Universidad Nacional Autónoma de México

Facultad de Ingeniería
Ingeniería Geofísica

Machine learning aplicado a registros geofísicos de pozos para la clasificación de litofacies

T E S I S

que para obtener el título de:

Ingeniera Geofísica

presenta:

Lourdes Michelle Cruz Natalio

Director de Tesis

MTRO. Ernesto Cisneros Vega

Ciudad Universitaria,CDMX, 2024

Resumen

En el presente trabajo plantea resolver a través de técnicas de Machine Learning (ML), la clasificación de facies litológicas utilizando datos de registros geofísicos de pozos. ML es una disciplina del campo de la Inteligencia Artificial que, a través de algoritmos, dota a las computadoras de la capacidad de identificar patrones en datos masivos y elaborar predicciones (análisis predictivo). Los métodos aplicados fueron: Support Vector Machine, Random Forest, K-Nearest Neighbor y Extreme Gradient Boost, todos ellos bajo el esquema de aprendizaje supervisado.

La importancia del reconocimiento y clasificación de facies reside en que permiten evaluar las características de la roca. En la industria petrolera el análisis de facies puede llevar a predicciones, para encontrar la mejor calidad de roca de los yacimientos petróleo y gas.

El proyecto incluyó la programación en un notebook de Python de los algoritmos de ML, para evaluar sus capacidades en la clasificación de las facies, comparando y evaluando los resultados de cada método, para tener un mejor criterio de aplicación de esta tecnología en datos reales.

Se diseñó un flujo de trabajo para la secuencia de procesamiento de los datos, el cual incluyó desde el acondicionamiento, hasta la evaluación de resultados. El método desarrollado se aplicó a datos del Golfo de México, en la Provincia Petrolera del Sureste, utilizando registros geofísicos de pozos marinos, considerando solamente la zona de interés que abarca: Cretácico, Jurásico Superior Tithoniano y Jurásico Superior Kimmerdigiano, donde se han desarrollado facies de bancos oolíticos, que constituyen la roca almacenadora de los yacimientos de aceite y gas.

La clasificación con datos geológicos reales puede presentar distintos problemas como la falta de datos, el ruido en los datos, el desbalance entre clases, todo esto afecta y dificulta la categorización de facies. Esta problemática se manejó aplicando diferentes técnicas para tratar estos problemas, para buscar una mejor interpretabilidad y sensibilidad a la presencia de datos dispersos y desbalanceados. Encontrando que Support Vector Machine (SVM), puede ser menos interpretable, debido a la complejidad del hiperplano, puede sobreajustarse fácilmente si no se trata el desbalance de clases. K-Nearest Neighbor (KNN), relativamente fácil de entender ya que clasifica en función de la proximidad a casos existentes, pero es sensible al desbalance de clases ya que la clase mayoritaria puede sesgar el modelo. Random Forest (RF), difícil de interpretar como trabaja debido a la combinación de múltiples árboles de decisión, puede manejar datos desbalanceados mediante técnicas como la ponderación de clases. Extreme Gradient Boosting (XGB), la interpretabilidad puede ser un desafío debido a la naturaleza del algoritmo, sin embargo, puede trabajar con datos nulos.

Los resultados observados revelan, la capacidad de clasificación de facies de cada uno de los métodos de ML, exponiendo sus ventajas y desventajas. La idea de utilizar varios métodos es para tener un mejor criterio de clasificación de las facies, de tal manera que se complementan unos con otros y nos presentan una solución con un espectro mayor.

Dedicatoria

Todos mis logros quiero compartirlos contigo, para mi mamá Lourdes Natalio.
Y para mi sobrino Ian, llegaste a alegrar mi vida.

Agradecimientos

Quiero agradecer primero a la persona que está detrás de todo esto, al director externo de esta tesis, el Dr. Luis Ramírez Cruz, quien desde que lo conozco siempre me ha impulsado a desarrollarme profesionalmente, siempre confiando en mi capacidad, gracias por todas las enseñanzas y consejos, no pude caer en mejores manos en Petrobal, una gran persona, a quien siempre estimaré y con la cual estaré infinitamente agradecida. También quiero agradecer al MTRO. Ernesto Cisneros, quien se unió, confió y apoyó este proyecto de tesis, infinitas gracias por todo.

Agradezco a mi mamá Lourdes Natalio, gracias por motivarme a seguir mis sueños, gracias por todas tus enseñanzas, eres una mujer maravillosa, a mi papá Jose Cruz, por apoyarme en todas mis decisiones, por siempre motivarme a aprender algo nuevo, a mi hermana y mejor amiga Carolina Cruz quien siempre ha estado conmigo apoyándome mejor que nadie, te admiro mucho, a mis hermanas perrunas Camila y KongaAna, gracias por irme a dejar en las mañanas a la parada de autobús y quedarse conmigo en mis noches de desvelo, a mi novio Ulises gracias por llegar en esta última etapa de mi vida universitaria, me motivaste a no rendirme, dándome ánimos cuando no corría mi programa y a Victor quien siempre celebra todos mis éxitos, gracias por ser un gran amigo.

Agradezco también a la Universidad Nacional Autónoma de México, por todo lo que me brindó desde la preparatoria y su formación durante estos años. Agradezco a mis profesores y sinodales de la Facultad de Ingeniería, al MTRO. Jorge Guizar que es una gran persona y del cual aprendí mucho, por más profesores como tú, al M.C. Miguel Vargas, en su clase encontré mi gusto por programar, al Ing. Mauricio Buendía tus comentarios y sugerencias hicieron crecer mis conocimientos en temas que desconocía, y a la Dra. Iza Canales por sus palabras y recomendaciones, muchas gracias a todos.

Agradezco a Petrobal por permitirme trabajar con sus datos y en su empresa. También a su equipo de geociencias al Ing. Javier Méndez, al Ing. Fernando López y al Ing. Hugo Saucedo, quienes me brindaron apoyo y orientación para el desarrollo de esta tesis, y siempre estuvieron dispuestos a resolver mis dudas.

Y por último pero no menos importante a las personas que hicieron mucho más divertida y amena la universidad, a mis amigas Amairany Rodriguez y Monserrat Peñaloza, reí mucho con ustedes, definitivamente viví los mejores momentos de la facultad con ustedes.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El avance de la Inteligencia Artificial en todas las áreas del mundo ha tenido un crecimiento acelerado, y en las geociencias no es la excepción, la aplicación de distintos modelos de Machine Learning en esta área ha ido en aumento, ya sea para clasificación, regresión y agrupación, teniendo a favor la gran cantidad de datos que se manejan en geofísica. Algunas aplicaciones de ML en la geofísica son predicción de facies con registros geofísicos de pozos, predicción de facies con sísmica, predicción de fallas, detección de sal en la sísmica, inversión sísmica, para problemas de regresión se puede aplicar a la atenuación de ruido, secciones sísmicas sintéticas, etc.

El objetivo del proyecto es desarrollar y probar las distintas técnicas de ML, que se pueden aplicar para la clasificación de facies. Establecer una relación entre las facies y los registros geofísicos de pozo es fundamental en la industria petrolera para comprender las características del subsuelo y la distribución de los yacimientos de hidrocarburos. En este proyecto se busca desarrollar un algoritmo que logre establecer una relación significativa entre los registros geofísicos de pozo y las facies.

El proyecto se aplicó a datos reales de pozos, ubicados en las Cuencas del Sureste. Las facies utilizadas son interpretaciones de un petrofísico, se sabe que las interpretaciones pueden diferir en cada intérprete, dependiendo la experiencia, el criterio, los conocimientos previos sobre la zona de estudio, entre otros factores. Se define como facies a las propiedades físicas, químicas y biológicas de las rocas, que permiten una descripción específica de esta (Cross & Homewood, 1997), las facies litológicas son las características físicas como textura, mineralogía y tamaño de grano de la roca.

En los últimos años se ha realizado varios modelos de ML aplicados a la clasificación de facies con registros geofísicos de pozos, que van desde el enfoque estadístico multivariante (Wolf, 1982), métodos supervisados como Support Vector Machine (Hall, 2016) y (Guerra et al., 2017), Gradient Boosting Machine (Bestagini et al., 2017), Naïve Bayes (Li & Anderson-Sprecher, 2006), hasta métodos de aprendizaje profundo. Este proyecto plantea usar múltiples métodos de aprendizaje supervisado y finalizar con un modelo en conjunto.

El uso de técnicas avanzadas de análisis de datos, como el aprendizaje supervisado y el análisis estadístico multivariado, puede ayudar a identificar patrones y relaciones complejas entre los registros geofísicos de pozo y las facies. Estas técnicas pueden mejorar la precisión de la correlación y la interpretación de las facies.

Capítulo 2

Marco Geológico

2.1. Contexto Geológico Regional

El área de estudio se encuentra en la provincia petrolera las Cuencas del Sureste, incluye una parte terrestre y una parte marina, en la Figura 2.1 se observan las cuencas petroleras de México, resaltando las Cuencas del Sureste, limitada al norte por la isobata de 500 [m], al sur por el Cinturón plegado de Chiapas, al oeste por la Cuenca de Veracruz y al este por la Plataforma de Yucatán (CNH, 2014).

Figura 2.1: Cuencas Petroleras de México. (CNH, 2014).

Las Cuencas del Sureste evolucionan dentro de un margen pasivo, desde la apertura del Golfo de México en el Jurásico Medio, la instalación y extensión de plataformas de sedimentación carbonatada durante el Cretácico, hasta las condiciones de cuenca subsidente de tipo antefosa que termina con el cierre y colmatación sedimentaria al final del Neógeno.

En la Figura 2.2 se muestran las cuencas que comprende las Cuencas del Sureste, las cuales son la cuenca Salina del Istmo, la cuenca de Comalcalco, la cuenca de Macuspana y el Pilar Reforma-Akal. Los pozos con los que se trabajó se encuentran en el Pilar Reforma-Akal, el cual consiste de un gran pilar enmarcado por la prolongación hacia el mar de las fallas de Comalcalco, Frontera y Macuspana que actuaron como verdaderas fallas de tiempo, permitiendo la acumulación de calizas clásticas con una enorme capacidad de almacenar hidrocarburos (Ángeles Aquino, 2006).

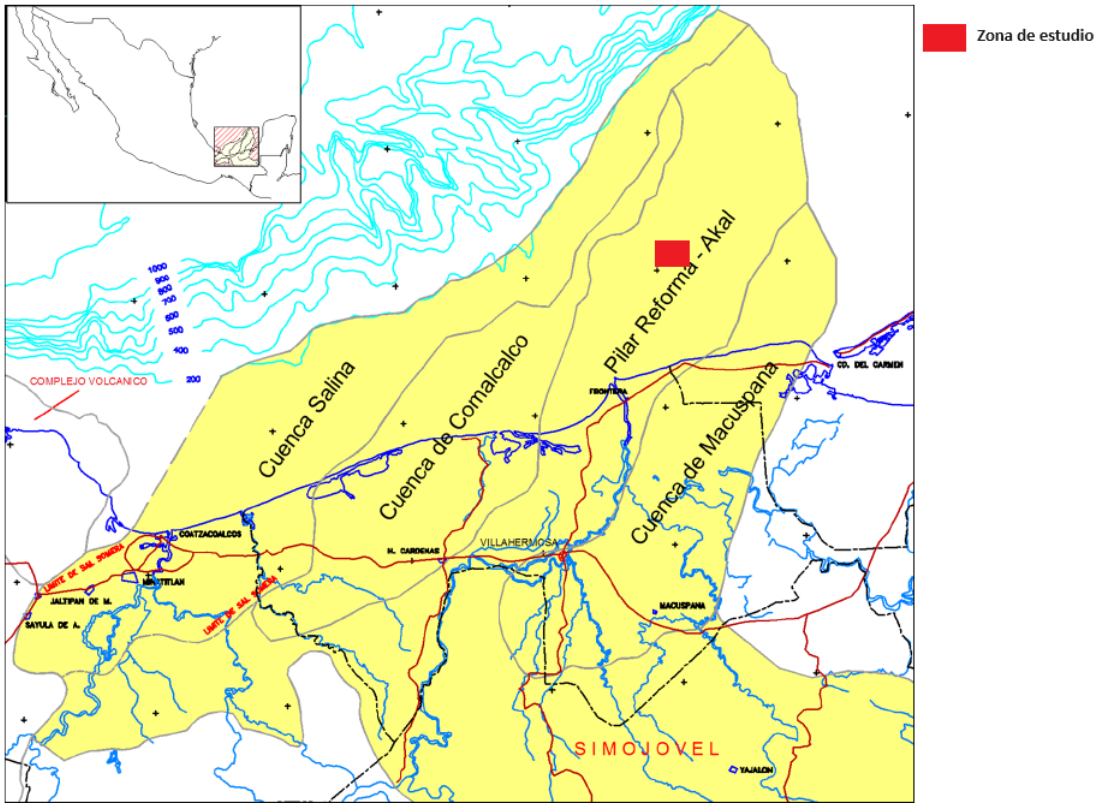


Figura 2.2: Cuencas del Sureste y ubicación de los pozos de estudio. (CNH, 2014).

Desde el punto de vista petrolero, ésta es la provincia más importante del país, ya que la mayor parte de la producción de las Cuencas del Sureste y del país se localiza en ella, en la Figura 2.3 se observan los campos de aceite y gas dentro de las Cuencas del Sureste.

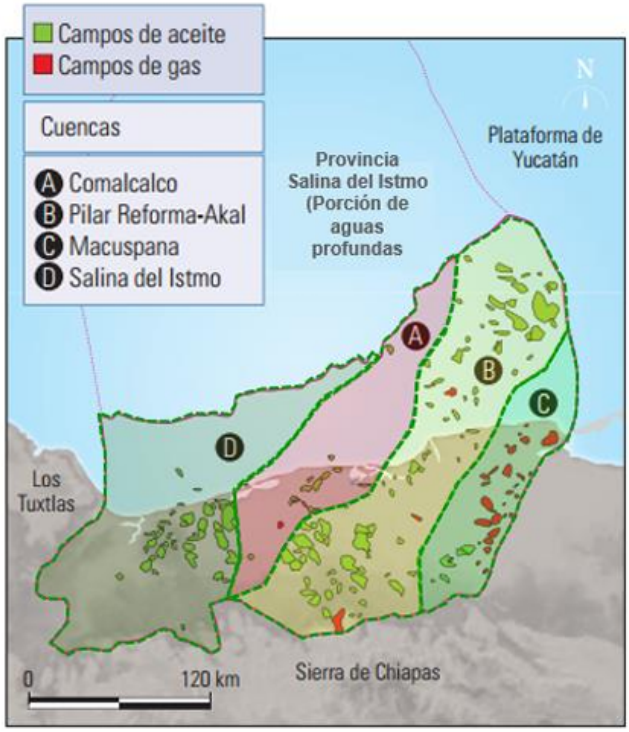


Figura 2.3: Cuencas del Sureste y campos de aceite y gas. (CNH, 2015)

Es probable que este pilar esté íntimamente ligado con fallas profundas en la corteza, que formaron bloques escalonados rellenados por flujos de detritos que constituían brechas. Estos bloques, son el resultado de esfuerzos de distensión y compresión a que ha estado sometida la corteza durante los diferentes eventos ocurridos en la zona (Ángeles Aquino, 2006).

2.2. Marco Estratigráfico

La columna estratigráfica de la zona abarca desde el Jurásico Medio hasta el Reciente, Figura 2.4. La zona de interés se encuentra en Jurásico Superior Kimmeridgiano, Jurásico Superior Tithoniano y Cretácico.

Paleozoico

Está constituido de sedimentos continentales rojos del Paleozoico Medio y de un basamento más antiguo cristalino y metamórfico (CNH, 2014).

Calloviano

La etapa inicial de la apertura del Golfo de México está relacionada con la fragmentación y dispersión de la Pangea. La sal del Golfo de México se depositó durante esta edad, aunque tienen diferentes edades y posiciones estratigráficas, siendo más

antiguas hacia el centro de la cuenca y más jóvenes hacia el borde de ésta, en donde también varía su litología a otro tipo de evaporitas (Padilla y Sánchez, 2007).

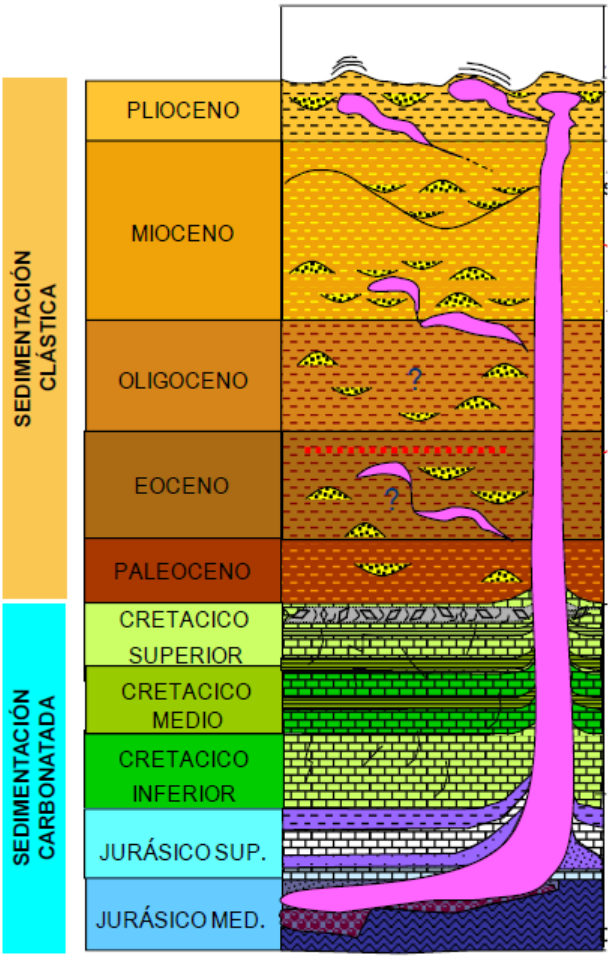


Figura 2.4: Columna estratigráfica. (CNH, 2014).

Oxfordiano

Se desarrollaron amplias plataformas de aguas someras, en las que se depositaron grandes volúmenes de carbonatos, con extensas barras oolíticas (Padilla y Sánchez, 2007).

Kimmeridgiano

Los sedimentos están representados por espesores de terrígenos que, en algunos lugares de la cuenca gradúan a carbonatos de bancos oolíticos parcialmente dolomitizados hacia la cima de la formación (CNH, 2014).

Tithoniano

Se depositaron una mezcla de terrígenos finos y carbonatos con intercalaciones delgadas, ricos en materia orgánica, que son la roca generadora de la mayoría de los volúmenes de hidrocarburos en el Golfo de México (CNH, 2015).

Cretácico

Se tiene una secuencia sedimentaria de plataforma, con litofacies de carbonatos y carbonatos arcillosos dolomitizados con pedernal e intercalaciones de horizontes bentoníticos. A finales del Cretácico se tiene un crecimiento de la plataforma carbonatada y el depósito de brechas y flujos turbidíticos en el talud continental de la Sonda de Campeche, cuyas litofacies predominantes son dolomías, calizas arcillosas y brechas dolomitizadas (CNH, 2014).

Cenozoico

Entre el límite del Período Cretácico y el Período Paleógeno, hay una brecha en carbonatos al Sur del Golfo de México. Durante el Eoceno tardío se depositaron gruesos espesores de arenas finas en los taludes occidentales, al mismo tiempo sedimentos más finos iban rellenando las partes más profundas. Como consecuencia a esto, la sal y la arcilla empezaron a moverse (Padilla y Sánchez, 2007).

2.3. Marco Estructural General

En la zona del Pilar de Akal se distinguen tres eventos tectónicos que han conformado el marco estructural regional.

- Un evento extensional en el Jurásico Medio, asociado a la separación de los continentes que de una u otra forma se manifestaron en el Golfo, formando inicialmente grandes fracturas que posteriormente constituyeron fallas normales con dirección N-S, formando bloques escalonados, algunos de los cuales constituyeron fallas de crecimiento dando lugar a las fosas de Macuspana y Comalcalco, respectivamente (Ángeles Aquino, 2006).
- Un evento compresivo ocasionado por empujes de una placa sobre un elemento rígido, formando en el Área Marina estructuras alineadas con dirección NW-SE, ocasionadas por giros de bloques que constituyeron fallas inversas perpendiculares a los esfuerzos NW-SE (Ángeles Aquino, 2006).
- Un evento extensional final durante el Neógeno grandes fallas de lístricas que, en algunos casos, están asociadas con intrusiones de arcilla y/o sal (CNH, 2014).

En la Figura 2.5 se observa en la parte de abajo una sección geológica que abarca desde el Cinturón plegado Chuktah-Tamil hasta la Cuenca de Macuspana, pasando por el Pilar de Akal, que es la zona de estudio.

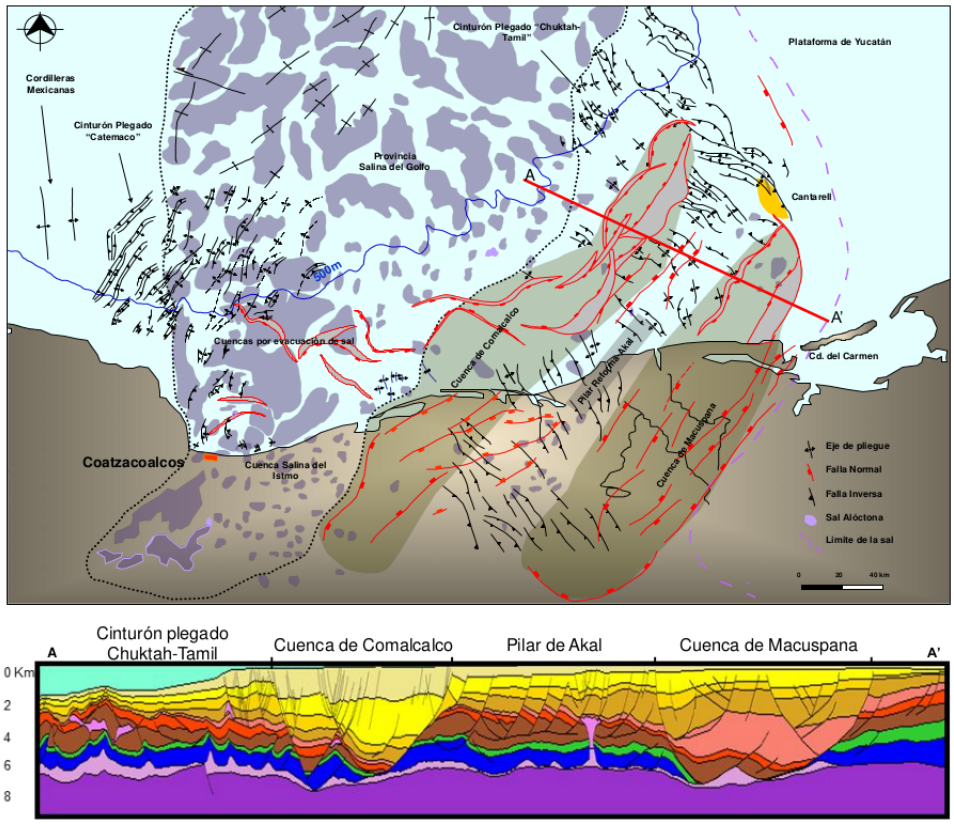


Figura 2.5: Marco tectónico y sección geológica regional de las Cuencas del Sureste. (CNH, 2014).

2.4. Facies Sedimentarias

Son tipos específicos de rocas sedimentarias que se caracterizan por tener ciertas características particulares en cuanto a su composición, textura, estructura y contenido de fósiles. Estas características se forman en ambientes sedimentarios específicos, como cuerpos de agua, desiertos, glaciares, entre otros, y son el resultado de procesos físicos, químicos y biológicos que ocurren en esos entornos.

Las facies sedimentarias son importantes en geología porque proporcionan información crucial sobre las condiciones ambientales en las que se formaron las rocas sedimentarias, lo que a su vez puede ayudar a los geólogos a reconstruir la historia geológica de una región particular. Además, las facies sedimentarias son útiles para interpretar ambientes pasados y predecir la presencia de recursos naturales como petróleo, gas natural y agua subterránea.

Una manera de describir facies sedimentarias es a través de las características fácilmente observables de las rocas y la interpretación de los ambientes deposicionales. Los seis aspectos más importantes son el color/composición, la textura, las estructuras sedimentarias, los fósiles, la asociación y la forma.

Las facies son de gran importancia en la exploración petrolera por varias razones:

- Ambientes de depósito: las facies sedimentarias indican los ambientes geológi-

cos en los que se depositaron los sedimentos que eventualmente se convirtieron en rocas sedimentarias. Estos ambientes pueden incluir deltas, playas, marismas, plataformas continentales, entre otros. Cada ambiente tiene características específicas que influyen en la presencia y distribución de depósitos de petróleo y gas.

- Rocas almacén: algunas facies sedimentarias, como las areniscas y calizas porosas, tienen la capacidad de almacenar y permitir el flujo de petróleo y gas. Estas rocas, conocidas como rocas reservorio, son el objetivo principal de la exploración petrolera. Comprender las características de estas facies es crucial para identificar y evaluar adecuadamente los yacimientos de hidrocarburos.

- Sistemas de sellado: junto con las rocas reservorio, las facies sedimentarias también pueden actuar como rocas selladoras que impiden que el petróleo y el gas escapen hacia la superficie. Las arcillas, por ejemplo, son facies sedimentarias comúnmente asociadas con la formación de sellos impermeables que retienen los hidrocarburos en los reservorios.

- Modelado de cuencas: el estudio de las facies sedimentarias en una cuenca sedimentaria puede proporcionar información sobre la evolución geológica de la cuenca a lo largo del tiempo. Esto incluye la identificación de los procesos deposicionales, los cambios ambientales y la distribución espacial de los depósitos sedimentarios, lo que puede ayudar a predecir la ubicación de posibles acumulaciones de hidrocarburos.

En resumen, comprender las facies sedimentarias es fundamental para la exploración petrolera ya que proporciona información clave sobre la presencia, distribución y calidad de los yacimientos de petróleo y gas, así como sobre los procesos geológicos que influyeron en su formación y preservación (Petrobal, 2023).

Facies oolíticas

El origen de las oolitas tiene lugar en un ambiente marino, a partir de un núcleo que puede ser un grano, fragmento de fósil o litoclasto y está asociado al movimiento constante de las olas o corrientes marinas someras. El grano al estar en movimiento se le adhieren algunas capas concéntricas de calcita y aragonita y tiene un crecimiento en diámetro, estos gránulos tienen un diámetro típico de entre 0.25 y 2 [mm] (Petrobal, 2023).

Los procesos de precipitación y acreción que dan lugar a la formación de ooides pueden ocurrir en varios entornos, como en playas, arrecifes de coral, lagunas costeras y plataformas continentales. Las formaciones oolíticas pueden variar en apariencia y composición dependiendo de factores como la naturaleza de los núcleos alrededor de los cuales se forman los ooides, la concentración de minerales en el agua, y la duración y la intensidad de los procesos de formación.

Posteriormente, durante la diagénesis presenta una estructura radial y concéntrica. Cuando la roca está soportada por la matriz de calcita espática y las oolitas están

diseminadas en la estructura de la roca, las características petrofísicas son bajas en términos de porosidad, tratándose de oolitas que cayeron a un ambiente de menor energía. En las arenas de playa constituida por fósiles y granos de oolitas en un ambiente de alta energía son las de mayor porosidad. Los bancos desarrollados en la plataforma somera están bajo condiciones de corrientes marinas someras y vientos, son las que tienen mayor distribución espacial y de mayor acumulación en espesor, en estos bancos la energía es mayor que en las áreas donde las arenas están dispersas.

Estas rocas son importantes como reservorios de petróleo y gas en algunas áreas. En este proyecto, las oolitas toman gran relevancia, debido a que constituyen la roca almacenadora (yacimiento) de hidrocarburo del área estudiada, lo cual se confirma en los registros geofísicos de los pozos analizados aquí.

Establecer una relación entre las facies y los registros geofísicos de pozo es fundamental en la industria petrolera para comprender las características del subsuelo y la distribución de los yacimientos de hidrocarburos (Petrobal, 2023).

2.5. Litofacies del sistema petrolero

Los hidrocarburos se encuentran almacenados principalmente en las brechas del Cretácico Superior y en las calizas oolíticas del Jurásico Superior.

2.5.1. Kimmeridgiano

Las zonas más proximales fueron ocupadas durante todo el ciclo de depósito del Kimmeridgiano por una rampa interna donde se producen y se depositan los arrecifes y *grainstones*, bajo un marco de agua somera de alta energía. La presencia de estas facies está controlada por la energía de las olas provenientes del NW y una tasa de subsidencia superpuestas que permite el depósito de una pila gruesa de carbonato. Estas facies son consideradas como los principales *plays* almacén de la sucesión del Kimmeridgiano.

La parte occidental, en su mayoría, está ocupada por carbonatos arcillosos, a veces dolomitizados, depositados en un ambiente de mar abierto en aguas relativamente profundas, pero con una tasa relativamente baja de la subsidencia. Al final del Kimmeridgiano, la parte más oriental, en su mayoría, está representada por heterolíticos de laguna en un "backshore" y ambiente de depósito protegido (CNH, 2014).

La roca almacén para el Kimmeridgiano está constituida por facies de bancos oolíticos que corresponden a packstone-grainstone de ooides y dolomías con sombras de ooides, en un paleoambiente de borde externo y frente de bancos; los espesores de la roca almacén varían de 50 [m] a más de 300 [m] (Faz Pérez, 2014).

2.5.2. Tithoniano

Las rocas de esta edad la constituyen calizas arcillosas color oscuro, con intercalaciones de lutitas bituminosas ligeramente calcáreas, en algunas partes se observan

carbonosas, contienen abundante materia orgánica de origen vegetal (algáceas, restos de plantas y carbonosas) y animal (restos de pez), gradúan lateralmente a mudstone arcilloso bentónico color negro olivo y anhidrita gris blanquizca (Ángeles Aquino, 2006).

Las rocas del Tithoniano constituyen el principal elemento generador de hidrocarburos de las Cuencas del Sureste. Se distribuyen ampliamente en el área con espesores entre los 100 y 400 metros. Las litofacies se asocian con ambientes carbonatados profundos que van de plataforma externa a cuenca. Esta roca ha sido considerada y comprobada, como responsable de las acumulaciones regionales de aceite ligero y mediano (CNH, 2015).

2.5.3. Cretácico

La mayor parte de la sucesión cretácica está dominada por el depósito de carbonatos sobre una plataforma con pendiente. Las facies de plataforma interna tienden a acuñarse rápidamente hacia el oeste. La mayor parte del área Aguas Someras se caracteriza por una columna gruesa de litofacies *mudstone* y *wackstone* depositados en un ambiente que va de talud a batial. Las facies son de tipo *mudstone* y el potencial como roca almacén se limita a un posible fracturamiento.

Durante el Cretácico Superior, Figura 2.6, aparece un cambio en el sistema y la sedimentación se vuelve clástica, con la edificación de una gruesa secuencia de brechas, la cual constituye un *play* almacén principal para esta secuencia. Esta brecha viene desde el noreste y cubre la parte de las Cuencas del Sureste hasta el oeste, se depositaron en forma caótica como enormes flujos de dentritos y fueron posteriormente cubiertos por sedimentos finos (Faz Pérez, 2014).

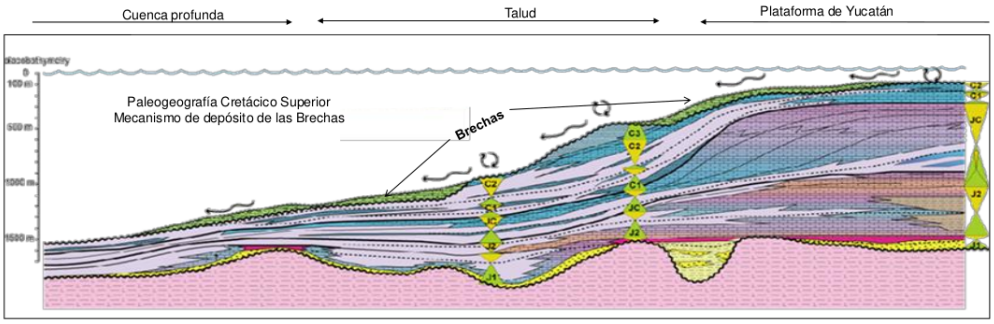


Figura 2.6: Litofacies del sistema almacenador Cretácico Superior. (CNH, 2014).

Capítulo 3

Machine Learning

Machine Learning es una rama de la Inteligencia Artificial, como su nombre lo dice son un conjunto de algoritmos que le dan la capacidad a la máquina de realizar actividades sin la necesidad de ser programados explícitamente para ello, son usados en clasificación, regresión y agrupación. Sin embargo, ML no es un proceso sencillo, conforme el algoritmo ingiere datos de entrenamiento, es posible producir modelos más precisos basados en datos. Un modelo de ML es la salida de información que se genera cuando se entrena el algoritmo con datos (Hurwitz & Kirsch, 2018). Después del entrenamiento, al proporcionar un modelo con una entrada, se le dará una salida. A los datos de entrada les llamamos características y a los datos de salida clases.

Se divide en varias ramas como son aprendizaje supervisado, aprendizaje no supervisado, aprendizaje reforzado y aprendizaje profundo, pueden combinarse entre ellos, este trabajo está enfocado en el aprendizaje supervisado.

Existen varias técnicas de ML que han sido utilizadas para clasificar facies utilizando datos de registros geofísicos de pozos. Algunas de estas técnicas son: el enfoque estadístico multivariante (Wolf, 1982), métodos supervisados como Support Vector Machine (Hall, 2016) y (Guerra et al., 2017), Gradient Boosting Machine (Bestagini et al., 2017), Naïve Bayes (Li & Anderson-Sprecher, 2006), las cuales muestran unos resultados visualmente buenos.

Estas son solo algunas de las técnicas de ML que han sido utilizadas para clasificar facies utilizando datos de registros geofísicos de pozos. La elección del método más adecuado dependerá de la naturaleza de los datos y de los objetivos específicos del proyecto. En este caso se evaluarán varias de las técnicas para mejorar el criterio de clasificación y reducir la incertidumbre en los resultados.

3.1. Aprendizaje supervisado

En el aprendizaje supervisado se tiene un conjunto de datos de entrenamiento el cual contiene características (datos de entrada) y las etiquetas o clases que son la respuesta o el resultado (datos de salida). Es decir, estos datos tienen características etiquetadas que definen el significado de los datos (Hurwitz & Kirsch, 2018).

Puede aplicarse para problemas de regresión y clasificación. En este trabajo se

utilizó para clasificación, se puede dividir los modelos a usar en dos grupos basados en la interpretación geométrica y basados en árboles de decisión.

3.1.1. Support Vector Machine

Es un método desarrollado en los años 1990's. Su idea principal es mapear los vectores de entrada y dividirlos con un hiperplano (Cortes & Vapnik, 1995).

Clasificador de margen máximo

Un hiperplano en p-dimensiones se define como:

$$\beta_o + \beta_1\chi_1 + \beta_2\chi_2 + \cdots + \beta_p\chi_p = 0, \tag{3.1}$$

Donde $\beta_o, \beta_1, \ldots, \beta_p$ son los parámetros que definen la función. Supongamos ahora que tenemos n observaciones de entrenamiento en un espacio p-dimensional. Estas observaciones se dividen en dos clases $y_1, ..., y_n \in (-1, 1)$ donde -1 es una clase y 1 otra clase.

En la Figura 3.1, tenemos dos clases separadas por varios hiperplanos. El objetivo es encontrar un hiperplano de separación óptimo, definido como una función de decisión lineal con un margen máximo entre las clases.

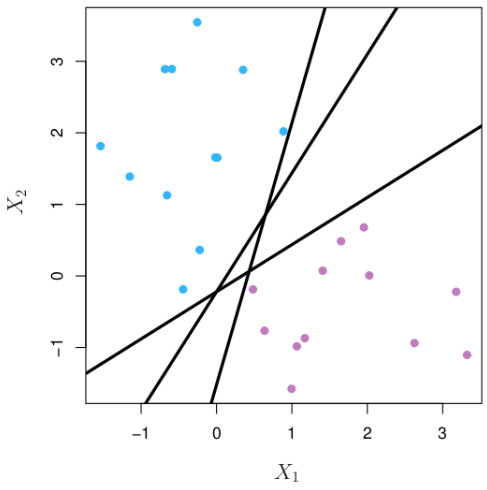


Figura 3.1: Dos clases separadas por diferentes posibles hiperplanos. (James et al., 2021).

En la Figura 3.2 se ven tres valores de entrenamiento que se encuentran en las líneas del margen. A estos valores se les conoce como vectores de soporte. El hiperplano y el margen dependen directamente de estos vectores de soporte.

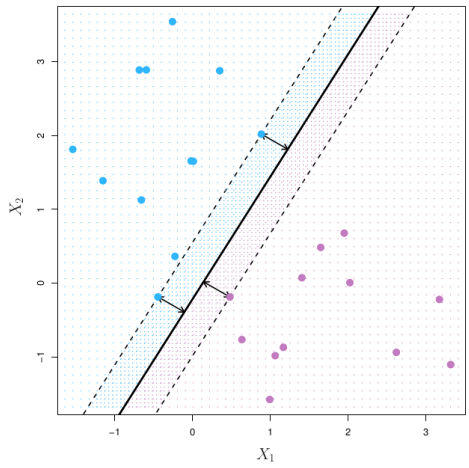


Figura 3.2: Dos clases separadas por un hiperplano óptimo. (James et al., 2021).

En este ejemplo se muestra un caso simple donde se puede hacer una separación clara de los datos, pero hay casos donde no existe el hiperplano de separación óptimo, y por lo tanto no hay un clasificador de margen máximo. Pero se puede desarrollar un hiperplano con margen suave y ahora tendremos un Clasificador de Vectores de Soporte ("SVC", por sus siglas en inglés).

Clasificador de vectores de soporte

En ocasiones el hiperplano de separación puede no ser la mejor opción, por ejemplo en la Figura 3.3 tenemos el hiperplano de separación (linea negra) tiene un margen pequeño lo cual no es bueno, como vimos anteriormente, este debe ser grande para mayor confianza. En este caso podemos considerar un hiperplano que no separa perfectamente las dos clases (línea punteada). Con esto nos referimos a que es preferible clasificar mal unas cuantas observaciones para poder clasificar mejor las otras.

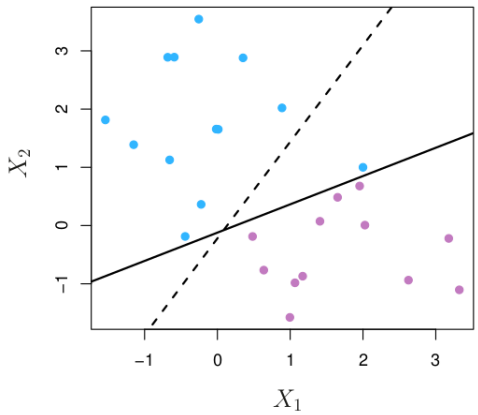


Figura 3.3: Hiperplano de separación óptimo vs. clasificador de margen suave. (James et al., 2021).

El clasificador de vectores de soporte, también conocido como clasificador de margen suave, permite que algunas observaciones se salgan de su margen o incluso estén del otro lado del hiperplano.

Detalles del SVC

Se debe construir un hiperplano que separe correctamente la mayoría de observaciones de entrenamiento $x_1, \ldots, x_n$, basado en estas y las etiquetas de clase $y_1, \ldots, y_n$.

$$\underset{\beta_0, \beta_1, \ldots, \beta_p, \epsilon_1, \ldots, \epsilon_n, M}{\text{maximizar}} \quad \text{M} \tag{3.2}$$

$$\text{sujeto a} \quad \sum_{j=1}^{p} \beta_j^2 = 1, \tag{3.3}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i), \tag{3.4}$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C, \tag{3.5}$$

donde M representa el margen mayor posible, y $\epsilon_1, \ldots, \epsilon_n$ son "variables de holgura" que nos dice dónde se encuentra la observación, en relación con el hiperplano y el margen, dependiendo de donde se encuentre se le asigna un valor, por ejemplo si $\epsilon_i = 0$ el punto se encuentra en su zona, si $0 < \epsilon_i < 1$ se encuentra dentro del margen sin pasar el hiperplano, si $\epsilon_i > 1$ el punto ya paso el hiperplano. Ahora, C es un parámetro que limita la suma de las variables de holgura, determina que tan tolerante será con las violaciones al margen e hiperplano, mientras más grande sea C más tolerante será y viceversa.

En la práctica, C se elige mediante la validación cruzada. Hay que saber que solo los datos que se encuentran en el margen y los que violan el margen afectan al hiperplano y por lo tanto, al clasificador. Estos datos se conocen como vectores de soporte.

Clasificación con límites de decisión no lineales

En la práctica tendremos casos donde las clases no se podrán separar linealmente, como en la Figura 3.4.

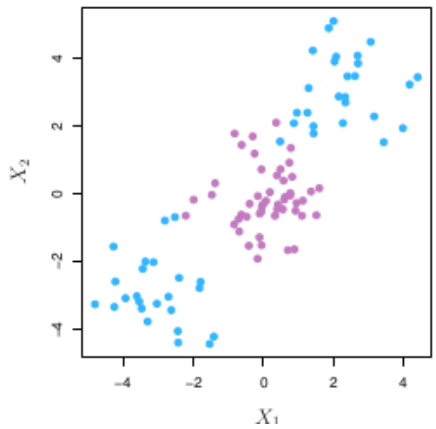


Figura 3.4: Caso donde las dos clases no se pueden separar linealmente. (James et al., 2021).

Está claro que no se puede usar un clasificador lineal, no funcionará, entonces ajustaremos un clasificador ampliando el espacio de características mediante funciones cuadráticas, cúbicas, etc.

Ahora las ecuaciones (3.2)-(3.5) se convertirán en:

$$\underset{\beta_0,\beta_{11},\beta_{12},\ldots,\beta_{p1},\beta_{p2}\epsilon_1,\ldots,\epsilon_n,M}{\text{maximizar}} \quad \text{M}$$

$$\text{sujeto a} \quad y_i\Big(\beta_0+\sum_{j=1}^{p}\beta_{j1}x_{ij}+\sum_{j=1}^{p}\beta_{j2}x_{ij}^2\Big)\geq M(1-\epsilon_i), \tag{3.6}$$

$$\sum_{i=1}^{n}\epsilon_i\leq C_1, \quad \epsilon_i\geq 0_1, \quad \sum_{j=1}^{p}\sum_{k=1}^{2}\beta_{jk}^2=1$$

Support Vector Machine

Es una extensión del SVC que resulta de ampliar el espacio de características utilizando kernels.

Se puede demostrar que el clasificador lineal de vectores de soporte se representa como:

$$f(x)=\beta_0+\sum_{i\in S}\alpha_i(x,x_i), \tag{3.7}$$

donde S es la colección de índices de los puntos de apoyo. Ahora vamos a sustituir el siguiente término

$$K(x_i,x_{i'}), \tag{3.8}$$

Donde K es una función llamada kernel que cuantifica la similitud de dos observaciones. Por ejemplo:

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}, \tag{3.9}$$

Esta ecuación se conoce como kernel lineal y cuantifica esencialmente la similitud de un par de observaciones. Pero podríamos sustituir por:

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d, \tag{3.10}$$

Esto se conoce como un kernel polinómico de grado d, aunque existen muchas otras alternativas. Utilizar esta ecuación en lugar de la anterior conduce a un límite de decisión mucho más flexible y también ajusta el SVC a un espacio de mayor dimensión. Cuando el SVC se combina con un núcleo no lineal, el clasificador resultante se conoce como máquina de vectores de soporte. En este caso la función (no lineal) tiene la forma de:

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i), \tag{3.11}$$

Y se obtiene como resultado la Figura 3.5:

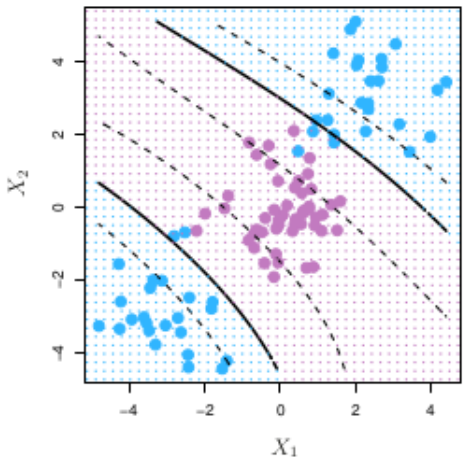


Figura 3.5: Ejemplo usando SVM con kernel polinómico. (James et al., 2021).

3.1.2. K-Nearest Neighbor

La regla de k vecinos cercanos formulada en 1951 por Fix y Hodges, y después por (Cover & Hart, 1967), se basa en mapear todas las clases las cuales llamamos "vecindarios", los nuevos ejemplos a clasificar se mapean y en base a sus k vecinos cercanos clasifican los nuevos datos, por ejemplo en la Figura 3.6 se tienen 3 clases y un punto nuevo a clasificar de color negro.

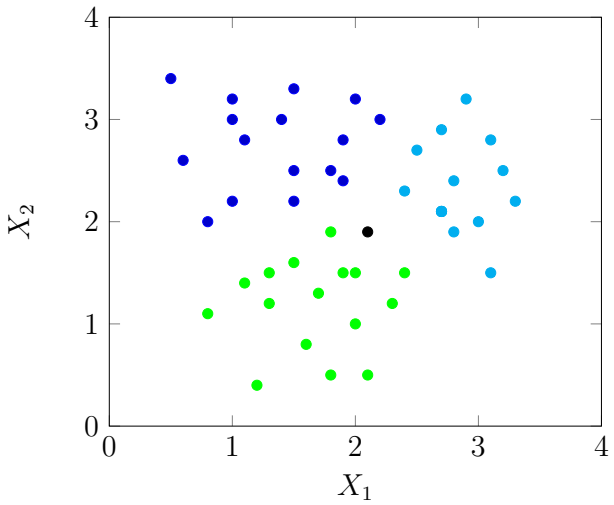


Figura 3.6: Tres vecindarios (clases) y un punto nuevo a clasificar en base a sus vecinos cercanos.

La clasificación consta de dos etapas (Cunningham & Delany, 2007):

1. Definir el número de vecinos (k).
2. Determinar a qué clase pertenece utilizando (k).

Tenemos un conjunto de entrenamiento D compuesto por las muestras $(x_i)_{i\in[1,|D|]}$, para cada dato de entrenamiento, también tenemos un conjunto de características F. Para cada ejemplo de entrenamiento hay una ejemplo de clase $y_j \in Y$. La idea es clasificar un dato desconocido q. Para cada $x_i \in D$ calculamos la distancia entre q y x_i, dada por:

$$d(q, x_i) = \sum_{f\in F} w f \delta(q_f, x_{if}), \tag{3.12}$$

Donde w es el peso y δ las posibles distancias.

Los k vecinos más cercanos se seleccionan dependiendo la distancia aplicada. Las más comunes son la distancia Euclidiana y la distancia Manhattan:

$$d(q, x_i) = \sqrt{\sum_{k=1}^{n} (q_k - y_{ik})^2}, \quad \text{Distancia Euclidiana} \tag{3.13}$$

$$d(q, x_i) = \sum_{k=1}^{n} |q_k - x_{ik}|, \quad \text{Distancia Manhattan} \tag{3.14}$$

Como vemos en la Figura 3.7, ambas miden la distancia de un punto a otro. Algunas medidas de distancia como la Euclidiana, pueden ser afectadas por la alta dimensionalidad de los datos (Steinbach & Tan, 2009), debido a que al incrementar las dimensiones la distancia entre puntos se vuelve más dispersa, de igual forma pueden verse afectados por valores atípicos, en este caso se prefiere usar la distancia Manhattan.

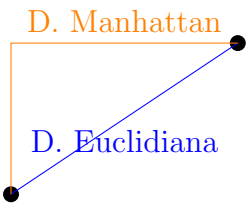


Figura 3.7: Diferencia entre distancias

3.1.3. Árboles de decisión

El método de árboles de decisión, fue propuesto por primera vez por (Breiman et al., 1984), consiste en divisiones a partir de un conjunto de datos X, dividiéndolo en subconjuntos. Es decir para dividir los datos se plantean "preguntas", se obtiene una respuesta y a partir de esta se plantea otra pregunta, hasta obtener un resultado (Taiz, s.f.), es un método muy fácil de entender.

Algunos conceptos básicos son:

- Nodo raíz: Es donde surge el árbol.
- Nodo decisión: Se encuentran entre la raíz y los nodos hoja, conducen a otros nodos.
- Nodos hoja: Es donde termina nuestro árbol, ya no surgen más preguntas después.

En la Figura 3.8, se muestra un árbol para tres clases, se puede distinguir las tres partes del árbol, el nodo raíz, los nodos decisión y los nodos hoja, el árbol tiene una profundidad de 2, esto quiere decir el número de niveles de nodos hasta los nodos hoja, sin contar el nodo raíz.

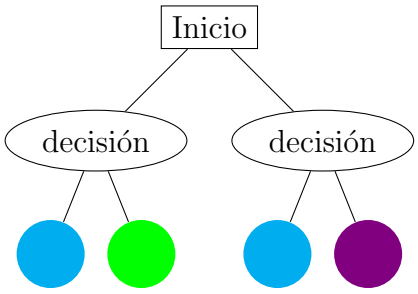


Figura 3.8: Árbol de decisión para 3 clases

El error de clasificación está dado por:

$$E = 1 - \max_k(\hat{p}_{mk}), \tag{3.15}$$

Donde $\hat{p}_{mk}$ representa los datos de entrenamiento de la m-ésima región de la k-ésima clase. Según (James et al., 2021), cada terminación del nodo hoja es llamada región y se delimitan por las preguntas que fueron formuladas, en los nodos decisión.

Para evaluar la calidad de una división en un nodo se tienen dos medidas:

Índice de Gini

Es una medida que nos indica la pureza del nodo, valores más pequeños nos indican que el nodo contiene más observaciones sobre una sola clase, entonces mientras más pequeño sea el valor mejor será la división .

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}), \tag{3.16}$$

Entropía

La entropía igual mide la pureza del nodo, teniendo valores más pequeños la división será mejor.

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} lg \hat{p}_{mk}, \tag{3.17}$$

En la Figura 3.9 podemos ver cómo varían los valores para medir la pureza del nodo y el error de clasificación, el índice de Gini va a tener valores de 0 a 0.5 y la entropía valores de 0 a 1. Se podría decir que son iguales, ya que miden la pureza del nodo, aunque en el tiempo de cómputo la entropía ocupa mayor tiempo.

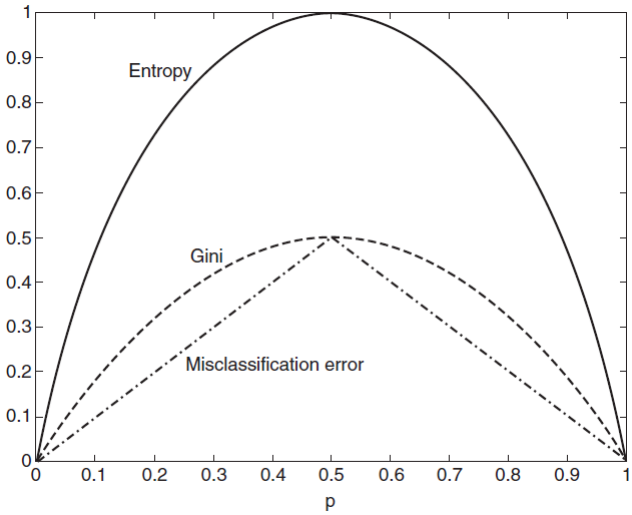


Figura 3.9: Comparación entre el índice de Gini, la entropía y el error de clasificación. (Taiz, s.f.).

Para entender mejor la gráfica de la Figura 3.9, se realiza el siguiente ejemplo:

Nodo 1: Clase 1 = 0 y Clase 2 = 6
Gini $= 1 - \frac{0}{6}^2 - \frac{6}{6}^2 = 0$
Entropía $= -\frac{0}{6} lg_2(\frac{0}{6}) - \frac{6}{6} lg_2(\frac{6}{6}) = 0$
Error $= 1 - \text{máx}[\frac{0}{6}, \frac{6}{6}] = 0$

Nodo 2: Clase 1 = 3 y Clase 2 = 3
Gini $= 1 - \frac{3}{6}^2 - \frac{3}{6}^2 = 0.5$
Entropía $= -\frac{3}{6} lg_2(\frac{3}{6}) - \frac{3}{6} lg_2(\frac{3}{6}) = 1$
Error $= 1 - \text{máx}[\frac{3}{6}, \frac{3}{6}] = 0.5$

Como se ve en el ejemplo mientras se tenga más datos de una sola clase, menor será el valor de las medidas, en cambio si hay un balance de las clases en el nodo, se tendrá para Gini un valor de 0.5 y para la Entropía un valor de 1.

Los árboles de decisión son métodos muy simples, con el tiempo se fueron implementando métodos más robustos basados en árboles, por ejemplo Bagging, Random Forest, Adaboost, Gradient Boosting Machine, Extreme Gradient Boosting, entre otros (James et al., 2021).

3.1.4. Random Forest

Random Forest es un clasificador que consta de una gran colección de árboles de decisión, en los cuales una vez generados se vota por la clase más popular (Breiman, 2001).

Para cada árbol k se genera un vector aleatorio o_k independiente de otros vectores generados pero con la misma distribución. Ahora se genera un árbol con los datos de entrenamiento y o_k, dando como resultado un clasificador $h(x, o_k)$ donde x son los datos de entrada (Breiman, 2001).

Dado un conjunto de clasificadores $h_1(x), h_2(x), \ldots, h_k(x)$ y un conjunto de entrenamiento $\mathbf{X}, \mathbf{Y}$, se define la función de margen como

$$mg(\mathbf{X}, Y) = av_k I(h_k(\mathbf{X}) = Y) - \max_{j \neq Y} av_k I(h_k(\mathbf{X} = j), \tag{3.18}$$

donde I es la función indicadora y av_k es la media. El margen mide los votos de la clase correcta que superan a los de cualquier otra clase. Si el margen es grande, se tendrá más confianza.

Al ser un método basado en árboles de decisión, se ocupan las mediciones de la pureza del nodo para evaluar la división de cada uno. La clase predicha es la que tiene la estimación de probabilidad más alta entre los árboles, se realiza una votación ponderada (Pedregosa et al., 2011). En la Figura 3.10 se muestra el diagrama, para entender mejor cómo funciona el método.

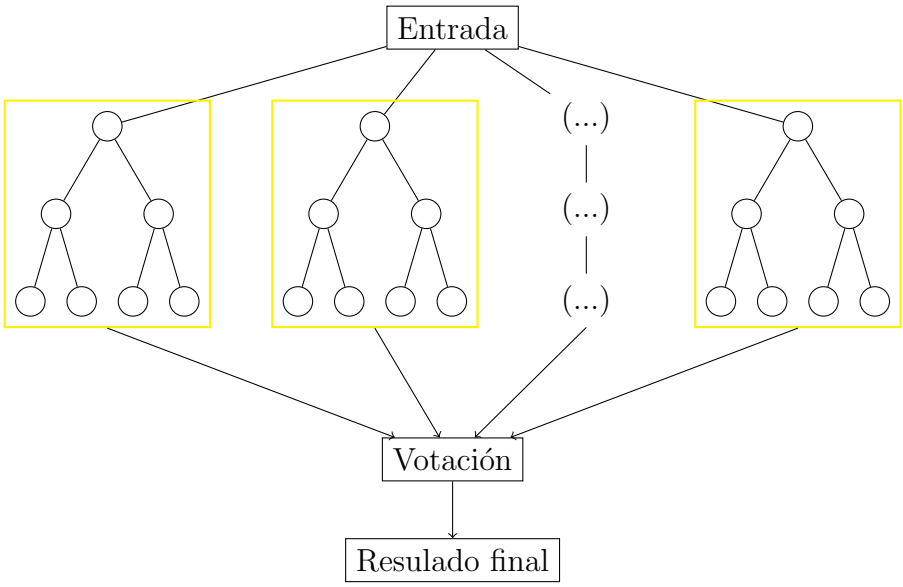


Figura 3.10: Diagrama Random Forest.

En la Figura 3.11, se muestra un ejemplo para predecir si a un usuario le gustara un videojuego, se tienen dos árboles independientes, al final se evalúan las probabilidades, para predecir si le gustara el videojuego o no. La figura fue obtenida de la página web de la librería de XGBoost de (Chen & Guestrin, 2016), donde menciona que la diferencia entre el siguiente método y este, es la manera de entrenar los árboles.

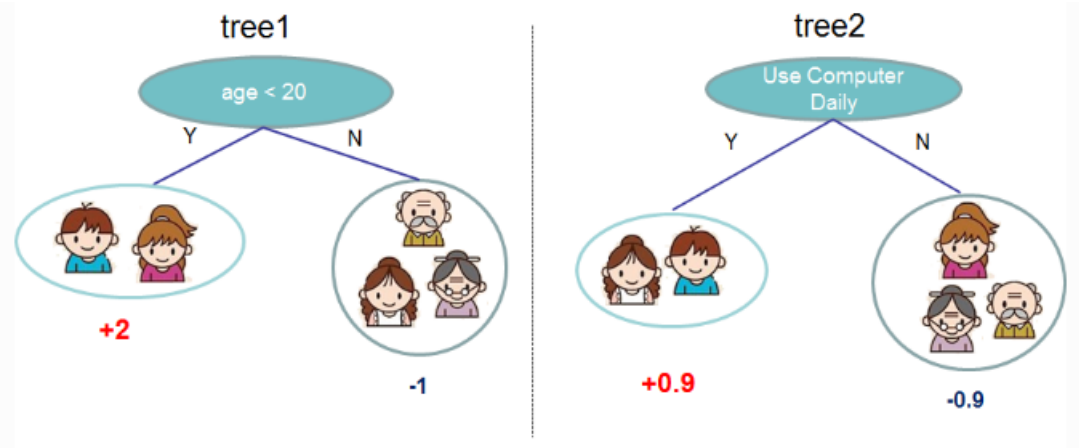


Figura 3.11: Dos conjuntos de árboles.

3.1.5. XGBoost

Gradient Boosting Machine

Para hablar del método Extreme Gradient Boosting (XGBoost) debemos definir el método Gradient Boosting Machine (GBM), el cual se basa en la idea de construir

una serie de árboles de decisión de forma secuencial, donde cada nuevo árbol intenta corregir los errores de los árboles anteriores, esta corrección se realiza mediante los residuales que es la diferencia del valor real de las clases con los valores de las predicciones, Figura 3.12.

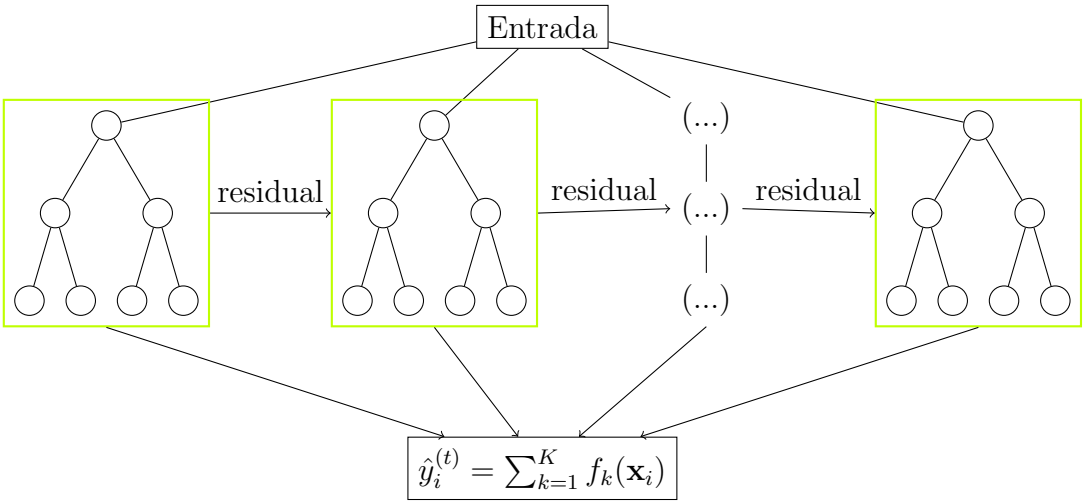


Figura 3.12: Diagrama GBM.

Sea la función f_t la que contiene la estructura de cada árbol, junto a sus puntuaciones y sea $\hat{y}_i^{(t)}$ la predicción de la n-ésima instancia de la n-ésima iteración.

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t), \tag{3.19}$$

Donde l es la función de pérdida del entrenamiento y Ω es el término de regularización que controla la complejidad del modelo, lo ideal es tener un modelo sencillo y que logre predecir los datos nuevos.

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2, \tag{3.20}$$

Donde w es las puntuaciones de las hojas y T es el número de hojas.

Agregando g_i y h_i que son los gradientes de primer y segundo orden de la función de pérdida (l).

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(\mathbf{x}_i)) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i) + \Omega(f_t), \tag{3.21}$$

Eliminamos los términos constantes

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(\mathbf{x}_i)) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i) + \Omega(f_t), \tag{3.22}$$

Definimos $I_j = \{i|q(\mathbf{x}_i = j\}$ como un conjunto de instancias de la hoja j. Expandimos Ω y queda:

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n}[g_i f_t(\mathbf{x}_i)) + \frac{1}{2}h_i f_t^2(\mathbf{x}_i) + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2, \tag{3.23}$$

Calculamos el peso óptimo

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda}, \tag{3.24}$$

Calculamos el valor óptimo correspondiente

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T, \tag{3.25}$$

Esta ecuación puede usarse como la función de puntuación de la calidad del árbol, es similar a la de impureza de los árboles de decisión, excepto que también toma en cuenta la complejidad del modelo.

Normalmente es imposible enumerar todas las estructuras del árbol q. Lo que se hace es que a partir de una hoja se añaden ramas de forma iterativa. Supongamos que I_L y I_R son instancias del nodo izquierdo y derecho, entonces $I = I_L \cup I_R$, ahora la reducción de pérdidas después de una división, está dada por:

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma, \tag{3.26}$$

Esta fórmula se usa para evaluar la división.

Regresando al ejemplo de predecir si a un usuario le gustara un videojuego, en la Figura 3.13 se muestra un árbol, indicando a qué corresponde cada valor requerido para calcular qué tan bueno es el árbol.

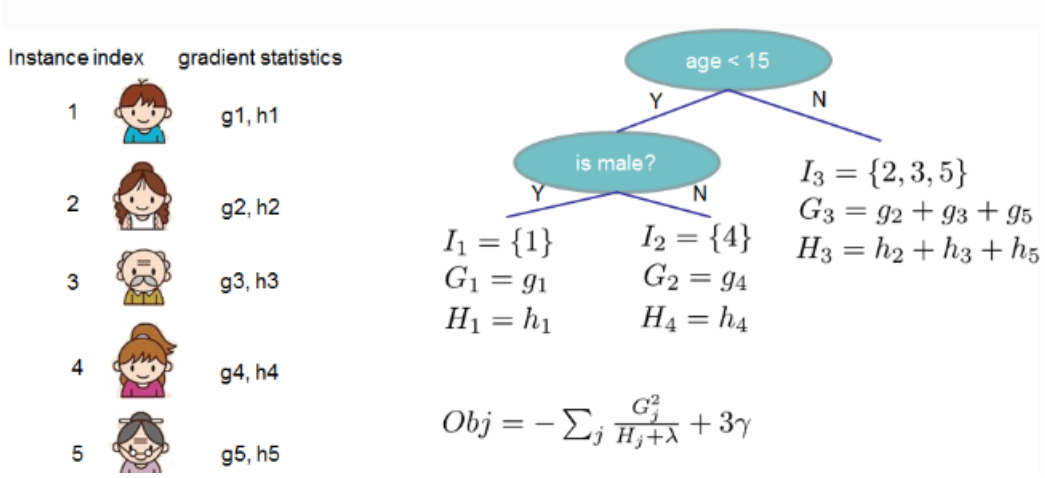


Figura 3.13: Árbol utilizando gradientes.

Reducción y submuestreo

Se reduce la influencia de cada árbol individual, para que los árboles futuros mejoren el modelo (Chen & Guestrin, 2016).

El submuestreo de columnas (características) ayuda a evitar el sobreajuste, mucho más que el submuestreo de filas. También ayuda a acelerar los procesos.

XGBoost

Ahora hablemos sobre el método Extreme Gradient Boosting, el cual es un método basado en GBM, teniendo mayor velocidad computacional y mejor rendimiento, en sus propias palabras busca llevar al extremo los límites computacionales de las máquinas (Chen & Guestrin, 2016). Sus principales innovaciones son manejar datos dispersos, un bosquejo cuantil sensible a los pesos, aprendizaje paralelo el cual hace más rápido el aprendizaje, explota el trabajo fuera del núcleo, lo que permite trabajar con grandes cantidades de datos. Todo esto da como resultado la capacidad de trabajar con grandes conjuntos de datos en un tiempo corto y con menor cantidad de recursos dentro de la máquina.

A continuación se explican mejor las implementaciones.

Búsqueda de divisiones

Algoritmo codicioso exacto

Para elegir la mejor división se enumeran todas las divisiones posibles. A esto lo llamamos *algoritmo codicioso exacto*. Se debe enumerar todas las posibles divisiones. Para ello se debe ordenar primero los datos según las características y ver los datos ordenados para acumular las estadísticas de gradiente para la puntuación de la ecuación (3.26).

Algoritmo aproximado

Calcular todas las posibles divisiones es una buena idea, pero ¿qué pasa cuando no caben en la memoria? XGBoost propone tener divisiones candidatas en función de los percentiles, después el algoritmo mapea las características en secciones divididas, agrega las estadísticas y elige la mejor solución entre las propuestas basadas en estadísticas.

Bosquejo cuantil ponderado

Múltiples conjuntos $\mathcal{D}_k = \{(x_{1k}, h_1), (x_{2k}, h_2) \cdots (x_{nk}, h_n)\}$ representan los k valores de características y las estadísticas del gradiente de segundo orden.

El objetivo es encontrar puntos de división $\{s_{k1}, s_{k2}, \cdots s_{kl}\}$, tales que

$$|r_k(sk, j - r_k(s_{k,j+1})| < \epsilon, \quad sk1 = \min_i \mathbf{x}_{ik}, s_{kl} = \max_i \mathbf{x}_{ik} \tag{3.27}$$

En el rango $r_k : \mathbb{R} \longrightarrow [0, +\infty)$

Donde ϵ es el factor de aproximación, entonces se dice que existen $1/\epsilon$ candidatos, y cada punto está ponderado por h_i. Se puede reescribir la ecuación (3.22), la cual ahora es la pérdida cuadrática ponderada.

$$\sum_{i=1}^{n} \frac{1}{2} h_i (f_t(\mathbf{x}_i - g_i/h_i)^2 + \Omega(f_t) + C, \tag{3.28}$$

Búsqueda de divisiones basada en la dispersión

Sabemos que la mayoría de datos en el mundo real difícilmente tienen datos completos, esta escasez puede depender de varios factores. Este modelo propone que al faltar un valor, la instancia se clasificará en una dirección por defecto óptimo, aprendida a partir de los datos.

Diseño

Bloques para el aprendizaje paralelo

Una parte que llega a consumir bastante tiempo en los métodos basados en árboles es ordenar los datos. XGBoost propone almacenar los datos en *bloques*. Este diseño de datos de entrada solo se calcula una vez antes de entrenar.

Acceso al caché

A pesar de la estructura de bloques, se requiere consultas indirectas de las estadísticas de gradiente. Se trata de un acceso no continuo a la memoria. Esto ralentiza la búsqueda de divisiones cuando las estadísticas ya no caben en el caché y se tienen pérdidas.

Para esto se asigna un búfer interno en cada subproceso, recuperando en el, las estadísticas, realizando la acumulación en mini lotes.

Cálculo fuera del núcleo

Para permitir esto se dividen los datos en bloques y se almacenan.

Compresión de bloques: Se comprimen los bloques por columnas y se descomprime sobre la marcha mediante un subproceso independiente cuando se carga en la memoria principal.

Bloques fragmentados: La segunda técnica consiste en repartir los datos en varios discos. Se tiene un subproceso de prerecarga de datos, así el subproceso de entrenamiento lee los datos que están cargados solamente.

3.2. Aprendizaje en conjunto

Llamamos aprendizaje en conjunto a los procesos donde se combinan múltiples modelos de predicción, existen varias formas de combinar nuestros modelos, las más populares son boosting, bagging, apilamiento y votación (Brownlee, 2021).

Existen clasificadores que están basados en el aprendizaje en conjunto, con boosting se tiene XGBoost, donde el resultado se basa en múltiples árboles y el aprendizaje de cada árbol influye en el siguiente árbol, para obtener un solo resultado.

Con bagging se tiene Random Forest donde el aprendizaje igual se basa en múltiples árboles, la diferencia aquí es que cada árbol es independiente del otro, el resultado se obtiene de forma iterativa con los resultados individuales de cada árbol.

Tanto la votación como el apilamiento son llamados metamodelos, ya que aprenden de otros algoritmos, sin interferir con ellos ya que se aplican después de su entrenamiento e implementación, no requieren que los predictores se basen en la misma metodología, aquí podemos tener múltiples modelos cada uno con diferentes habilidades y criterios para clasificar.

La votación realiza una serie de votos "*estricta*" o "*suave*", en estricta se elige al que tiene mayor número de votos y en suave se suma las probabilidades para elegir el resultado.

El apilamiento es un poco más robusto, ya que se utiliza una doble clasificación, donde se tiene a los modelos base, y a partir de ellos se tiene otro modelo que combina las predicciones de los modelos base, puede ser con regresión logística, árboles de decisión, entre otros, y así obtener un modelo final.

Capítulo 4

Clases desbalanceadas

4.1. ¿Qué son las clases desbalanceadas?

Con clases desbalanceadas nos referimos a problemas donde las clases no tienen el mismo número de muestras, donde se tiene una clase mayoritaria y una clase minoritaria, podemos tener un desequilibrio ligero o un desequilibrio severo donde por cada muestra de la clase minoritaria hay cientos, miles o millones de muestras de la clase mayoritaria (Brownlee, 2020).

Esto supone un problema ya que la mayoría de modelos de ML suponen que las clases están balanceadas y al implementar estos algoritmos le es más difícil predecir a la clase minoritaria, ya que es más difícil aprender las características de esta clase y diferenciarla con la otra clase. En muchos casos la clase minoritaria es la de mayor interés.

Existen múltiples maneras de tratar estos datos, lo recomendable es probar que le viene mejor a nuestros datos.

4.2. Evaluación

4.2.1. Matriz de confusión

La matriz de confusión es una comparación entre las clases reales y las clases que fueron predichas por nuestro modelo.

En la Figura 4.1 se muestra una matriz de confusión para un problema binario, la clase positiva y la clase negativa, en la diagonal principal están los datos que fueron clasificados correctamente, los llamamos Verdadero Negativo o Verdadero Positivo y a los datos que no fueron clasificados correctamente se les llama Falso Negativo y Falso Positivo.

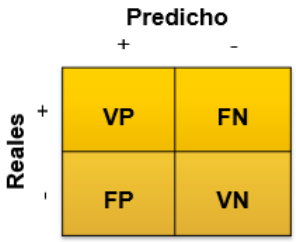


Figura 4.1: Matriz de confusión para un problema binario.

4.2.2. Métricas de evaluación

Exactitud

Esta métrica suele ser usada con frecuencia para evaluar el desempeño de nuestro modelo, pero para datos desequilibrados no es conveniente, ya que nos puede dar un resultado muy bueno, que en realidad no es confiable, debido a que no le interesa el desempeño por clase, si no general.

$$\text{Accuracy} = \frac{vp + vn}{vp + vn + fp + fn}, \tag{4.1}$$

Existen métricas que se centran en el desempeño de cada clase y nos ayudan a saber cómo está clasificando nuestro modelo las distintas clases, estas métricas son Precision, Recall y F1.

Precision

De todas las predicciones positivas de una clase, ¿cuántas son verdaderas?

$$\text{Precision} = \frac{vp}{vp + fp}, \tag{4.2}$$

Recall

De todos los datos correctos de una clase, ¿cuántos se predijeron correctamente?

$$\text{Recall} = \frac{vp}{vp + fn}, \tag{4.3}$$

F1

La medida F1 es una combinación de las dos métricas pasadas, donde se expresa el resultado de ambas.

$$\text{F1} = 2 * \frac{(\text{Precision})(\text{Recall})}{\text{Precision} + \text{Recall}}, \tag{4.4}$$

Ahora para medir el desempeño del modelo aplicado en este proyecto:

Macro promedio

Se define como la media tratando a las clases con la misma importancia. Por ejemplo el macro promedio para la medida F1 es:

Donde:
C es Clase

$$\text{Macro avarage} \quad F1 = \frac{F1_{c1} + F1_{c2} + ... + F1_{cn}}{numero \quad de \quad clases}, \tag{4.5}$$

4.3. Remuestreo

Existen múltiples técnicas para remuestrear nuestros datos que van desde las más simples hasta las más complejas; se puede sobremuestrear la clase minoritaria o submuestrear la clase mayoritaria, o también se puede hacer una combinación de sobremuestreo y submuestreo (Brownlee, 2020).

En la Figura 4.2 tenemos una distribución de 2 clases desbalanceadas, la clase 0 con 901 datos y la clase 1 con 99.

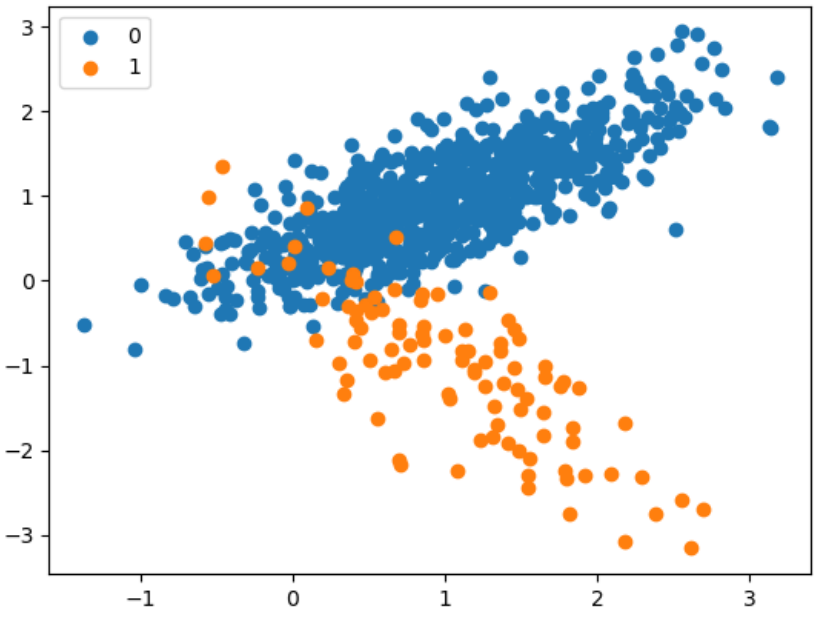


Figura 4.2: Distribución binaria desbalanceada.

A continuación se describen algunas de las técnicas para remuestrear, aunque no son todas, son las que se probaron en este proyecto.

4.3.1. Sobremuestreo

SMOTE

La Técnica de Sobremuestreo Minoritario Sintético ("SMOTE", por sus siglas en inglés) fue creada en 2002. Anteriormente una forma de sobremuestrear los datos era por reemplazo de los datos originales pero esta forma no ayuda significativamente a los datos, y en algunos casos no tiene ninguna mejora.

(Chawla et al., 2002) propusieron un nuevo enfoque donde en vez de reemplazar se crean nuevos datos, estos se crean a partir de las muestras de la clase minoritaria, a lo largo de una línea imaginaria que los une con otras muestras de la misma clase a partir de sus vecinos cercanos.

En resumen, la creación de datos sintéticos con SMOTE ayuda a crear una región de decisión más grande para la clase minoritaria, y así no pasar desapercibida al momento de clasificar.

En la Figura 4.3 se tienen los mismos datos de la Figura 4.2 sobremuestreada con la técnica SMOTE, ahora con una distribución de la clase 0 con 901 datos y la clase 1 con 901 datos.

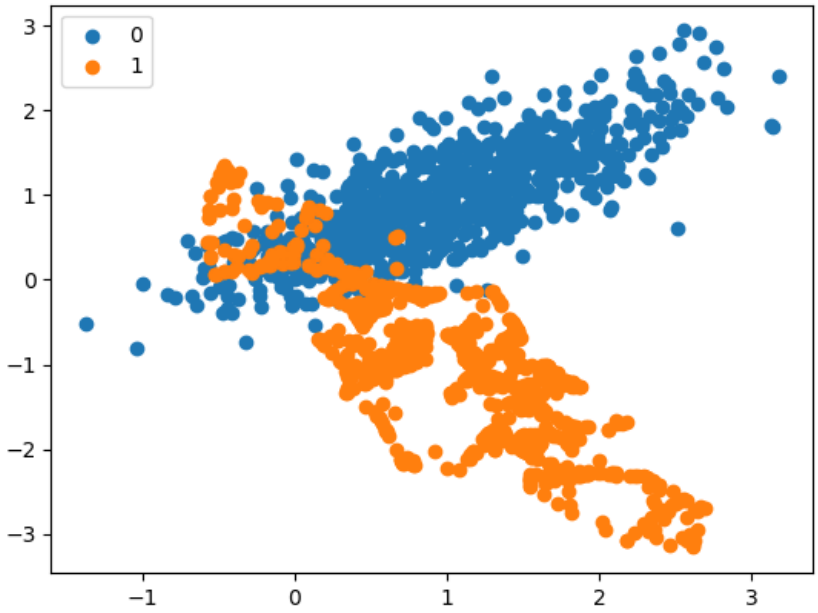


Figura 4.3: Distribución binaria sobremuestreada con la técnica SMOTE.

4.3.2. Submuestreo

CNN

La técnica Vecinos Cercanos Condensados ("CNN", por sus siglas en inglés), está basada en la regla de vecinos cercanos, el objetivo de este método es reducir el tamaño de los datos eliminando determinadas muestras sin afectar el rendimiento de la clasificación, con la regla de vecinos cercanos se clasifica todos los datos y se manda a "STORE" o "GRABBAG", finalmente los datos retenidos en "GRABBAG" son

introducidos en un bucle para decidir si son transferidos a "STORE" o se quedan, finalmente los que se quedan retenidos en "GRABBAG" son descartados (Hart, 1968).

Tomek Links

Tomek presenta dos desventajas del método CNN, las cuales son retención de muestras innecesarias y retención de muestras internas en lugar de muestras en los límites entre las clases, propone una modificación al método CNN que se enfoca solamente en los límites entre ambas clases, en lugar de todo el conjunto (Tomek, 1976).

En la Figura 4.4 se tienen los mismos datos de la Figura 4.2 submuestreada con una distribución de la clase 0 con 892 datos y la clase 1 con 99 datos.

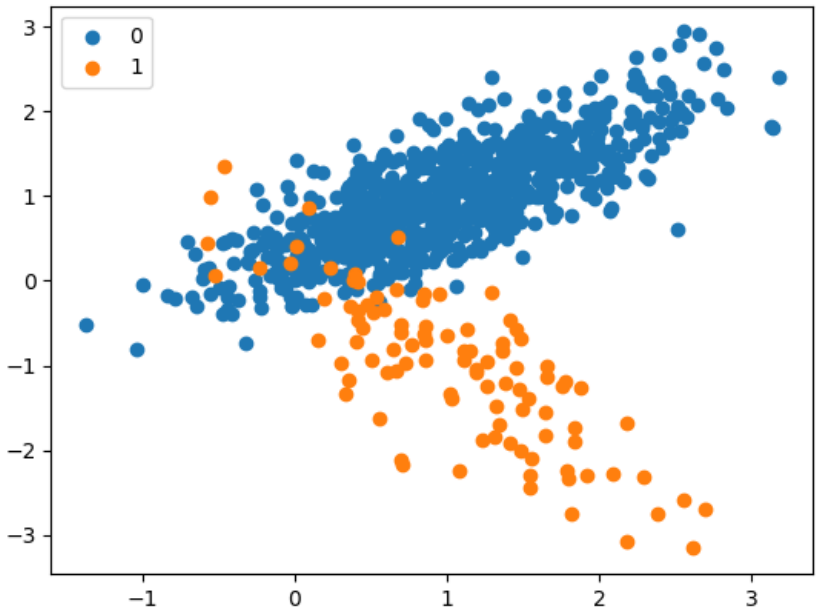


Figura 4.4: Distribución binaria submuestreada con la técnica Tomek Links.

ENN

La técnica Vecinos Cercanos Editados ("ENN", por sus siglas en inglés), submuestrea la clase mayoritaria. Este método elimina "ejemplos ruidosos" que parte de sus vecinos cercanos no pertenecen a su misma clase (Brownlee, 2020).

En la Figura 4.5 se tienen los mismos datos de la Figura 4.2 submuestreada con una distribución de la clase 0 con 862 datos y la clase 1 con 99 datos.

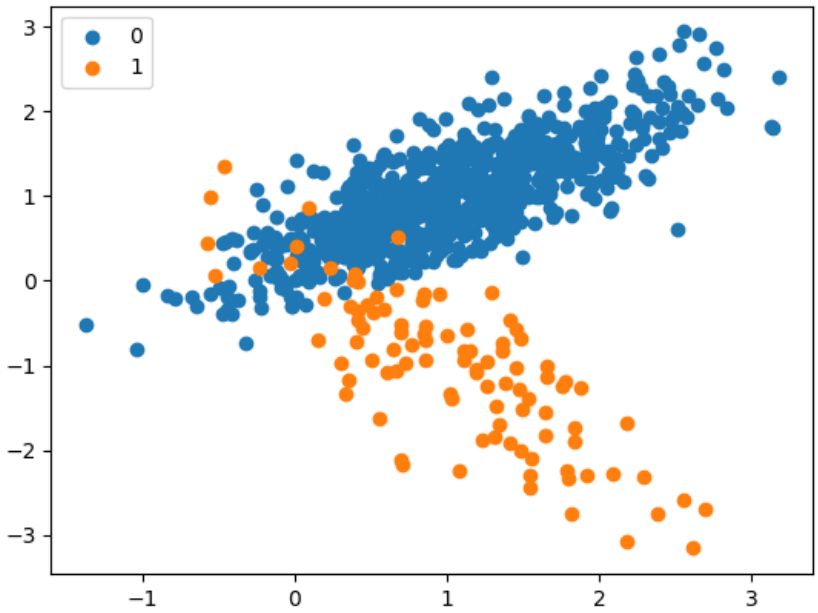


Figura 4.5: Distribución binaria sobremuestreada con la técnica ENN.

4.4. Pesos

Algunos métodos de ML son sensibles a los pesos, y se pueden ajustar para que se preocupen más por clasificar bien la clase minoritaria que por la mayoritaria.

Con esto se refiere, que al tener menor número de muestras mayor será el peso y mayor penalización se tendrá al clasificar incorrectamente esta clase. En muchos casos la clase minoritaria es la de mayor interés, aunque es pequeña es más importante, por ejemplo, en la detección de alguna enfermedad, si el clasificador es sensible a los pesos, la clase enferma tendrá más importancia.

Estos pesos se calculan y se dan los pesos a cada clase al momento de entrenar, o bien algunos modelos tienen la opción de pesos "balanceados".

```
clasificador = sklearn.svm.SVC (class_weight = ↩
    'balanced')
```

Código 4.1: Pesos en método Support Vector Machine.

Capítulo 5

Aplicación

El conjunto de datos utilizado en este proyecto, son 6 pozos (Pozo 0, Pozo 1, Pozo 1b, Pozo 2, Pozo 4 y Pozo 6) con sus respectivas facies. Los métodos de aprendizaje supervisado utilizados son Support Vector Machine, Random Forest, K-Nearest Neighbor y XGBoost.

El flujo de trabajo será el siguiente:

1. Edición de datos.
2. Exploración y visualización de datos.
3. Limpieza de datos.
4. Ingeniería de características.
5. Preprocesamiento.
6. Tratamiento clases desbalanceadas.
7. Entrenamiento.
8. Implementación.
9. Post-Procesamiento
10. Evaluación del modelo de clasificación.
11. Aplicación del modelo de clasificación.

En la Figura 5.1 se tiene el diagrama de flujo para el proyecto.

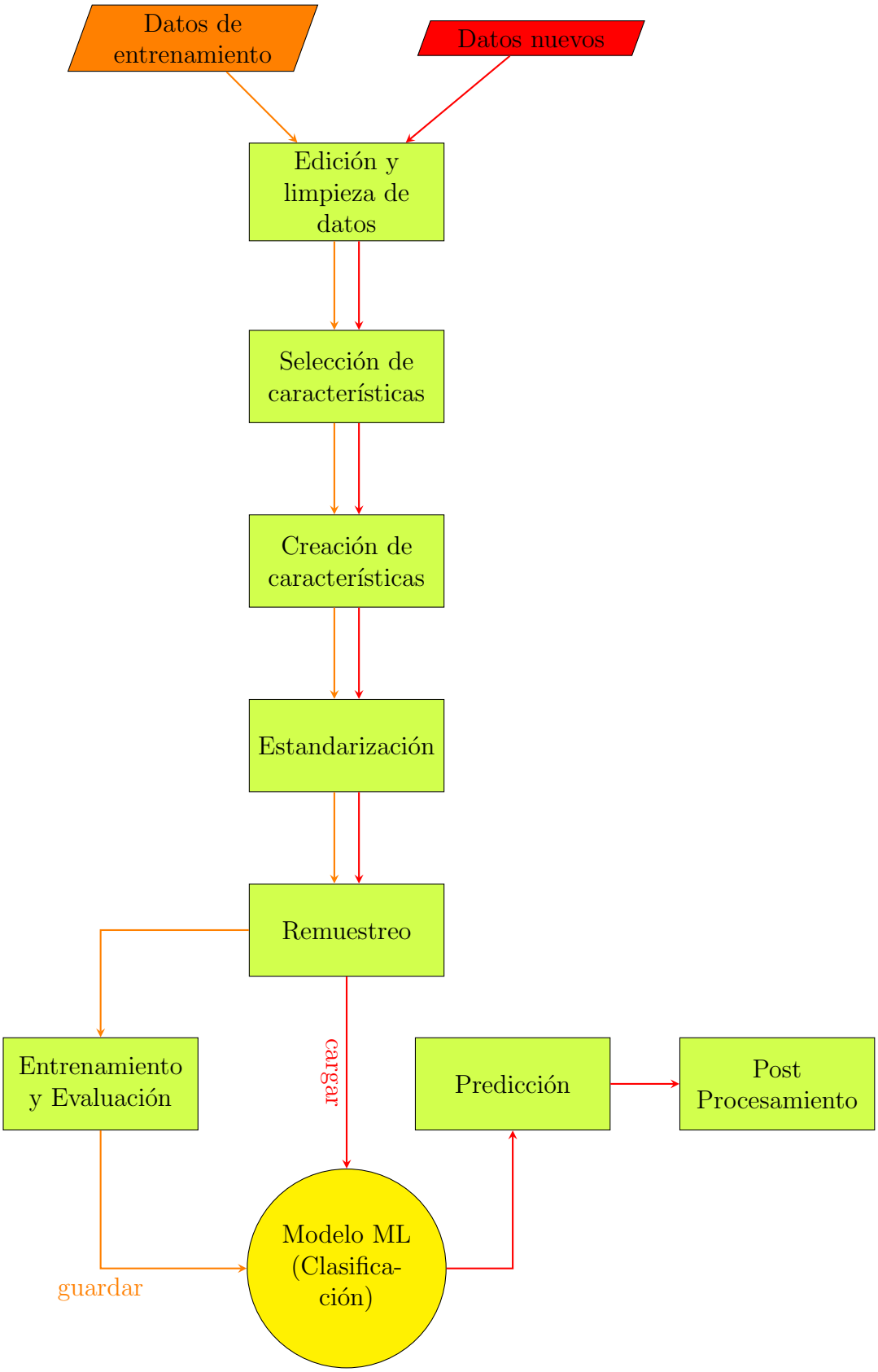


Figura 5.1: Diagrama de flujo de los modelos de ML.

5.1. Edición de datos

La edición de datos es la más sencilla, pero a su vez más tardada si no se tiene un software para ello, una buena alternativa es hacerlo en Python. Los datos anómalos producidos por agentes externos al medio geológico, pueden generar singularidades en el proceso de clasificación produciendo errores en los resultados. Esta edición se considera al inicio porque tiene que ver con los archivos .las.

Las irregularidades encontradas en los datos son las siguientes:

1. Al trabajar con pozos siempre es importante revisar el encabezado de estos, en este caso se tenían dos archivos para cada pozo, uno que contenía los registros y otro con las facies. En la Figura 5.2 el intervalo de muestreo para cada pozo no es el mismo, para este caso es sencilla la corrección ya que uno era múltiplo del otro.

```
STEP .M              0.15240:  STEP
```

(a) Para los registros de pozos

```
STEP  .M                0.07620 |                  : STEP OF INDEX
```

(b) Para los datos de facies

Figura 5.2: Intervalos de muestreo.

2. Así como es importante leer el encabezado, también es importante ver cómo se ven los registros graficados. Todos los pozos usados en este proyecto, tienen en su encabezado como valor nulo el valor -999.25, en la Figura 5.3, se muestra como viene en los pozos el valor nulo. Los pozos se cargaron en su formato original con lasio que es una librería de Python de lectura y escritura de archivos .las, al leer el encabezado, por default todos los -999.25 los toma como valor nulo (en lenguaje de Python "NaN"). Al graficar los registros de cada uno de los pozos se observó que había un valor de -9999.25 y lo consideraba como un valor de medición, entonces alteraba los registros al momento de graficar y posteriormente afectaría el entrenamiento, se modificó a -999.25.

```
~Well Information Block
#MNEM.UNIT        Data Type    Information
#---------- ----------------  -----------
STRT .M           204.06360:  START DEPTH
STOP .M          5011.06440:  STOP DEPTH
STEP .M             0.15240:  STEP
NULL .               -999.25:  NULL VALUE
```

Figura 5.3: Encabezado del pozo 1.

3. El archivo .las que contenía las facies, las tenía en formato numérico y categórico, los valores categóricos tenían errores de ortografía, lo que quiere decir que

al momento de contar el número de clases en el formato categórico nos generaba 6 clases, cuando en realidad eran 5 clases, ya que una sola clase la separaba en dos clases, las que estaban escritas correctamente y las que no.

Para los humanos es fácil saber que es un simple error, pero para las máquinas no es así, se debe tener los datos más precisos y limpios posibles para el proyecto a desarrollar.

5.2. Exploración y visualización de datos

Ahora con los datos editados, sigue examinar el conjunto de datos que se utilizarán para entrenar el clasificador. En la terminología de ML, cada medición de registro es un vector de características que asigna un conjunto de características (las mediciones de registro) a una clase (el tipo de facie).

Siguiendo la configuración de (Bestagini et al., 2017) se considera que en cada profundidad p de cada pozo w, tenemos un conjunto de 6 características, las cuales son:

- Lentitud de onda de compresión ($f_{p,w}^{\mathbf{DTCO}}$)
- Lentitud de onda de corte ($f_{p,w}^{\mathbf{DTSH}}$)
- Rayos gamma ($f_{p,w}^{\mathbf{GR}}$)
- Porosidad - Neutrón ($f_{p,w}^{\mathbf{NPHI}}$)
- Densidad ($f_{p,w}^{\mathbf{RHOB}}$)
- Porosidad efectiva ($f_{p,w}^{\mathbf{PHIE}}$)

A cada profundidad se tiene un valor para cada característica, que de igual manera está asociada a un pozo.

$$\mathbf{f}_{p,w} = [(f_{p,w}^{\mathbf{DTCO}}), (f_{p,w}^{\mathbf{DTSH}}), (f_{p,w}^{\mathbf{GR}}), (f_{p,w}^{\mathbf{NPHI}}), (f_{p,w}^{\mathbf{RHOB}}), (f_{p,w}^{\mathbf{PHIE}})] \tag{5.1}$$

Se tienen 5 facies asociadas también a una profundidad y a un pozo, se definen como $c_{p,w} \in \{$Calizas, Margas, Dolomías, Intercalación C-D y Calizas oscuras$\}$, definir estas funciones ayuda a visualizar mejor el proceso a realizar. En la Figura 5.4 se muestran las 5 clases.

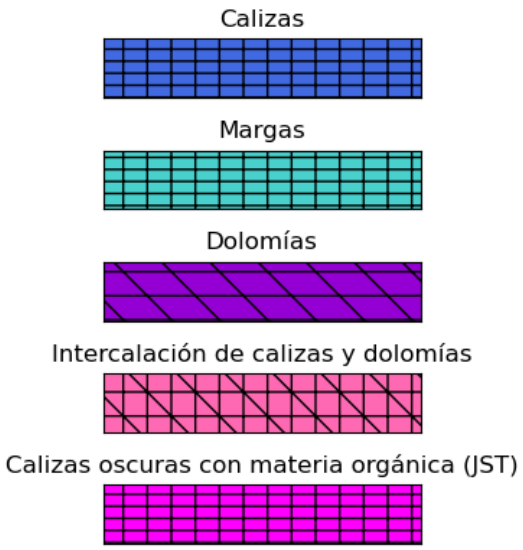


Figura 5.4: Clases.

Para el entrenamiento del clasificador se utilizaron los pozos 4 y 6. En la Tabla 5.1 se tiene la distribución de las clases para el conjunto de datos de entrenamiento, obsérvese que las margas en este caso son la clase minoritaria, seguido de las dolomías, en cambio las calizas son la clase mayoritaria, como se vio en el Capítulo 5 se tiene un desequilibrio ligero, que más adelante se tratará.

Calizas	39.4 %
Margas	2.5 %
Dolomías	12.7 %
Intercalación calizas y dolomías	15 %
Calizas oscuras con materia orgánica	30.3 %

Tabla 5.1: Distribución de las clases.

La matriz de dispersión es una herramienta para visualizar cómo cambian dos mediciones de registros con el tipo de roca, en nuestro caso con el tipo de facies, de igual forma ayuda a comprender el comportamiento de las clases y la relación entre ellas.

En la Figura 5.5 se muestran graficados todos los registros utilizados , para los datos de entrenamiento, se observa que las clases tienen un comportamiento similar, no tan disperso, aunque no es muy clara la separación entre ellas y en algunos casos llegan a estar encimadas.

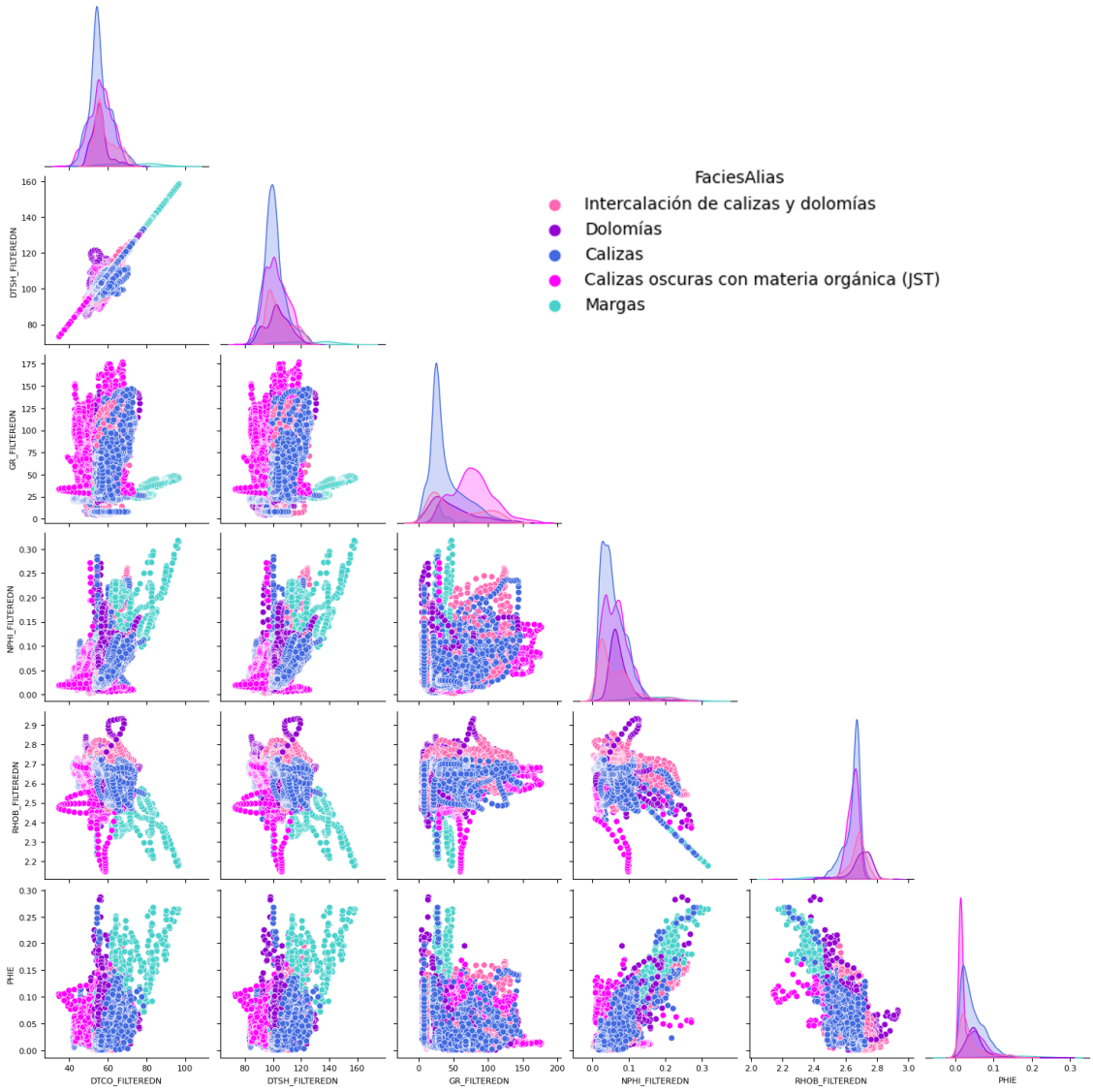


Figura 5.5: Matriz de dispersión entre los registros.

5.3. Limpieza de datos

Al desarrollar un modelo de ML, la mayor parte del tiempo se dedica a la preparación de los datos, que va desde la limpieza de datos, la ingeniería de características y el preprocesamiento de los datos.

En un curso de ML de Google (Google, s.f.), se hace énfasis, que la calidad y el tamaño del conjunto de datos son mucho más importantes que el algoritmo más brillante que uses.

Dentro de la limpieza de datos, se edita o elimina tanto los datos nulos como los datos que puedan generar ruido, en nuestro caso, en la zona de estudio que abarca el Cretácico, Jurásico Superior Tithoniano y Jurásico Superior Kimmerdigiano, no había datos nulos.

Los métodos indirectos como los registros geofísicos de pozos, por diferentes factores se pueden tener mediciones fuera de lo normal, como se ve en la Figura 5.5, los gráficos cruzados nos ayudan a visualizar la relación entre los datos, pero otro punto importante es que también se observan los datos anómalos que pueden generar ruido, muchas veces estos valores se dan por fracturas o en el caso del registro de densidad por derrumbes y en otros casos son ruido por parte del aparato de medición, aunque son datos reales, lo mejor es eliminarlos para el buen desarrollo del modelo.

Para tener mejor panorama en la limpieza de datos, se realizó la matriz de dispersión por cada pozo utilizado para entrenar, aunque algunos de estos datos anómalos se ven desde la Figura 5.5.

5.4. Ingeniería de características

La ingeniería de características es uno de los pasos más importantes en el desarrollo de un modelo de ML, ya que ayuda mucho al buen desempeño de nuestro modelo.

5.4.1. Selección de datos

Se tiene el conjunto de datos el cual consta de 6 pozos, se deben separar en el conjunto de entrenamiento y el conjunto de prueba, aunque desde el inicio se comentó que se utilizaría el pozo 4 y el pozo 6 para entrenar, este paso forma parte de la ingeniería de características. Para separar los datos lo mejor es elegir un conjunto de entrenamiento que logre generalizar bien los datos y que tenga un mejor balance de los datos donde se incluyan todas las clases.

5.4.2. Selección de características

Existen diferentes formas de realizar la selección de características, en el curso de Google (Google, s.f.), recomienda empezar el desarrollo de un modelo con pocas características e ir aumentando poco a poco, así sabremos qué características ayudan al modelo a encontrar una buena relación de los datos.

Los registros utilizados para la asignación de facies son el registro de Rayos Gamma, el registro de Porosidad-Neutrón, el registro de Densidad y los registros de Lentitud, en este caso el registro de Porosidad efectiva no era tan importante, sin embargo debemos analizar cómo afectan estos registros en el modelo de clasificación.

Otra forma de realizar la selección de características es calcular el coeficiente de correlación, el cual indica qué tan relacionadas linealmente están dos variables, existen tres formas de medir esta correlación, las cuales son el coeficiente de correlación de Pearson, el de Spearman y el de Kendall (Dagnino, 2014). Para el proyecto se utilizó la de Pearson, la cual es calculada como:

$$r = \frac{\sum XY}{\sqrt{\sum X^2 \sum Y^2}}, \tag{5.2}$$

Se realizó una matriz de correlación de Pearson entre los datos, representada mediante un mapa de calor, Figura 5.6, con el objetivo de discretizar las funciones a usar en el proyecto. Saber la correlación entre funciones es importante debido a que veces se cree que en la creación de modelos de ML, la redundancia de datos es beneficiosa para el modelo, pero no siempre es así, muchas veces esa repetición de datos hace que el algoritmo se sobreajuste y no generalice bien los datos, y por otra parte estos datos redundantes nos ocupan mayor espacio en el conjunto datos y en consecuencia mayor tiempo de cómputo.

En la Figura 5.6, se observa que los registros de lentitud de onda de compresión y lentitud de onda de corte, tienen una alta correlación entre ellos, se consideró quitar alguno, pero la precisión de los resultados disminuía un 5 %. Finalmente, se decidió usar los 6 registros para el entrenamiento del modelo, se debe considerar tanto la puntuación como el tiempo de cómputo, al no ser muchas características, se eligió una mayor puntuación aunque el tiempo de cómputo aumente.

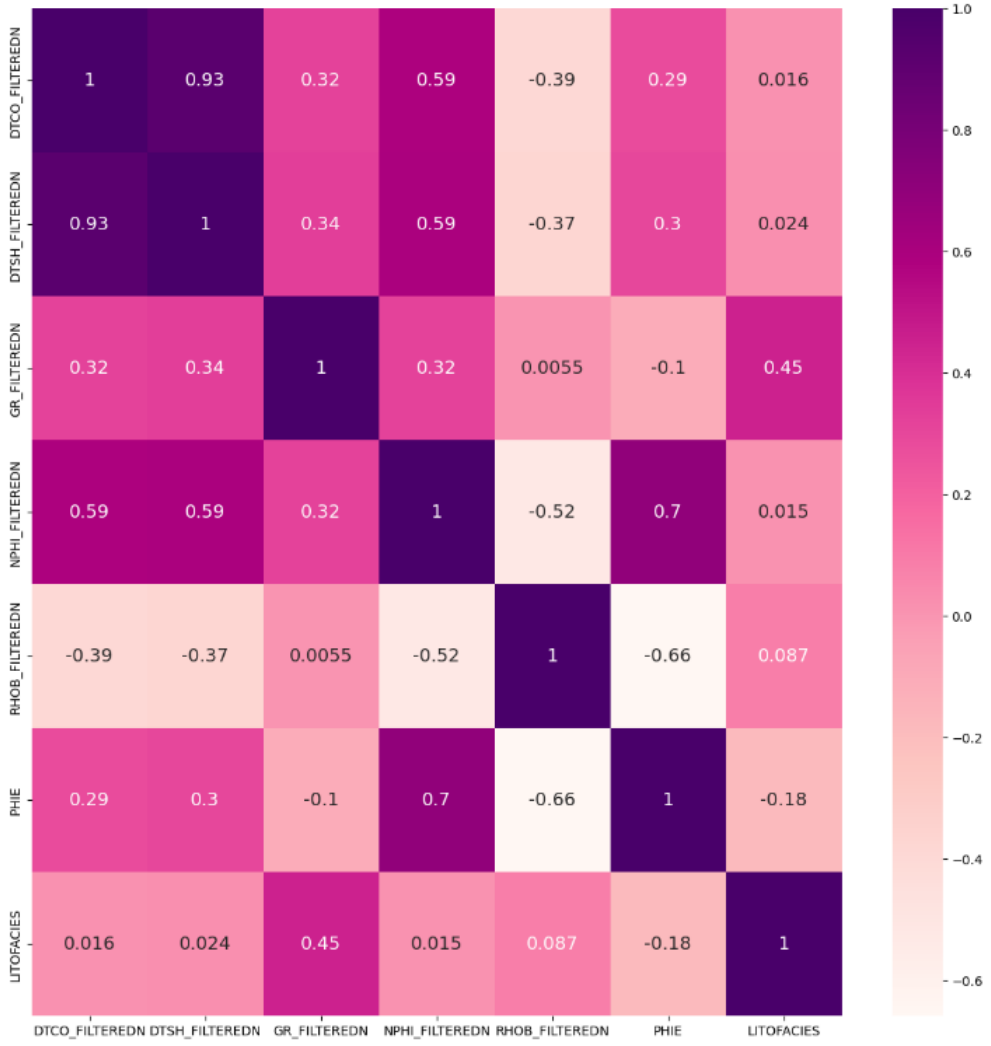


Figura 5.6: Mapa de color de la matriz de correlación de los registros y litofacies.

5.4.3. Generación de características

Al escuchar generación de características muchas veces se tiene la idea errónea que es duplicar otras características, pero no, ya que entonces no tendría sentido el paso anterior donde se comenta que la redundancia no siempre es buena. La idea de generación de características es que, a partir de los datos, se puede obtener nuevos datos que nos ayuden a encontrar una mejor relación entre ellos. Como se ve en la Figura 5.5, las facies están mezcladas entre sí, y la mayoría de ellas no pueden separarse linealmente.

Se realizó la creación de características con algunas líneas del notebook de Paolo Bestagini (Bestagini et al., 2017).

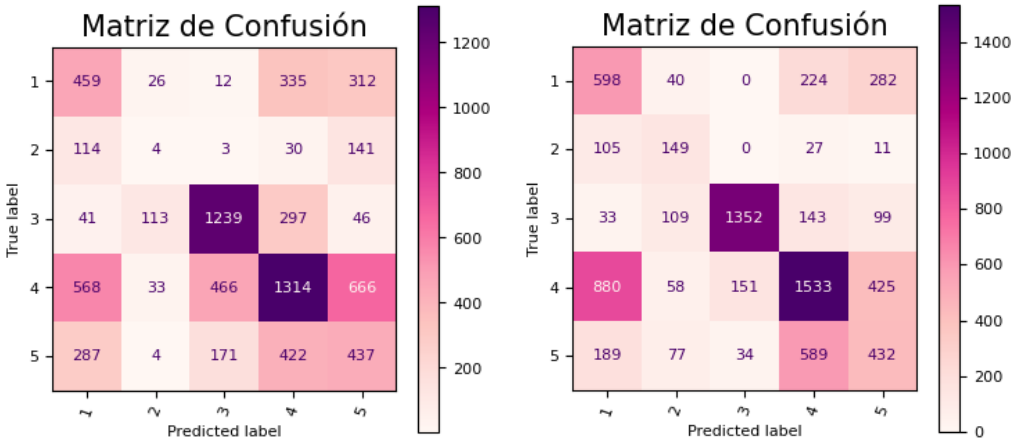
Además de estas características creadas, con la librería scikit-learn, se crean otras características con la función

```
sklearn.preprocessing.PolynomialFeatures(degree=2)
```

Código 5.1: Generación de características polinómicas.

La cual genera nuevas características que constan de todas las combinaciones polinómicas de las características. Por ejemplo, si los datos de entrada tienen la forma $[a, b]$, las características polinómicas de grado 2 son $[1, a, b, a^2, ab, b^2]$.

En la Figura 5.7 se tiene una comparación de las matrices de confusión, una sin crear características y otra con creación de características, ambas para un mismo modelo, se ve un buen incremento de predicciones correctas con la creación de características, en el cual la puntuación aumentó un 10 % .



(a) Sin creación de características. (b) Con creación de características.

Figura 5.7: Comparación pozo 1b con el modelo de SVM.

5.5. Preprocesamiento

Cuando se habla de preprocesamiento se refiere a todas las transformaciones que se aplican a los datos antes de entrenar.

Al medir diferentes propiedades de la roca, cada registro está en una escala diferente, lo ideal para los métodos de ML que se basan en la distancia, es transformar los datos a una misma escala para que el modelo logre encontrar una mejor relación, se puede realizar una estandarización o normalización.

5.5.1. Estandarización

Muchos algoritmos de ML requieren que las características sean datos estándar distribuidos normalmente. Los factores utilizados para estandarizar el conjunto de entrenamiento, también deben aplicarse a cualquier conjunto de características a clasificar.

Con estandarizar se refiere a dar una varianza unitaria y una media cero a los datos, es decir, darles una distribución normal a los datos.

Se puede estandarizar los datos con la función de la librería scikit-learn

```
sklearn.preprocessing.StandardScaler()
```

Código 5.2: Estandarización.

al inicio se usó esta estandarización, aunque ya se realizó un paso de eliminación de ruido, se pudo pasar por alto algún dato, aquí es donde entra el siguiente escalador.

RobustScaler

Escalar funciones utilizando estadísticas sólidas frente a valores atípicos (Pedregosa et al., 2011), esto es lo que hace RobustScaler.

Se basa en quantiles, y no toma en cuenta los datos anómalos, ya que estos influyen de manera negativa al momento de estandarizar. Podemos llamarlo de la siguiente forma:

```
sklearn.preprocessing.RobustScaler()
```

Código 5.3: Estandarización robusta.

5.6. Tratamiento clases desbalanceadas

Como se vió en el Capítulo 4 se debe aplicar algunos métodos para poder equilibrar el desbalance de clases, para los métodos de Support Vector Machine, K-Nearest Neighbor y XGBoost, se utilizó una combinación de sobremuestreo con SMOTE y submuestreo con ENN, con la librería imbalanced-learn

```
imblearn.combine.SMOTEENN()
```

Código 5.4: Tecnica SMOTE y ENN.

para el método de Random Forest, se utilizó el parámetro que equilibra los pesos de las clases

```
clasificador = ↩
   sklearn.ensemble.RandomForestClassifier( ↩
   class_weight = 'balanced')
```

Código 5.5: Random Forest Clasificador con pesos balanceados.

5.7. Fuga de datos

La fuga de datos (Data Leakage) es un tema del cual no hay mucha información en los libros o artículos relacionados a ML, pero el cual también es muy importante. Como su nombre lo dice es cuando los datos empiezan a fugarse, en este proyecto al realizar las pruebas del modelo, se obtenían resultados mayores al 90 % de precisión con los datos de prueba, y al tratar de implementar el modelo y aplicarlo a nuevos datos, el resultado tenía una precisión menor al 20 %, es decir el modelo no había aprendido nada y arrojaba resultados erróneos. Lo que ocurre aquí es que se tiene una fuga de datos, ya que primero se realizó la estandarización, después la separación de los datos en datos de entrenamiento y datos de prueba, y por último se entrenaba, al probar el modelo con los datos de prueba, el algoritmo reconoce esos datos ya que ya había trabajado con ellos, aunque solo fuera en el preprocesamiento.

Lo que se debe hacer, es aplicar los procesos primero a los datos de entrenamiento y después a los datos de prueba, esto se puede con la función Pipeline de la librería de scikit-learn, el cual genera una "tubería" donde se canalizan todos los procesos, evitando la fuga datos y poder seguir un orden, asegurando que todos los datos pasen por el mismo proceso. En la Figura 5.8 podemos ver el proceso para el modelo de KNN, primero la estandarización, seguido de creación de características polinómicas, seguido de 2 técnicas de remuestreo y finalizando con el clasificador KNeighbors Classifier.

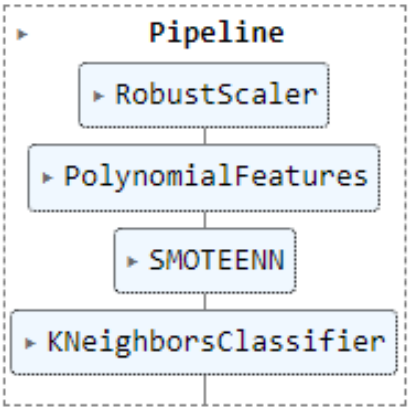


Figura 5.8: Pipeline para el modelo K-Nearest Neighbor.

5.8. Entrenamiento del clasificador

5.8.1. Sobreajuste y subajuste

Un tema importante dentro del ML es el ajuste del modelo, ya que el objetivo de un modelo es que logre generalizar bien los datos de entrenamiento, para poder aplicarlo a futuro con datos nuevos.

Sobreajuste

El sobreajuste se da cuando el modelo se ajusta demasiado a los datos de entrenamiento, y al momento de probarlo con nuevos datos no es capaz de generalizarlos correctamente. Un factor importante para saber si se tiene sobreajuste es tener una buena respuesta al entrenar pero un mal desempeño con datos nuevos. Existen muchos factores que afectan, por ejemplo el desbalance de las clases, el ruido de los datos de entrenamiento y los valores que asignamos a los hiperparámetros.

Subajuste

El subajuste se da cuando el modelo no logra encontrar una relación entre los datos de entrenamiento, por lo que no es capaz de clasificar correctamente, las causas más comunes de este problema es la falta de datos y la mala asignación de los hiperparámetros.

5.8.2. Validación Cruzada

La validación cruzada k-fold, divide el conjunto de datos en k pliegues, en la Figura 5.9, el proceso de entrenamiento se realiza 5 veces donde en cada iteración un pliegue es el conjunto de validación, aunque es un proceso que ocupa mayor tiempo computacional, se obtienen buenos resultados ya que se ocupa todo el conjunto para entrenar pero también para validar, claro en diferentes momentos, en cambio si se aplica una separación aleatoria de los datos, donde un porcentaje del conjunto de datos es el conjunto de prueba, se está a la deriva de si el segmento con el que se está entrenando es el que mejor se ajusta al modelo. La validación cruzada "k" da un mayor alcance para encontrar una mejor relación entre el conjunto de datos.

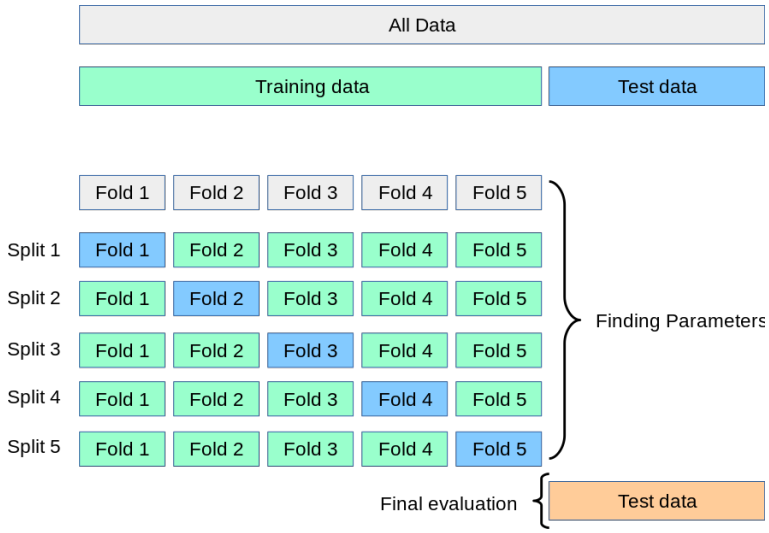


Figura 5.9: Validación cruzada para 5 pliegues. (Pedregosa et al., 2011).

Debido a que en el proyecto se tienen datos desbalanceados, para entrenar el clasificador se utilizó la validación cruzada estratificada con 5 pliegues, con la librería de scikit-learn

```
sklearn.model_selection.StratifiedKFold(n_splits=5)
```

Código 5.6: División de los datos.

esto asegura que en todas las divisiones se tendrá el mismo número de muestras por clase.

5.9. Implementación

Para hablar de implementación de un modelo se deben definir dos conceptos, los cuales son parámetros e hiperparámetros. Los parámetros son valores que no define el programador, si no el conjunto de datos, por ejemplo, los vectores de soporte del método SVM. Los hiperparámetros en cambio son valores que el programador puede ajustar manualmente, por ejemplo, el hiperparámetro C de SVM.

Hasta ahora, el clasificador se ha construido con los hiperparámetros por defecto. Sin embargo, se puede obtener mejores resultados de clasificación con la elección de hiperparámetros óptimos.

Para los 4 modelos se realizó una búsqueda probando diferentes valores, con la función GridSearchCV de la librería scikit-learn, a continuación, se muestra un ejemplo para el modelo SVM

```
hiperparametros = {'C' : [1,10,100,1000], 'gamma' ↩
   : ['scale', 'auto']}
sklearn.model_selection.GridSearchCV(SVM_modelo, ↩
   hiperparametros, validacioncruzada, puntuacion)
```

Código 5.7: Búsqueda de hiperparámetros con validación cruzada.

la búsqueda consta de un estimador que es el modelo a usar, los hiperparámetros que se definieron arriba, lo cual da 8 combinaciones, se realiza una validación cruzada para todas las combinaciones posibles y la puntuación es la medida que se usará para medir el rendimiento del modelo, al finalizar todas las iteraciones da los mejores valores que se ajustan al modelo para cada hiperparámetro.

Support Vector Machine

Consideraremos dos hiperparámetros. El hiperparámetro C es un factor de regularización, y le dice al clasificador cuánto queremos evitar clasificar mal los ejemplos de entrenamiento. Un valor grande de C intentará clasificar correctamente más ejemplos del conjunto de entrenamiento, pero si C es demasiado grande puede sobreajustar el modelo y no generalizar al clasificar nuevos datos. Si C es demasiado pequeño, el modelo nos llevará a un subajuste. Si al aumentar más C no ayuda, probablemente no hay más puntos de entrenamiento dentro del margen o mal clasificados, o al menos no se puede encontrar una solución mejor. Si las puntuaciones son iguales, puede tener sentido utilizar C valores más pequeños, ya que los valores muy altos de C normalmente aumentan el tiempo de adaptación.

Por otro lado, los valores C más bajos generalmente conducen a más vectores de soporte, lo que puede aumentar el tiempo de predicción. Por lo tanto, el valor de C implica un equilibrio entre el tiempo de adaptación y el tiempo de predicción.

El otro hiperparámetro es γ, el cual define la influencia de un ejemplo de entrenamiento. Si γ es muy grande, el radio de influencia de los vectores de soporte solo influye al propio vector de soporte. En cambio con un γ muy pequeño, la región de influencia de un vector de soporte incluirá todo el conjunto de entrenamiento.

Random Forest

Se puede definir el número de árboles y el tamaño de estos, pero se debe considerar que demasiados árboles con gran profundidad pueden llevar a un sobreajuste, y viceversa pocos árboles con poca profundidad puede no entrenar suficientemente el modelo, por lo tanto es importante encontrar un ajuste entre ambos, lo que se aplicó en esta búsqueda fueron 3 opciones, grandes cantidades de árboles con poca profundidad, pocos árboles con gran profundidad y un equilibrio entre ambos, cantidad moderada de árboles y una profundidad intermedia. Aquí también se evalúa qué medida de pureza del nodo es mejor para el modelo.

K-Nearest Neighbor

En el método de vecinos cercanos son pocos los hiperparámetros que requieren, k es el número de vecinos con el que se clasificará, y p la distancia que ocupará para elegir a los vecinos cercanos, se tienen dos opciones distancia Euclidiana y distancia Manhattan.

XGBoost

Al igual que Random Forest, al ser un método basado en árboles de decisión el número y profundidad de los árboles puede llevar a un sobreajuste o subajuste.

La mayoría de los parámetros en XGBoost tienen que ver con la compensación

de la varianza y el sesgo. El mejor modelo debería negociar cuidadosamente la complejidad del modelo con su poder predictivo.

Para controlar el sobreajuste, se puede realizar de dos formas: la primera forma es controlar directamente la complejidad del modelo (profundidad y el parámetro gamma). La segunda forma es agregar aleatoriedad para que el entrenamiento sea resistente al ruido ("subsample" que controla la fracción de observaciones utilizadas y "colsample_bytree" que controla la fracción de características utilizadas).

5.10. Votación de clasificadores

Cada modelo tiene diferentes criterios de clasificación, principalmente SVM y KNN que se basan en interpretación geométrica, en cambio RF y XGB se basan en árboles de decisión. Como se vio anteriormente, cada modelo se entrenó independiente de otros modelos.

Al calcular las métricas de puntuación se sabe cuál método clasifica mejor, pero a veces hay métodos que aunque su puntuación es más baja, logran ver una clase mejor que otros métodos. Así que se aplicó un método en conjunto, con la función:

```
model = ↩
    sklearn.ensemble.VotingClassifier(estimators = ↩
    [('svm', svm_mod), ('knn',knn_mod), ('rf', ↩
    rf_mod), ('xgb', xgb_mod)], voting='soft')
```

Código 5.8: Votación de clasificadores.

la cual por cada muestra clasificada de los cuatro métodos, se vota "*suavemente*" por una clase. Obteniendo una sola salida para cada pozo, desde el punto de vista visual ayuda mucho a tener un modelo más homogéneo de las predicciones, quitando el ruido residual.

El proceso para este proyecto se muestra en la Figura 5.10, donde los 4 modelos clasifican, cada uno con sus procesos internos y al final se comparan para llegar a un resultado final.

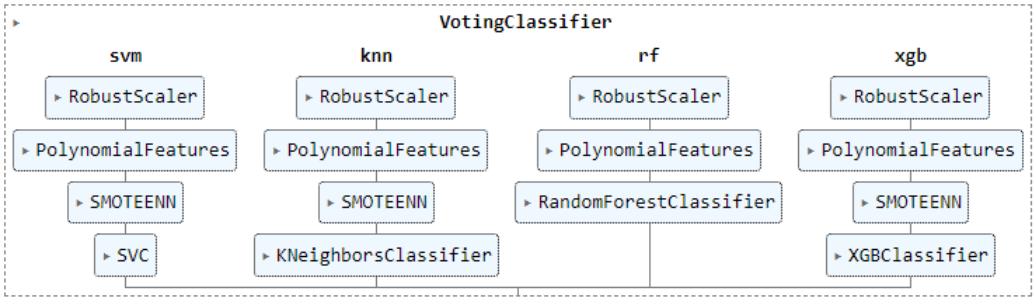


Figura 5.10: Proceso de la votación.

5.11. Post - Procesamiento

Para mejorar el resultado de la clasificación, se realiza el post-procesamiento de las clases, con el objetivo de atenuar los residuales generados en el proceso de clasificación, utilizando filtros estadísticos en una ventana de 8 [m], Figura 5.11.

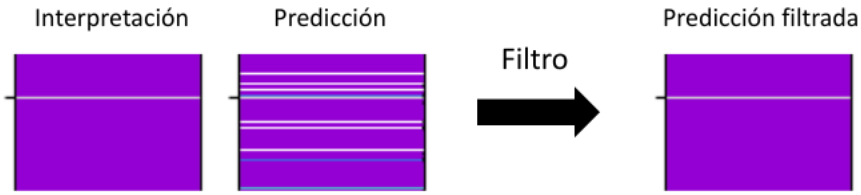


Figura 5.11: Ejemplo del post-procesamiento.

5.12. Evaluación del modelo de clasificación

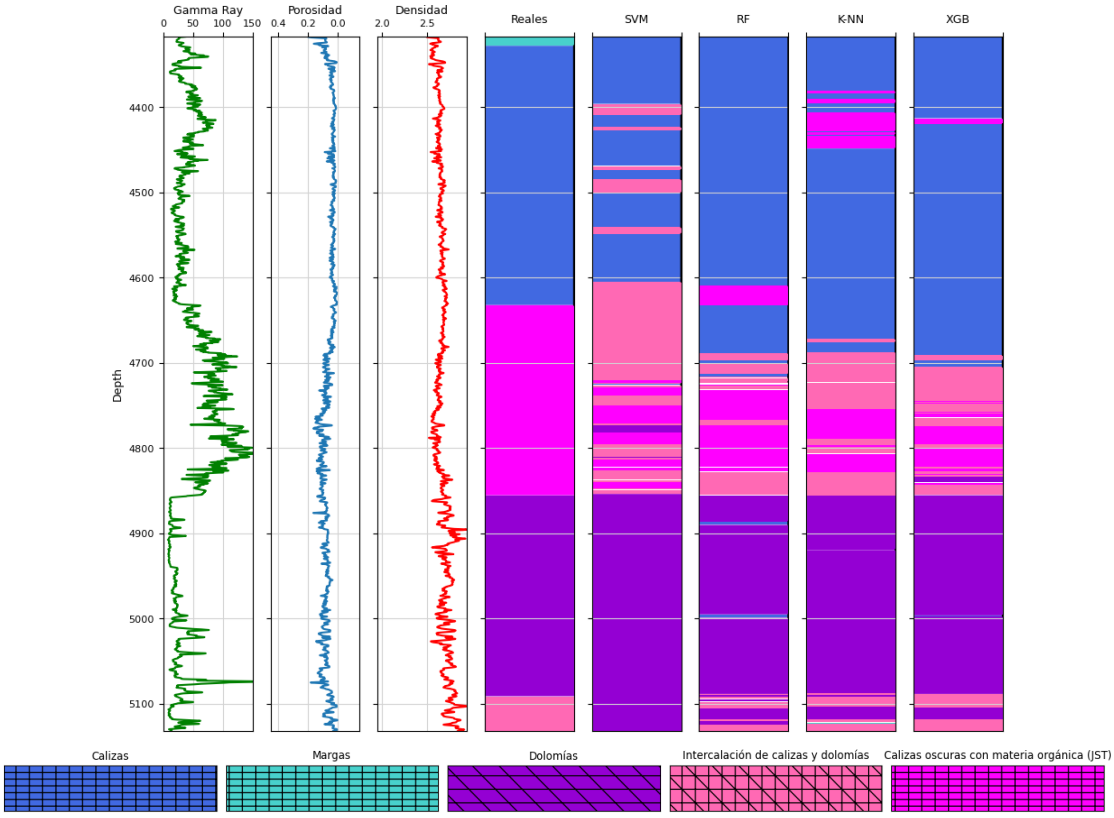
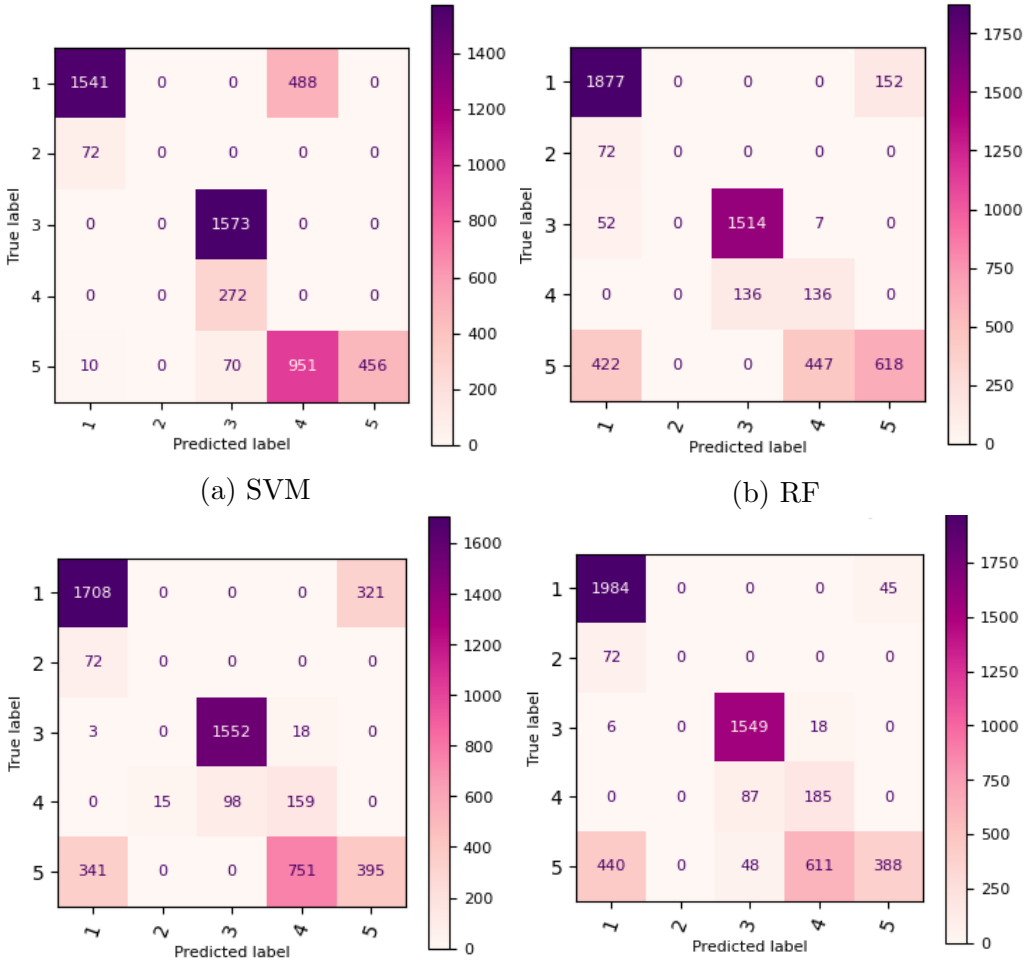


Figura 5.12: Pozo 0: Carril 1 Rayos Gamma, 2 Porosidad, 3 Densidad, 4 Facies interpretadas, 5 Clasificación con SVM, 6 Clasificación con RF, 7 Clasificación con KNN y 8 Clasificación con XGB.

Para la evaluación del modelo de clasificación, se utilizó el pozo 0. En la Figura 5.12 tenemos los 3 primeros carriles con los registros de Rayos Gamma, Porosidad y Densidad, en el cuarto carril se tiene las facies interpretadas, y en los siguientes carriles las facies clasificadas de cada uno de los modelos entrenados.

Se puede ver que no clasifica las margas, aunque se les aplicó un tratamiento para clases desbalanceadas, de igual forma le cuesta un poco la clase caliza oscura con materia orgánica, en cambio todos muestran muy buena respuesta al clasificar las dolomías.



(a) SVM (b) RF

(c) KNN (d) XGB

Figura 5.13: Matriz de confusión de los 4 modelos para el pozo 0.

Ahora en la Figura 5.13 se observan las matrices de confusión de los 4 modelos, analicemos el comportamiento por clase, para la caliza tuvo una clasificación excelente, ningún modelo la clasificó ni como margas ni como dolomías, las margas no las clasificó correctamente ningún modelo, en su lugar las clasificaron como calizas, las dolomías de igual forma el resultado es muy bueno, teniendo en cuenta que también era una clase minoritaria, la intercalación de calizas y dolomías, en algunos tiene buena respuesta, pero en SVM, la clasifica totalmente como dolomía, para la clase

que se tiene un poco más de problema es para la caliza oscura ya que tienden a clasificarla como calizas o intercalación de calizas y dolomías.

Recordemos lo que las métricas de puntuación nos dicen, la precision indica el porcentaje de predicciones que se clasificaron como una clase que fueron verdaderas, y recall el porcentaje de predicciones que se clasificaron correctamente por toda esa clase. En la Tabla 5.2 se tienen las métricas de evaluación promediadas, se observa que para la mayoría hay una relación equilibrada entre precision y recall.

Macro promedio				
	SVM	RF	KNN	XGB
Precision	59 %	58 %	53 %	60 %
Recall	45 %	60 %	57 %	61 %
F1	48 %	56 %	51 %	54 %

Tabla 5.2: Macro promedio de las métricas para el pozo 0.

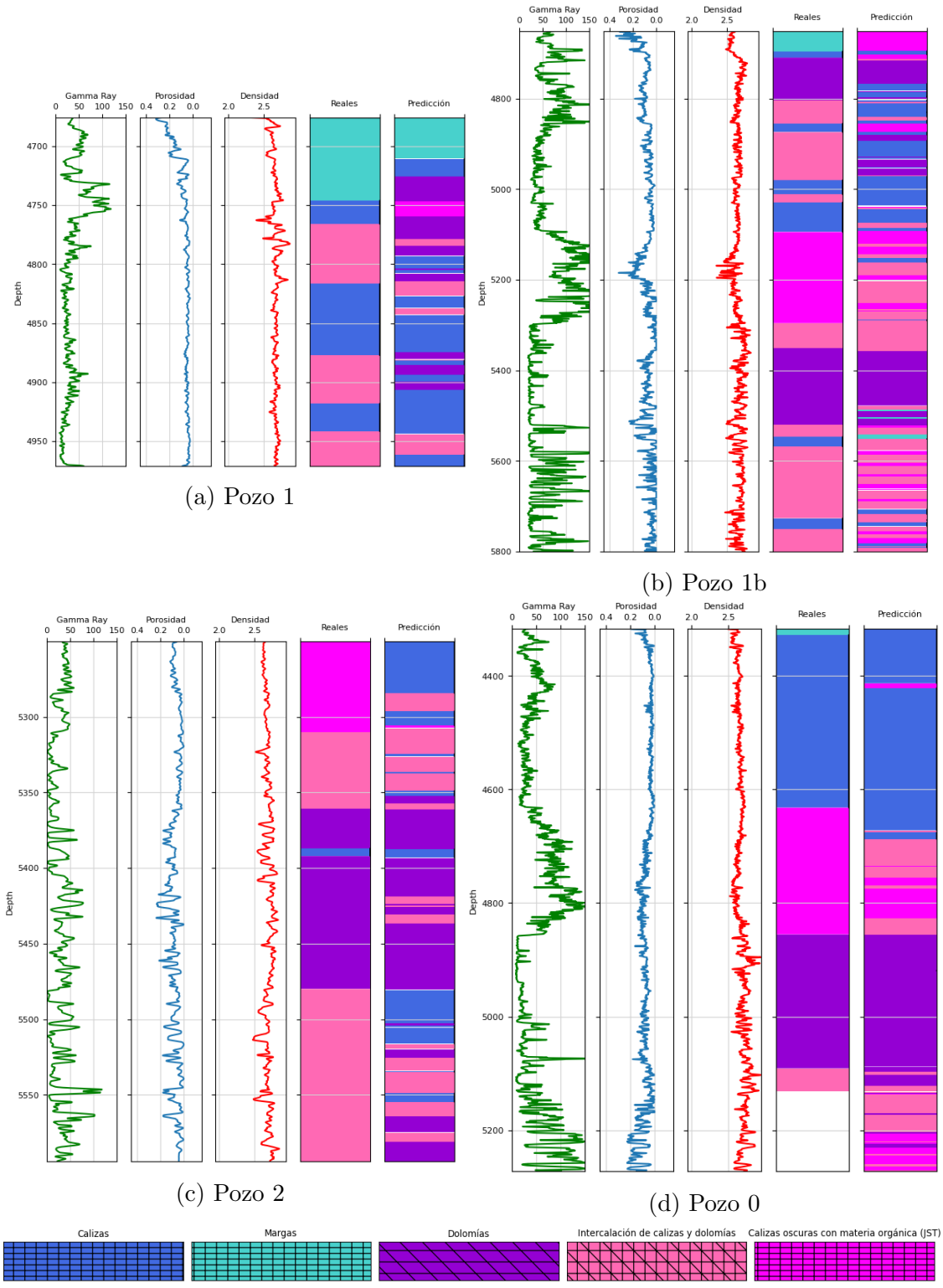
Al ver las puntuaciones de esté pozo se podrían considerar bajas, debido a que el macro promedio indica el promedio tratando a las clases con la misma importancia, por lo tanto al no predecir ninguna marga la puntuación disminuye, lo cual aunque clasifiquemos correctamente las otras clases, la puntuación más alta que se puede esperar es del 80 %, y el problema con las calizas oscuras es que tiende a confundirlas con calizas o intercalación de calizas y dolomías. Es importante comparar las tres formas de representar los resultados, que son gráficamente, en la matriz de confusión y en la puntuación, para definir si el clasificador es bueno.

5.13. Aplicación del modelo de clasificación

Se aplicó el clasificador a los pozos que no se utilizaron para entrenar, Figura 5.14. Desde el punto de vista de la clasificación de facies individual las dolomías (clase morada) son las que tienden a clasificar mejor, la intercalación de calizas y dolomías en algunas secciones tiende a clasificarla como la clase caliza o la clase dolomía, lo cual no está del todo mal, ya que lo identifica como uno o como otro, la clase que más les cuesta clasificar es la clase caliza oscura con materia orgánica.

Se exportaron a Petrel los datos del pozo 1b, Figura 5.15, en el primer carril está el registro de lentitud de onda de compresión, en el segundo carril el registro de Rayos Gamma, en el tercer carril el registro de densidad, en el cuarto carril el registro de porosidad efectiva, en el quinto carril se tiene marcado donde se ha encontrado aceite, en el sexto carril se tiene una interpretación de facies basadas en ambientes sedimentarios, donde el color más claro son facies de baja energía (ambiente de laguna) y el color más oscuro son facies de alta energía (bancos oolíticos), en el séptimo carril se cargaron las facies predichas por la votación de los 4 modelos, solo para la parte del Jurásico Superior Kimmeridgiano.

Las facies litológicas son diferentes a las facies de ambientes sedimentarios, pero tienen relación y eso se puede observar en la Figura 5.15.



(a) Pozo 1

(b) Pozo 1b

(c) Pozo 2

(d) Pozo 0

Figura 5.14: Resultados para cada pozo: Carril 4 Facies interpretadas, Carril 5 Clasificación mediante votación de los 4 modelos.

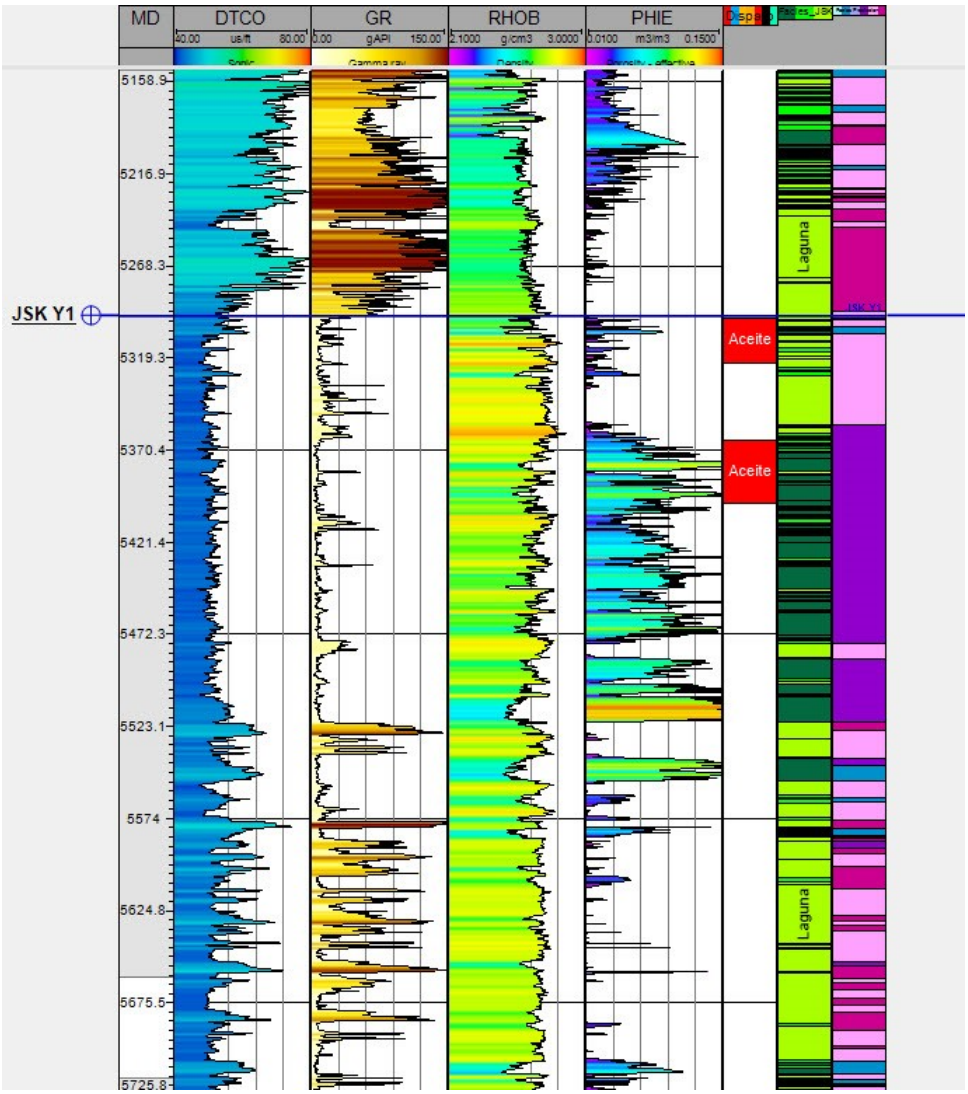


Figura 5.15: Pozo 1b en Petrel.

Capítulo 6

Conclusión

La clasificación de facies permite evaluar las características de la roca, para encontrar la mejor calidad de roca de los yacimientos de petróleo y gas. De aquí surge la idea principal de esta tesis, crear un clasificador que sirva de apoyo a los geocientíficos al momento de realizar las interpretaciones. Los distintos métodos logran encontrar relaciones en el comportamiento entre las 6 curvas de los registros para asignarlos a una clase, en un tiempo muy corto, menor a 1 minuto para cada pozo, aunque los clasificadores no son exactos, su respuesta en algunas clases es muy precisa, como lo es en la dolomía, al igual que en algunos límites entre clases, pero aún pueden mejorar más los modelos, por ejemplo la clase caliza oscura con materia orgánica se encuentra en el Jurásico Superior Tithoniano y las margas se encuentran la mayor parte en el Cretácico, así surge la idea de agregar la edad como una característica, también se podrían agregar otros registros como los de resistividad.

La creación de modelos de ML, es algo sencillo de desarrollar desde el punto de vista de la programación, pero el desarrollo de un buen modelo es algo más complicado, para alguien principiante en ML. Inicialmente este trabajo se basó totalmente en el código de Brendon Hall (Hall, 2016) que utiliza el método SVM, pero después se creó un código totalmente diferente con múltiples métodos, algo importante como recomendación, es que si a una persona con el mismo problema o idea ya trabajó con un método y se obtuvieron buenos resultados, no quiere decir que no se pueda probar otros métodos y obtener mejores resultados. La idea es buscar qué otros métodos existen, indagar un poco en la teoría, cuáles son sus ventajas y desventajas, todo esto dependiendo de las necesidades.

La mayoría de los temas que se abordan en este trabajo, son problemas enfrentados en el desarrollo de este modelo. Se inició con una exactitud del 13 % al 17 %, pero finalmente se obtuvo un incremento en la puntuación de los modelos. Logrando desarrollar un clasificador que tiene su media de predicción del 60 % con datos reales. Existen múltiples factores que ayudan a obtener un buen modelo de clasificación, una de ellas es la calidad de los datos, si se ingresan buenos datos se obtienen buenos resultados, si se ingresan datos con ruido el clasificador aprenderá de ese ruido. La cantidad también es importante, entre más datos mejor será el ajuste del modelo y se podrá definir mejor la tendencia de cada clase. Evitar el sobreajuste, el subajus-

te, la fuga de datos e integrar a los pasos la ingeniería de características, son piezas clave que se deben tener presente.

Como continuación de este trabajo de tesis se planea trasladar el notebook de Python a una interfaz web más amigable, para las personas que no están familiarizados con el lenguaje de Python. Así como extender esta técnica de clasificación, al modo semi-supervisado con otros algoritmos de ML, con el objetivo de eliminar la clase "intercalación de calizas y dolomías", para poder clasificar solamente como una clase u otra, igualmente utilizar otros pozos que no están interpretados. Lo que ayudaría en la automatización del proceso de toma de decisiones para realizar agrupaciones automáticamente sin intervención humana, ahorrando tiempo y recursos.

Entre las ventajas que observamos de las técnicas de ML están: La capacidad para manejar datos complejos con características altamente no lineales y que pueden no ser fácilmente manejables con enfoques tradicionales. Mejora continua al utilizar más datos, ya que los modelos de ML tienden a mejorar su rendimiento a medida que se alimentan con más datos, lo que les permite aprender patrones más complejos y mejorar su precisión con el tiempo. La identificación de patrones sutiles en los datos, que pueden no ser evidentes para el ojo humano o que pueden ser difíciles de expresar de manera explícita mediante reglas programáticas. La escalabilidad, lo que significa que pueden manejar conjuntos de datos de gran tamaño y pueden ser implementados en sistemas distribuidos para un procesamiento eficiente.

Referencias

Batista, G., Prati, R., & Monard, M.-C. (2004). A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *SIGKDD Explorations*, *6*, 20-29. doi: 10.1145/1007730.1007735

Bestagini, P., Lipari, V., & Tubaro, S. (2017). A Machine Learning Approach to Facies Classification Using Well Logs. En *Seg technical program expanded abstracts* (p. 2137-2142). Society of Exploration Geophysicists.

Breiman, L. (2001). Random Forests. *Machine learning*, *45*, 5-32.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification And Regression Trees*. Chapman and Hall/CRC.

Brownlee, J. (2016). *Master Machine Learning Algorithms Discover How They Work and Implement Them From Scratch*. Machine Learning Mastery.

Brownlee, J. (2018). *Data Preparation for Machine Learning: Data Cleaning, Feature Selection and Data Transforms in Python*. Machine Learning Mastery.

Brownlee, J. (2020). *Imbalanced Classification with Python: Better Metrics, Balance Skewed Classes, Cost-Sensitive Learning*. Machine Learning Mastery.

Brownlee, J. (2021). *Ensemble learning algorithms with Python: Make better predictions with bagging, boosting, and stacking*. Machine Learning Mastery.

Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, W. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal Artificial Intelligence Research*, *16*, 321-357. doi: 10.1613/jair.953

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. En *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (p. 785-794). ACM. doi: 10.1145/2939672.2939785

CNH. (2014). Cuencas del Sureste Aguas Someras Síntesis Geológica Petrolera.

CNH. (2015). Atlas Geológico Cuencas del Sureste - Cinturón Plegado de la Sierra de Chiapas.

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine learning*, *20*, 273-297.

Cover, T. M., & Hart, P. E. (1967). Nearest Neighbor Pattern Classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 21-27.

Cross, T. A., & Homewood, P. W. (1997). Amanz Gressly's role in founding modern stratigraphy. *Geological Society of America Bulletin*, *109*(12), 1617-1630.

Cunningham, P., & Delany, S. (2007). *k-Nearest Neighbour Classifiers* (Vol. 54; Inf. Téc.). doi: 10.1145/3459665

Dagnino, J. (2014). Correlación. *Rev Chil Anest*, *43*, 150-153.

Faz Pérez, P. (2014). *Análisis de Sello Lateral en Oportunidades Exploratorias de Hidrocarburos con Objetivos Cretácico y Jurásico Superior Kimmeridgiano, en la Cuenca Petrolífera del Sureste en su Porción Marina.* (Tesis de Master, Universidad Nacional Autónoma de México). Descargado de `https://hdl.handle.net/20.500.14330/TES01000718966`

Google. (s.f.). *Data Preparation and Feature Engineering.*

Guerra, A., A. C. da P., C., & Nery, G. (2017). Facies classification in well logs of the Namorado oilfield using Support Vector Machine algorithm. En *15th international congress of the brazilian geophysical society expogef* (p. 1853-1858). Brazilian Geophysical Society.

Hall, B. (2016). Facies classification using machine learning. *The Leading Edge*, *35*(10), 906-909.

Hart, P. (1968). The Condensed Nearest Neighbor Rule. *IEEE transactions on information theory*, *14*(3), 515-516.

Hurwitz, J., & Kirsch, D. (2018). Machine Learning for dummies. *IBM Limited Edition*.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning*. Springer.

Li, Y., & Anderson-Sprecher, R. (2006). Facies identification from well logs: A comparison of discriminant analysis and naïve Bayes classifier. *Journal of Petroleum Science and Engineering*, *53*, 149-157. doi: 10.1016/j.petrol.2006.06.001

McDonald, A. (2021). *Python and Petrophysics Notebook Series.* https://github.com/andymcdgeo/Petrophysics-Python-Series. (v1.0.0)

O'Neil, C., & Schutt, R. (2013). *Doing Data Science: Straight talk from the frontline*. O'Reilly Media, Inc.

Padilla y Sánchez, R. (2007). Evolución geológica del sureste mexicano desde el Mesozoico al presente en el contexto regional del Golfo de México. *BOLETÍN DE LA SOCIEDAD GEOLÓGICA MEXICANA*, *59*, 19-42. doi: 10.18268/BSGM2007v59n1a3

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Petrobal. (2023). *Interpretación y Análisis de Distribución de Bancos Oolíticos.* (Informes internos de Petrobal)

Steinbach, M., & Tan, P.-N. (2009). kNN: k-Nearest Neighbors. En (p. 151-162).

Taiz. (s.f.). Classification: Basic Concepts, Decision Trees and Model Evaluation. , 145-205. Descargado de `https://www-users.cse.umn.edu/~kumar/dmbook/ch4.pdf`

Tomek, I. (1976). Two Modifications of CNN. *IEEE Transactions on Systems, Man, and Cybernetics*, 769-772.

Ángeles Aquino, F. J. (2006). Monografía petrolera de la Zona Marina. *Asociación Mexicana de Geológos Petroleros*, 77.