrenamiento de la máquina con 9 vecinos cercanos se muestra en la

```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=9)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[339  29]
 [ 32  82]]
Accuracy: 87.34%
```

Figura 19, donde se puede apreciar una precisión del 87,34%


```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=9)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[339  29]
 [ 32  82]]
Accuracy: 87.34%
```


Figura 19: Modelado KNN con 9 vecinos cercanos

La creación y entrenamiento de la máquina con 10 vecinos cercanos se muestra en la Figura 20, donde se puede apreciar una precisión del 87,55%

```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=10)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[345  23]
 [ 37  77]]
Accuracy: 87.55%
```


```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=10)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[345  23]
 [ 37  77]]
Accuracy: 87.55%
```


Figura 20: Modelado KNN con 10 vecinos cercanos

La creación y entrenamiento de la máquina con 11 vecinos cercanos se muestra en la

```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=11)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[339  29]
 [ 31  83]]
Accuracy: 87.55%
```

Figura 21, donde se puede apreciar una precisión del 87,55%


```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=11)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[339  29]
 [ 31  83]]
Accuracy: 87.55%
```


Figura 21: Modelado KNN con 11 vecinos cercanos

La creación y entrenamiento de la máquina con 12 vecinos cercanos se muestra en la

```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=12)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[344  24]
 [ 41  73]]
Accuracy: 86.51%
```

Figura 22, donde se puede apreciar una precisión del 86,51%


```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=12)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[344  24]
 [ 41  73]]
Accuracy: 86.51%
```


Figura 22: Modelado KNN con 12 vecinos cercanos

La creación y entrenamiento de la máquina con 13 vecinos cercanos se muestra en la

```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=13)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[341  27]
 [ 34  80]]
Accuracy: 87.34%
```

Figura 23, donde se puede apreciar una precisión del 87,34%


```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=13)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[341  27]
 [ 34  80]]
Accuracy: 87.34%
```


Figura 23: Modelado KNN con 13 vecinos cercanos

Tabla 4: Precisión KNN en función de los vecinos cercanos

Número de vecinos cercanos	Precisión [%]
6	86,31
7	87,14

8	88,17
9	87,34
10	87,55
11	87,55

En la Tabla 4 se puede apreciar que el modelo responde mejor con 8 vecinos cercanos, por tanto se selecciona este valor para el hiperparámetro de este modelo.

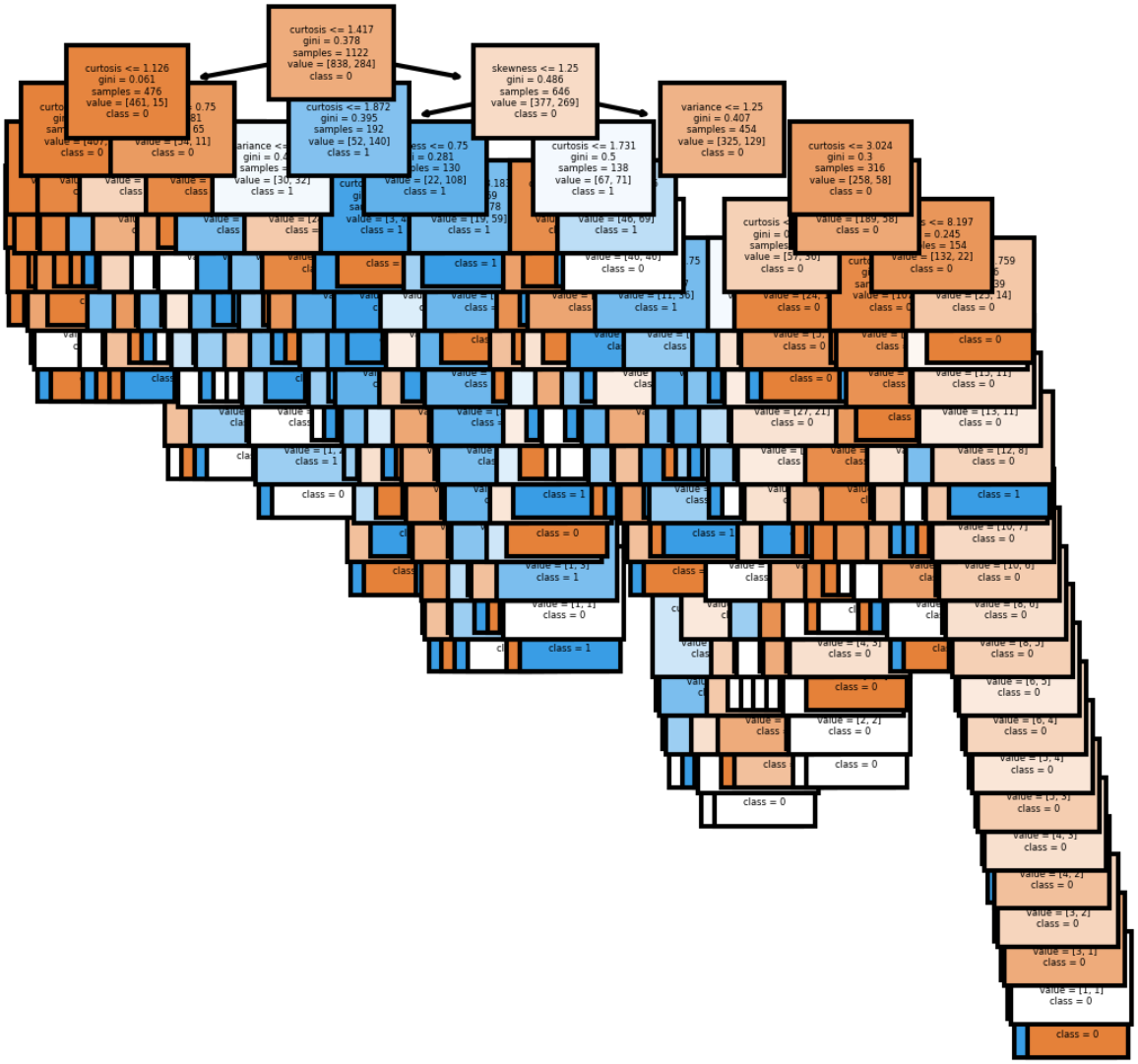
4.2.1.2 Modelado con árboles de decisión

Para el modelado con árboles de decisión se puede ajustar el hiperparámetro de la profundidad del árbol y de no hacerlo, el modelo avanzará hasta que no sea posible tener mas bifurcaciones.

En este caso y para el valor de la mayor precisión posible, no se ha truncado la profundidad del árbol, tal como se muestra en la Figura 24.

```
# Entrenar el modelo
from sklearn.tree import DecisionTreeClassifier
DT_Classifier = DecisionTreeClassifier()
DT_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_DT = DT_Classifier.predict(X_test)
predictions_DT = [round(value) for value in y_pred_DT]
```

Figura 24: Modelado con árboles de decisión



La

Figura 25 muestra de manera ilustrativa como el modelo va realizando la clasificación. Dada la dimensión del gráfico no es posible apreciar todos los ramales, sin embargo, sirve como referencia del proceso de clasificación

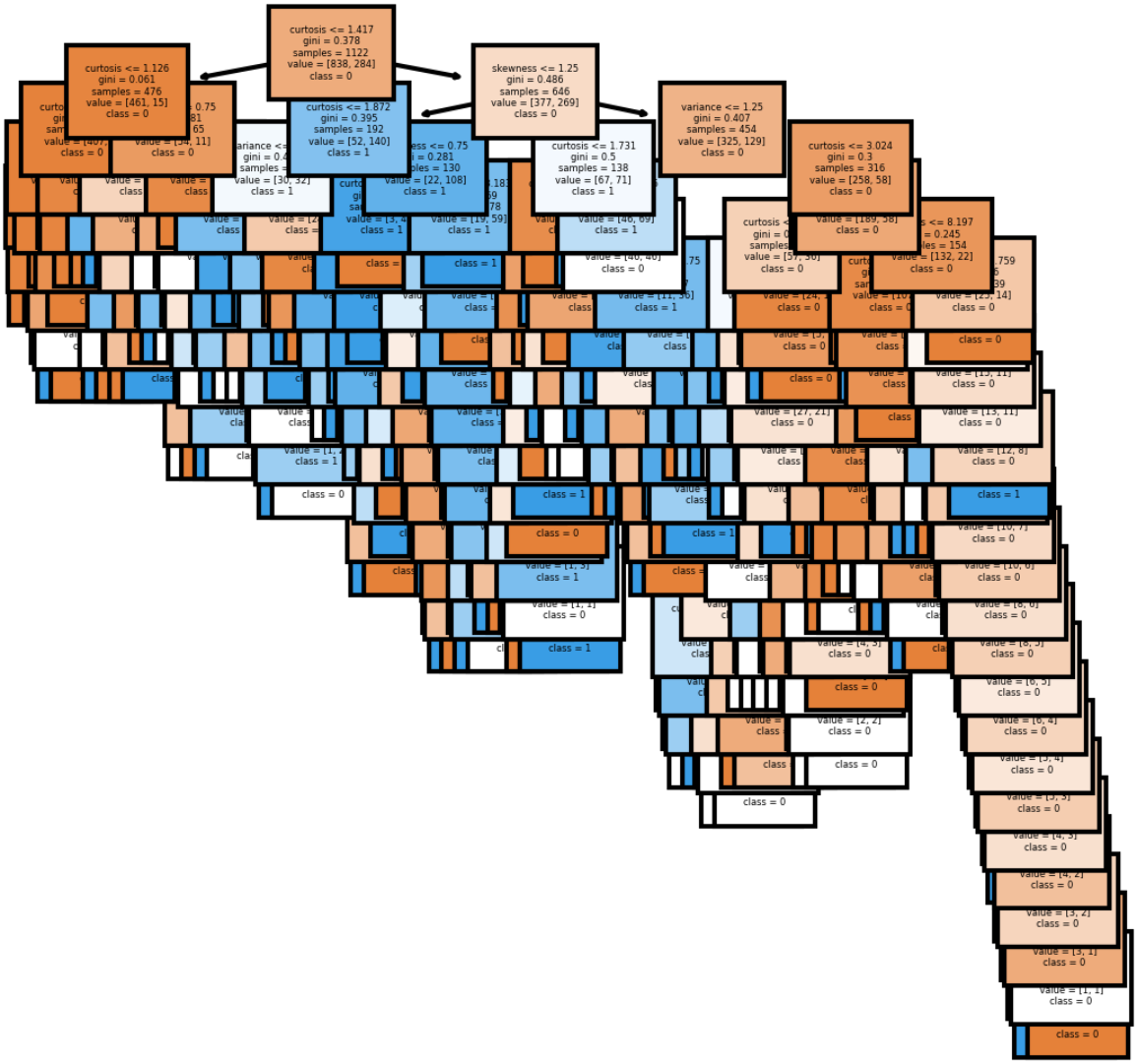


Figura 25: Gráfica del modelado de árboles de decisión

4.2.1.3 Modelado con Random Forest

En el modelo de Random Forest, el hiperparámetro “n_estimators” corresponde al número de árboles que desea construir antes de tomar la máxima ponderación o promedio de las predicciones, en este caso se probaron distintos valores y se obtuvo el mismo resultado, por lo que se lo definió en 10. La creación de la máquina con Random Forest y su modelado se muestra en la

```
# Entrenar el modelo
from sklearn.ensemble import RandomForestRegressor
RF_Classifier = RandomForestRegressor(n_estimators=10, random_state=0)
RF_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_RF = RF_Classifier.predict(X_test)
predictions_RF = [round(value) for value in y_pred]
```

Figura 26.


```
# Entrenar el modelo
from sklearn.ensemble import RandomForestRegressor
RF_Classifier = RandomForestRegressor(n_estimators=10, random_state=0)
RF_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_RF = RF_Classifier.predict(X_test)
predictions_RF = [round(value) for value in y_pred]
```


Figura 26: Modelado con Random Forest

4.2.1.4 Modelado de clasificación por Regresión Logística

Dado en este caso se tienen dos clases de salida, su puede usar también el modelo de regresión logística cuya creación y entrenamiento se muestra en la

```
# Entrenar el modelo
from sklearn import linear_model
LR_Classifier = linear_model.LogisticRegression()
LR_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_LR = LR_Classifier.predict(X_test)
predictions_LR = [round(value) for value in y_pred_LR]
```

Figura 27


```
# Entrenar el modelo
from sklearn import linear_model
LR_Classifier = linear_model.LogisticRegression()
LR_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_LR = LR_Classifier.predict(X_test)
predictions_LR = [round(value) for value in y_pred_LR]
```


Figura 27: Modelado con Regresión Logística

Una vez que se han creado y entrenado los distintos modelos de clasificación se los puede evaluar para seleccionar el mejor de ellos.

4.3 Evaluación de modelos que cumplen con los objetivos de la data

Como premisa se partió con el hecho de que una vez que se hayan seleccionado y entrenado los modelos, estos deben ser evaluados de tal forma que se pueda seleccionar el modelo de aprendizaje de máquina que mejor se ajuste a la data.

Se comparará la precisión obtenida por cada uno de ellos y su matriz de confusión.

4.3.1 Evaluación del modelo vecinos cercanos KNN

En el caso de KNN el hiperparámetro de vecinos cercanos presento su mejor desempeño con el valor de 8, el mismo que será usado para comprar este modelo con los otros, el resultado de la evaluación de este modelo se muestra en la

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy_KNN=accuracy_score(y_test,predictions_KNN)
print("Accuracy_KNN: %.2f%%" % (accuracy_KNN * 100.0))
```

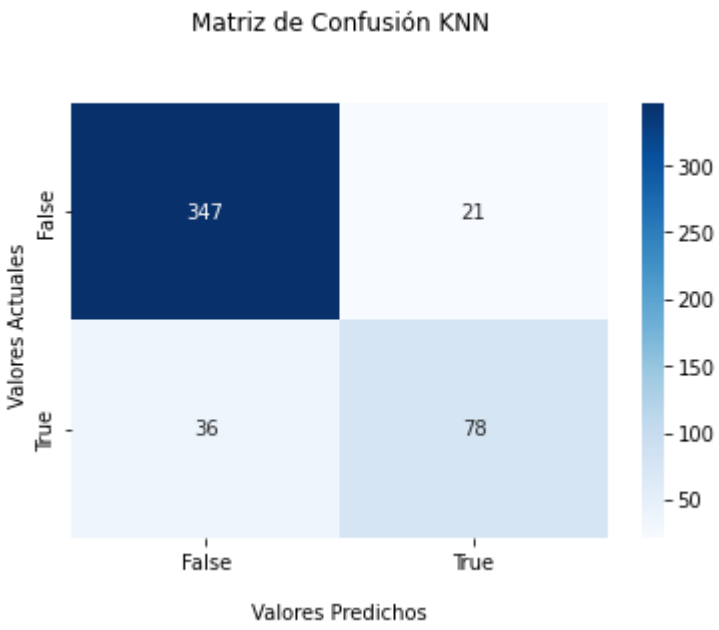
```
[[347  21]
 [ 36  78]]
Accuracy_KNN: 88.17%
```

Figura 28.


```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy_KNN=accuracy_score(y_test,predictions_KNN)
print("Accuracy_KNN: %.2f%%" % (accuracy_KNN * 100.0))
```

```
[[347  21]
 [ 36  78]]
Accuracy_KNN: 88.17%
```


Figura 28: Evaluación del modelo KNN



La Figura 29 muestra la matriz de confusión de este modelo

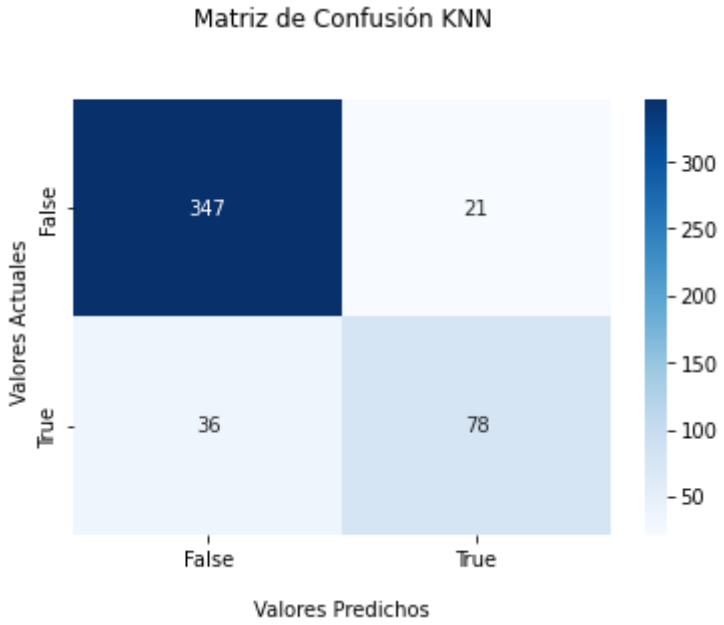


Figura 29: Matriz de Confusión del modelo KNN

4.3.2 Evaluación del modelo Árboles de Decisión

En el caso de árboles de decisión y con el fin de conseguir la mayor precisión posible no se ajustó el hiperparámetro de profundidad del árbol, el resultado de la evaluación de este

modelo se muestra en la

```
# Hacer predicciones con los datos de prueba
cf_matrixDT = confusion_matrix(y_test, predictions_DT)
print(cf_matrixDT)
accuracy_DT=accuracy_score(y_test,predictions_DT)
print("Accuracy: %.2f%%" % (accuracy_DT * 100.0))
```

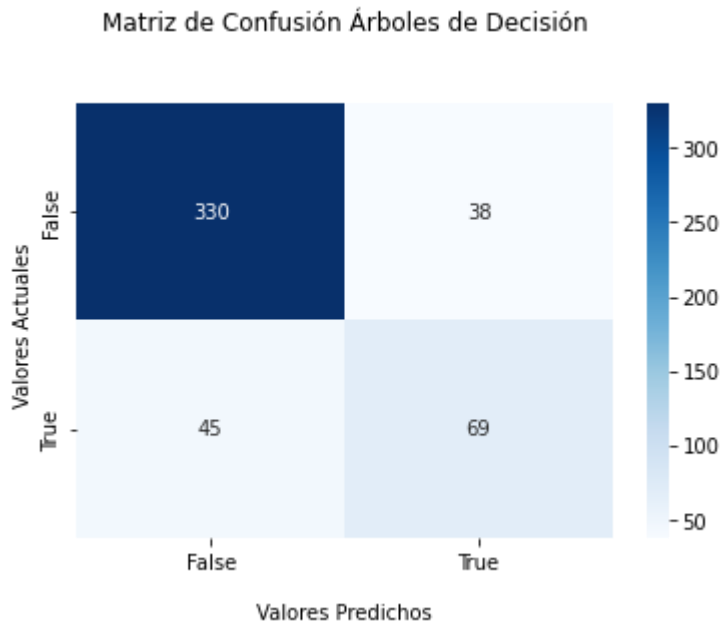
```
[[330  38]
 [ 45  69]]
Accuracy: 82.78%
```

Figura 30.


```
# Hacer predicciones con los datos de prueba
cf_matrixDT = confusion_matrix(y_test, predictions_DT)
print(cf_matrixDT)
accuracy_DT=accuracy_score(y_test,predictions_DT)
print("Accuracy: %.2f%%" % (accuracy_DT * 100.0))
```

```
[[330  38]
 [ 45  69]]
Accuracy: 82.78%
```


Figura 30: Evaluación del modelo Árboles de Decisión



La Figura 31 muestra la matriz de confusión de este modelo

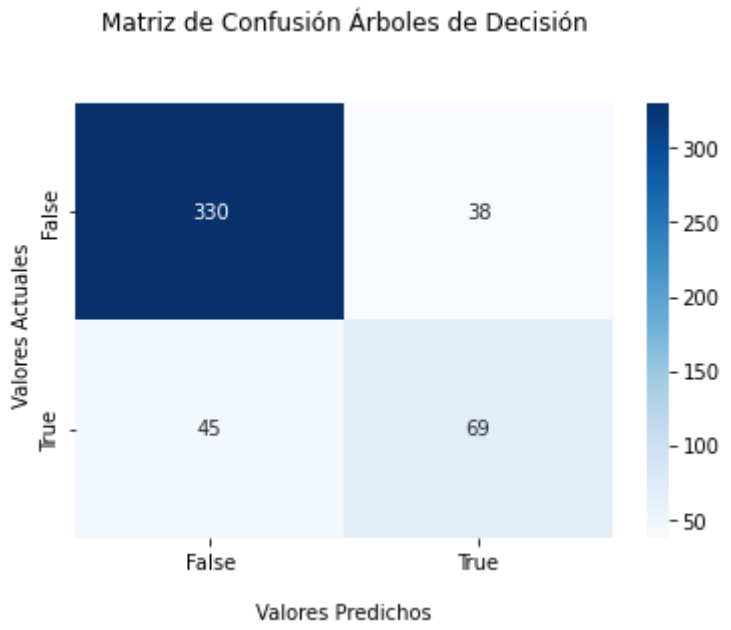


Figura 31: Matriz de Confusión del modelo Árboles de Decisión

4.3.3 Evaluación del modelo Random Forest

En el caso de Random Forest el hiperparámetro “n_estimators” se lo definió en 10 ya que no existió impacto al cambiarlo por otro, el resultado de la evaluación de este modelo se muestra en la Figura 32.

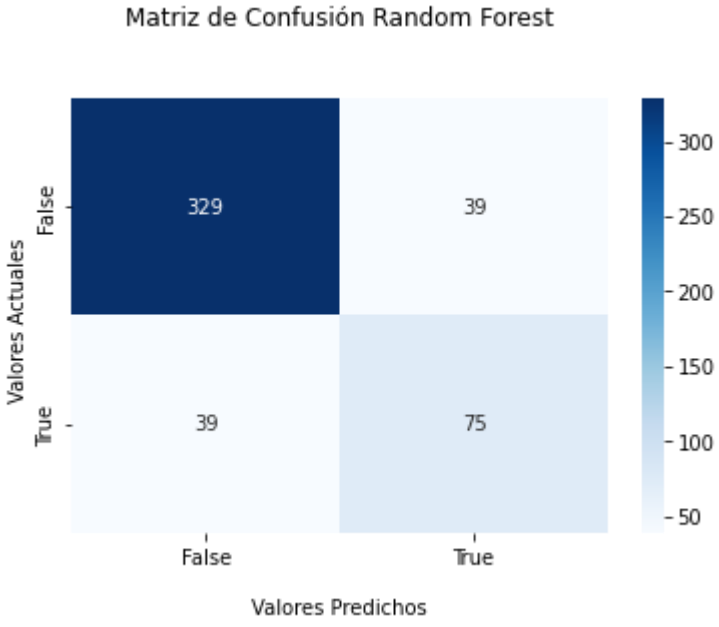
```
# Evaluar las predicciones
cf_matrixRF = confusion_matrix(y_test, predictions_RF)
print(cf_matrixRF)
accuracy_RF=accuracy_score(y_test,predictions_RF)
print("Accuracy: %.2f%%" % (accuracy_RF * 100.0))
```

```
[[329  39]
 [ 39  75]]
Accuracy: 83.82%
```


```
# Evaluar las predicciones
cf_matrixRF = confusion_matrix(y_test, predictions_RF)
print(cf_matrixRF)
accuracy_RF=accuracy_score(y_test,predictions_RF)
print("Accuracy: %.2f%%" % (accuracy_RF * 100.0))
```

```
[[329  39]
 [ 39  75]]
Accuracy: 83.82%
```


Figura 32: Evaluación del modelo Random Forest



La Figura 33 muestra la matriz de confusión de este modelo

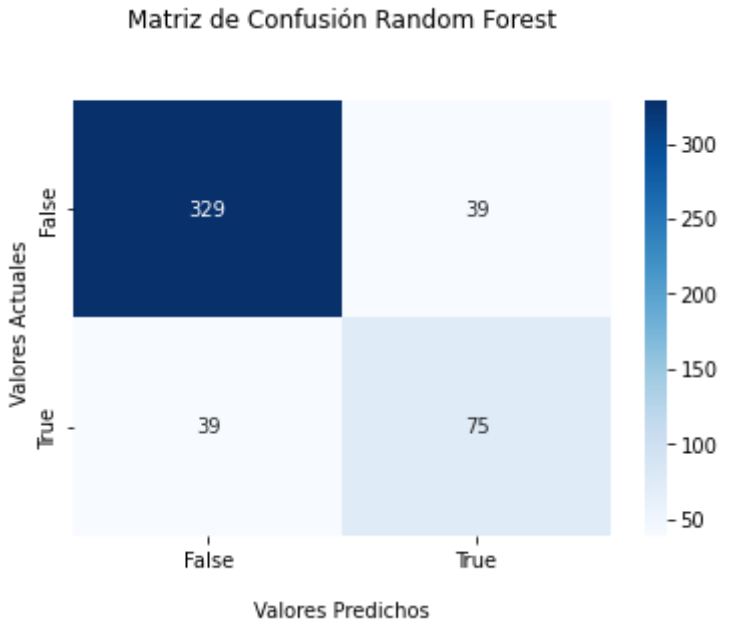


Figura 33: Matriz de Confusión del modelo Random Forest

4.3.4 Evaluación del modelo de Clasificación por Regresión Logística

El resultado de la evaluación de este modelo se muestra en la

```
# Evaluar las predicciones
cf_matrixLR = confusion_matrix(y_test, predictions_LR)
print(cf_matrixLR)
accuracy_LR=accuracy_score(y_test,predictions_LR)
print("Accuracy: %.2f%%" % (accuracy_LR * 100.0))
```

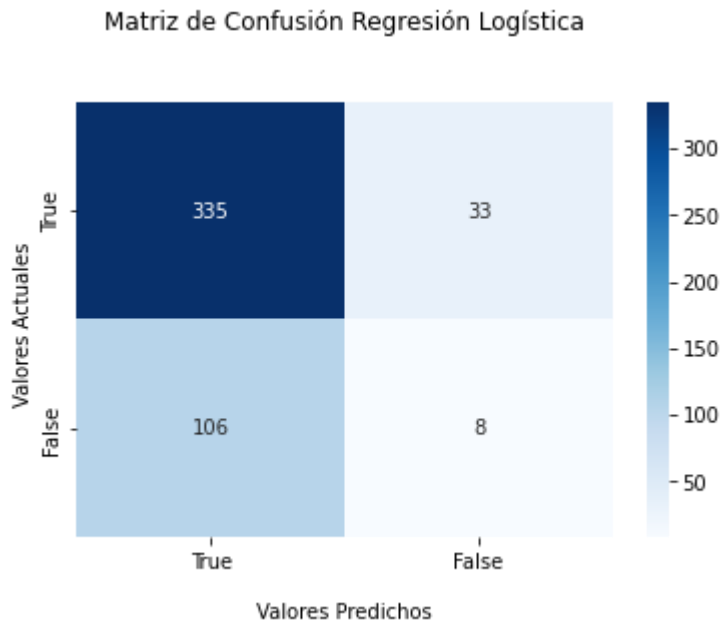
```
[[335  33]
 [106   8]]
Accuracy: 71.16%
```

Figura 34.


```
# Evaluar las predicciones
cf_matrixLR = confusion_matrix(y_test, predictions_LR)
print(cf_matrixLR)
accuracy_LR=accuracy_score(y_test,predictions_LR)
print("Accuracy: %.2f%%" % (accuracy_LR * 100.0))
```

```
[[335  33]
 [106   8]]
Accuracy: 71.16%
```


Figura 34: Evaluación del modelo Regresión Logística



La Figura 35 muestra la matriz de confusión de este modelo

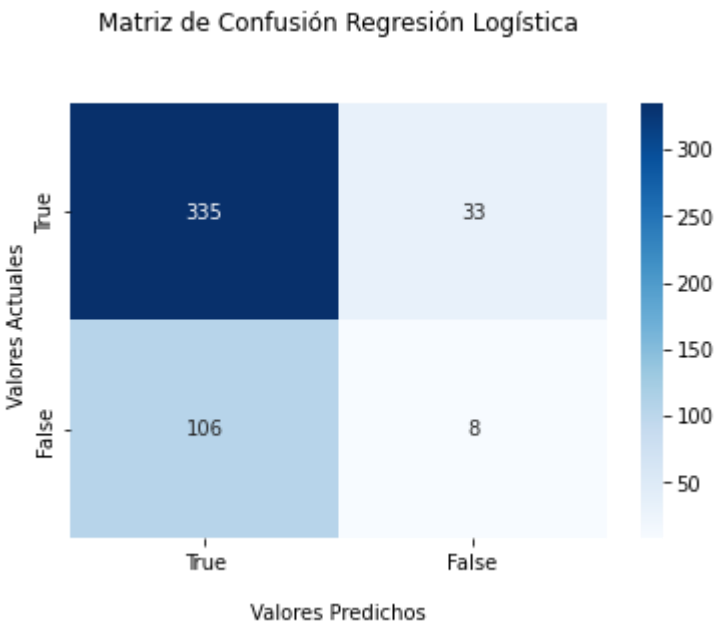


Figura 35: Matriz de Confusión del modelo Regresión Logística

4.3.5 Comparativa y selección de los modelos

La Tabla 5 muestra la precisión de cada uno de los modelos aplicados y en ella se puede apreciar que el modelo que mejor responde al conjunto de datos es KNN con una ligera diferencia con los modelos que le siguen, a excepción del de regresión logística, por lo que es el modelo seleccionado.

Tabla 5: Tabla comparativa de los modelos aplicados, primera corrida

Modelo	**Precisión**
KNN	87,55
Árboles de Decisión	82,7
Random Forest	83,82
Regresión Logística	71,16

En una segunda y tercera corrida, cuyos resultados se muestran en la Tabla 6, los resultados son similares y coherentes, por lo que se puede concluir que el modelo KNN es el que mejor responde a este conjunto de datos.

Tabla 6: Tabla comparativa de los modelos aplicados, segunda corrida

Modelo	**Precisión Corrida 2**	**Precisión Corrida 3**
KNN	86,51	84,85
Árboles de Decisión	82,99	80,49
Random Forest	82,78	64,93
Regresión Logística	70,75	70,53

4.4 Despliegue

Dado que estas modelaciones están orientadas a la investigación y se requiere cumplir criterios de replicabilidad, se recomienda que el despliegue se realice como cuadernos de trabajo en los que se detalle claramente las celdas explicativas y de código durante todo el proceso de aprendizaje de máquina.

Como anexo a este documento se adjunta el código exportado a formato .py desde el cuaderno de trabajo jupyter.

Por el momento no se incluyen los datos ya que pertenecen al “Laboratorio de Resistencia de Materiales, Mecánica de Suelos, Pavimentos y Geotécnica de la Pontificia Universidad Católica del Ecuador” (PUCE, 2022)

5. Conclusiones y Recomendaciones

5.1 Conclusiones

- Es posible usar de manera satisfactoria el modelo CRISP-DM para predecir la influencia de la inclusión de fibras de abacá en la resistencia a la compresión simple de especímenes reconstituidos de limos arenosos a través de aprendizaje automático usando el conjunto de datos resultado de ensayos del LMC.
- Se aplicó la metodología CRISP-DM para entender y preparar la data para aprendizaje automático donde se identificó las características y la variable objetivo.
- Se probaron satisfactoriamente varias técnicas de modelado de aprendizaje automático para predecir la deformación por resistencia a la compresión no confinada de limos arenosos reconstituidos debido a la influencia de la inclusión de fibra de abacá.
- Dada la naturaleza de las características y variable objetivo se determinó que el modelo a aplicarse es aprendizaje automático supervisado de clasificación, esto debido a que la variable objetivo seleccionada es categórica booleana.
- Al evaluar los modelos con los objetivos de la data se encontró que el que mejor se ajusta a los datos es KNN seguido de cerca por árboles de decisión, mientras que regresión logística y random forest mostraron, en l menos una de las corridas resultados de menor precisión.

5.2 Recomendaciones

- Con el conjunto de datos existentes se pueden realizar otros tipos de modelado cambiando el enfoque de las variables predictoras y objetivo de tal forma que el modelo cambie a regresión o agrupamiento.
- Desde el punto de vista matemático se podrían realizar pruebas de modelos de regresión polinómicos considerando la variable ‘Fu’ como variable de salida considerando las curvas tensión-deformación con variaciones del contenido de fibra y la longitud de la fibra.
- De manera didáctica se podrían probar modelos de aprendizaje profundo con redes neuronales ya que, aunque en la práctica consumirían recursos de procesamiento que podrían ser escasos, si se considera computación de borde, y que podrían aportar a un mejor entendimiento de los modelos de aprendizaje automático.
- Como ejercicio, para el caso de agrupación se pueden ignorar las etiquetas y analizar si el agrupamiento en dos grupos coincide con las clases de las etiquetas ignoradas.
- Dado que al barajar los datos en entrenamiento y prueba los resultados varían, se recomienda realizar varias corridas para verificar que los resultados sean coherentes.

Bibliografía

[1] CRoss Industry Standard Process for Data Mining (CRISP-DM), Data Science Process Alliance (2021)

[2] Pontificia Universidad Católica del Ecuador, (2022). Laboratorio de Resistencia de Materiales, Mecánica de Suelos, Pavimentos y Geotécnica https://www.puce.edu.ec/servicios/laboratorio-de-suelos/

[3] Dominios Académicos y Líneas de Investigación, PUCE, 2017

[4] Albuja-Sánchez, J.; Alcívar, E.; Escobar, D.; Montero, J.; Realpe, G.; Muñoz, A.; Peñaherrera-Aguirre, M. Influence of Abaca Fiber Inclusion on the Unconfined Compressive Keywords: simple compression test; unconfined compression test; abaca fiber; soil improvement; fiber-reinforced soil Strength of Reconstituted Sandy Silts. Fibers 2022, 10, 99. https://doi.org/10.3390/fib10110099

[5] 103 instancias, Yeh, I-Cheng, "Modeling slump flow of concrete using second-order regressions and artificial neural networks," Cement and Concrete Composites, Vol.29, No. 6, 474-480, 2007.

[6] Jiawei Han, Data Mining: Concepts and Techniques Second Edition, University of Illinois at Urbana-Champaign, 2006

Anexos

Código Jupyter Notebook exportado como código .py

```
#!/usr/bin/env python

# coding: utf-8


# ## MINERÍA DE DATOS PARA EL MODELADO DE APRENDIZAJE AUTOMÁTICO EN LA RESISTENCIA DE MATERIALES POR LA INCLUSIÓN DE FIBRAS NATURALES.

#

# ### CASO DE ESTUDIO:  INFLUENCIA DE LA INCLUSIÓN DE FIBRA DE ABACÁ EN LA RESISTENCIA A LA COMPRESIÓN DE LIMOS ARENOSOS, LABORATORIO DE, MECÁNICA DE SUELOS, PAVIMENTOS Y GEOTÉCNICA DE LA PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

#

# ## 4.1.2 Comprensión de datos

# In[1]:


import pandas as pd

lime=pd.read_csv("Lime.csv",sep=",")


# In[2]:


lime.shape


# In[3]:


lime.head(10)


# In[4]:


lime.columns


# In[5]:


#Mostrar todas las columnas

pd.options.display.max_columns = None
```

In[6]:

# Estadísticos
lime.describe()

In[7]:

#Mostrar todas las filas
pd.options.display.max_rows = None

In[8]:

lime

4.1.3 Preparación de datos
In[9]:

lime_proc=lime[['ID','Cm','Per','Def','Fu']]

In[10]:

lime_proc.head(10)

In[11]:

lime_sorted=lime_proc.sort_values(['ID','Cm','Def'], ascending=True,ignore_index=True)

In[12]:

lime_sorted.head(20)

In[13]:

lime_sorted.to_csv('lime_proc.csv')

```
# In[14]:

lime_sorted[:2]['Cm']

# In[15]:

lime_sorted.iloc[0]

# In[16]:

lime_sorted.iloc[1]

# In[17]:

lime_sorted.iloc[1]['Fu']

# In[18]:


lime_sorted['ID'].unique()

# In[19]:

len(lime_sorted)

# In[121]:

Probeta=lime_sorted.iloc[0]['ID']
for i in range(len(lime_sorted)):
    if Probeta==lime_sorted.iloc[i]['ID']:
        if (lime_sorted.iloc[i]['Fu'] > lime_sorted.iloc[i-1]['Fu']) or (i==0) :
            #print(i," ",lime_sorted.iloc[i]['ID']," ",lime_sorted.iloc[i]['Fu']," ",0)
            lime_sorted.loc[i,'EMC-NC']=0
        else:
            #print(i," ",lime_sorted.iloc[i]['ID']," ",lime_sorted.iloc[i]['Fu']," ",1)
```

```
            lime_sorted.loc[i,'EMC-NC']=1
    else:
        Probeta=lime_sorted.iloc[i]['ID']
        #print(i," ",lime_sorted.iloc[i]['ID']," ",lime_sorted.iloc[i]['Fu']," ",0)
        lime_sorted.loc[i,'EMC-NC']=0

# In[21]:

lime_sorted.head(21)



# In[22]:
lime_sorted.iloc[0]

# In[23]:

lime_sorted.iloc[0]._is_copy

# In[24]:

lime.iloc[0]._is_copy

# ## 4.2 Determinación de las técnicas de modelado de aprendizaje automático

# ### Asignación de las características y variable predictora objetivo

# In[25]:

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# In[63]:

X=np.array(lime_sorted[['Cm','Per','Def']])
```

In[64]:

X

In[65]:

y=np.array(lime_sorted['EMC-NC'])

In[66]:

y

Separación de la data en datos de prueba y entrenamiento

In[142]:

# Separar la data
from sklearn.model_selection import train_test_split

In[191]:

Separar la data en datos de entrenamiento y datos de prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
print(X_train)

4.2.1.1 Modelado con vecinos cercanos
In[192]:

# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=2)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]

In[193]:

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy_KNN=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy_KNN * 100.0))
```

In[194]:

```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=3)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

In[195]:

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

In[196]:

```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
```

```
KNN_Classifier = KNeighborsClassifier(n_neighbors=4)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]

# In[197]:

# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))


# In[198]:


# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=5)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]


# In[199]:


# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

In[200]:

```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=6)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

In[201]:

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

In[202]:

```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=7)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

In[203]:

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))


# In[204]:


# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=8)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]


# In[205]:


# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))


# In[206]:


# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
```

```
KNN_Classifier = KNeighborsClassifier(n_neighbors=9)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

```
# In[207]:
```

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
# In[208]:
```

```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=10)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

```
# In[209]:
```

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
```

```
print("Accuracy: %.2f%%" % (accuracy * 100.0))


# In[210]:


# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=11)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]


# In[211]:


# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))


# In[212]:


# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=12)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

In[213]:

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

In[214]:

```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=13)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]
```

In[215]:

```
# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

In[216]:

```
# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=14)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]


# In[217]:


# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
print(cf_matrixKNN)
accuracy=accuracy_score(y_test,predictions_KNN)
print("Accuracy: %.2f%%" % (accuracy * 100.0))


# In[218]:


# Entrenar el modelo
from sklearn.neighbors import KNeighborsClassifier
KNN_Classifier = KNeighborsClassifier(n_neighbors=7)
KNN_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_KNN = KNN_Classifier.predict(X_test)
predictions_KNN = [round(value) for value in y_pred_KNN]


# In[219]:


# evaluar las predicciones
cf_matrixKNN = confusion_matrix(y_test, predictions_KNN)
```

```
print(cf_matrixKNN)
accuracy_KNN=accuracy_score(y_test,predictions_KNN)
print("Accuracy_KNN: %.2f%%" % (accuracy_KNN * 100.0))

# In[224]:

ax = sns.heatmap(cf_matrixKNN, annot=True, cmap='Blues',fmt='g')

ax.set_title('Matriz de Confusión KNN\n\n');
ax.set_xlabel('\nValores Predichos')
ax.set_ylabel('Valores Actuales ');

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])

## Display the visualization of the Confusion Matrix.
plt.show()


# ### 4.2.1.2 Modelado con árboles de decisión

# In[220]:

# Entrenar el modelo
from sklearn.tree import DecisionTreeClassifier
DT_Classifier = DecisionTreeClassifier()
DT_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_DT = DT_Classifier.predict(X_test)
predictions_DT = [round(value) for value in y_pred_DT]

# In[221]:

from sklearn import tree
text_representation = tree.export_text(DT_Classifier)
```

```
#print(text_representation)

# In[123]:

#tree.plot_tree(DT_Classifier, fontsize=10)

# In[104]:

fn=['variance','skewness','curtosis','entropy']
cn=['0','1']
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)
tree.plot_tree(DT_Classifier,
               feature_names = fn,
               class_names=cn,
               filled = True,fontsize=2)
plt.savefig('tree_high_dpi', dpi=200)

# In[222]:

# Hacer predicciones con los datos de prueba
cf_matrixDT = confusion_matrix(y_test, predictions_DT)
print(cf_matrixDT)
accuracy_DT=accuracy_score(y_test,predictions_DT)
print("Accuracy: %.2f%%" % (accuracy_DT * 100.0))

# In[223]:

ax = sns.heatmap(cf_matrixDT, annot=True, cmap='Blues',fmt='g')

ax.set_title('Matriz de Confusión Árboles de Decisión\n\n');
ax.set_xlabel('\nValores Predichos')
ax.set_ylabel('Valores Actuales ');

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])
```

```
## Display the visualization of the Confusion Matrix.
plt.show()

# ### 4.2.1.3 Modelado con Random Forest

# In[183]:

# Entrenar el modelo
from sklearn.ensemble import RandomForestRegressor
RF_Classifier = RandomForestRegressor(n_estimators=20, random_state=0)
RF_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_RF = RF_Classifier.predict(X_test)
predictions_RF = [round(value) for value in y_pred_RF]

# In[225]:

# Evaluar las predicciones
cf_matrixRF = confusion_matrix(y_test, predictions_RF)
print(cf_matrixRF)
accuracy_RF=accuracy_score(y_test,predictions_RF)
print("Accuracy: %.2f%%" % (accuracy_RF * 100.0))

# In[226]:

ax = sns.heatmap(cf_matrixRF, annot=True, cmap='Blues',fmt='g')

ax.set_title('Matriz de Confusión Random Forest\n\n');
ax.set_xlabel('\nValores Predichos')
ax.set_ylabel('Valores Actuales ');

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])
```

```
## Display the visualization of the Confusion Matrix.
plt.show()

# ### 4.2.1.4 Modelado con clasificación por regresión logística

# In[227]:

# Entrenar el modelo
from sklearn import linear_model
LR_Classifier = linear_model.LogisticRegression()
LR_Classifier.fit(X_train, y_train)
# Hacer predicciones con los datos de prueba
y_pred_LR = LR_Classifier.predict(X_test)
predictions_LR = [round(value) for value in y_pred_LR]

# In[228]:

# Evaluar las predicciones
cf_matrixLR = confusion_matrix(y_test, predictions_LR)
print(cf_matrixLR)
accuracy_LR=accuracy_score(y_test,predictions_LR)
print("Accuracy: %.2f%%" % (accuracy_LR * 100.0))

# In[229]:

ax = sns.heatmap(cf_matrixLR, annot=True, cmap='Blues',fmt='g')

ax.set_title('Matriz de Confusión Regresión Logística\n\n');
ax.set_xlabel('\nValores Predichos')
ax.set_ylabel('Valores Actuales ');

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['True','False'])
ax.yaxis.set_ticklabels(['True','False'])

## Display the visualization of the Confusion Matrix.
```

```
plt.show()

# In[230]:

Accuracy_List=[accuracy_KNN*100,accuracy_DT*100,accuracy_RF*100,accuracy_LR*100]
Accuracy_Names=['KNN','Árboles de Decisión','Random Forest','Regresión Logística']

# In[231]:

Acc=pd.DataFrame(Accuracy_List,Accuracy_Names)
Acc
```