

Universidad Central Marta Abreu de las Villa (UCLV)

Facultad de Matemática-Física-Computación

Autores: Carlos Elías González Valdés   
 René Espinosa Arteaga

**PRÁCTICA LABORAL**

**E INVESTIGATIVA**

**RESUMEN**

El Machine Learning (ML) es una rama de la inteligencia artificial que permite a las máquinas aprender a partir de datos. El desarrollo de este trabajo ha permitido un acercamiento al ámbito del aprendizaje automático, facilitando la familiarización con conceptos clave y la implementación de modelos prácticos de machine learning para abordar tareas de clasificación, regresión y agrupamiento. Se obtuvieron resultados satisfactorios en cada una de estas tareas. Se siguió un flujo de trabajo estructurado y se logró una organización eficiente del proceso, desde la recolección y preprocesamiento de datos hasta la evaluación de modelos

**INDICE**

[**1.** **INTRODUCCION** 3](#_Toc178809443)

[**2.** **DESARROLLO** 4](#_Toc178809444)

[**2.1** **Aspectos Teóricos** 4](#_Toc178809445)

[**2.2** **Descripción del flujo de trabajo.** 9](#_Toc178809446)

[**2.2.1** **Clasificación sobre el *Wine Dataset*** 9](#_Toc178809447)

[**2.2.2** **Regresión sobre el California Housing Dataset** 12](#_Toc178809448)

[**2.2.3 Clusterización 14**](#_Toc178809449)

[**3.** **CONCLUSIONES** 17](#_Toc178809450)

[**4.** **BIBLIOGRAFÍA** 17](#_Toc178809451)

# **INTRODUCCION**

En la última década, el campo del Machine Learning (ML) ha emergido como una de las áreas más prometedoras y transformadoras dentro de la inteligencia artificial. Su capacidad para extraer patrones y conocimientos a partir de grandes volúmenes de datos ha revolucionado sectores tan diversos como la salud, las finanzas, el marketing y la tecnología. El trabajo con Machine Learning es relevante desde una perspectiva profesional, por lo que la adquisición de conocimientos y habilidades en esta área motivan la unión al Grupo de Trabajo Científico.

Este trabajo investigativo tiene como objetivo general el desarrollo de modelos de Machine Learning para abordar tareas específicas de clasificación, regresión y clusterización.

Para alcanzar este objetivo, se investiga acerca de los conceptos fundamentales que rigen esta disciplina, incluyendo los distintos tipos de aprendizaje, las diversas tareas que se pueden realizar, así como las métricas y algoritmos más utilizados en la práctica. Además, se estudia el flujo de trabajo típico en proyectos de Machine Learning, desde la recolección y preprocesamiento de datos hasta la implementación y evaluación de modelos.

A través de esta investigación, se buscará no solo desarrollar modelos aplicados a casos concretos, sino también evaluar críticamente los resultados obtenidos, permitiendo así una comprensión más profunda de las capacidades y limitaciones de cada enfoque. Con ello, se espera contribuir al entendimiento y aplicación efectiva de técnicas de Machine Learning en contextos reales, y de esta forma enfrentar los desafíos que presenta esta disciplina en constante evolución.

# **DESARROLLO**

* 1. **Aspectos Teóricos**

**¿Qué es Machine Learning?**

El Machine Learning (ML) es una rama fascinante de la inteligencia artificial que permite a las máquinas aprender a partir de datos y mejorar su rendimiento en tareas específicas sin que necesiten ser programadas explícitamente para cada acción. En lugar de depender de reglas predefinidas, los modelos de ML son capaces de detectar patrones en los datos y utilizar esos patrones para hacer predicciones, clasificaciones, o agrupar información de manera significativa. Esto abre un mundo de aplicaciones, desde la clasificación de imágenes hasta la predicción de precios o el análisis de riesgos.

Lo crucial en el proceso de aprendizaje automático es la capacidad de los modelos de generalizar, es decir, de identificar patrones en los datos de entrenamiento y aplicarlos de manera efectiva a datos nuevos.

A continuación, se explorarán los tipos de aprendizaje, las principales tareas, estrategias de pre-procesamiento y entrenamiento, y una introducción a los principales algoritmos y métricas para evaluar el rendimiento de los modelos.

* + 1. **Tipos de Aprendizaje en Machine Learning**

El aprendizaje en ML se clasifica en varios paradigmas, dependiendo de la naturaleza de los datos y del problema a resolver:

* **Aprendizaje supervisado:** Este es quizás el enfoque más común. El modelo aprende a partir de datos etiquetados, donde la salida correcta ya está especificada. Las tareas principales incluyen: Clasificación y Regresión.
* **Aprendizaje no supervisado:** Aquí el modelo trabaja con datos no etiquetados y su tarea es descubrir patrones ocultos. Las aplicaciones incluyen: Clusterización y Reducción de dimensionalidad.
* **Aprendizaje semi-supervisado:** Combina una pequeña cantidad de datos etiquetados con una gran cantidad de datos no etiquetados, muy útil en escenarios donde etiquetar todos los datos sería demasiado costoso o difícil.
* **Aprendizaje por refuerzo:** El algoritmo aprende interactuando con un entorno, tomando decisiones y recibiendo recompensas o penalizaciones. Este enfoque es especialmente útil en escenarios como el entrenamiento de agentes autónomos, por ejemplo, un robot que aprende a moverse o un agente que juega videojuegos mejorando sus estrategias con cada ronda.
  + 1. **Tareas en Machine Learning**

Cada uno de estos tipos de aprendizaje puede aplicarse a diversas tareas clave:

* **Clasificación:** En la clasificación, el objetivo es asignar una etiqueta a los datos. Esto es útil en áreas como el reconocimiento de imágenes, detección de correos electrónicos de spam o diagnóstico médico.
* **Regresión:** Cuando el objetivo es predecir un valor continuo (como el precio de una casa o el riesgo de crédito)
* **Clusterización:** Utilizada para encontrar grupos de datos similares en aplicaciones como segmentación de mercados o análisis de clientes. Este es un enfoque no supervisado.
  + 1. **Técnicas para el Análisis Exploratorio de Datos (EDA)**

El análisis exploratorio de datos (EDA) es una etapa clave en proyectos de machine learning que permite comprender las características y patrones de los datos. Algunas de las técnicas más clásicas incluyen:

* **Estadísticas descriptivas**: Resumen de datos mediante medidas como la media, mediana, moda, desviación estándar y percentiles, que permiten entender la distribución y dispersión de los valores.
* **Visualización de datos**: Gráficos como histogramas, diagramas de caja (boxplots) y gráficos de dispersión ayudan a identificar tendencias, outliers y relaciones entre variables.
* **Matriz de correlación**: Utilizada para medir la relación entre variables numéricas y visualizar posibles correlaciones, permitiendo identificar características redundantes.
  + 1. **Estrategias de Pre-procesamiento**

El pre-procesamiento es fundamental para garantizar que los datos estén en la mejor forma posible antes de entrenar los modelos. Esto implica limpiar, transformar, y organizar los datos para maximizar el rendimiento del modelo:

* **Feature Engineering:** Crear nuevas características o modificar las existentes. Por ejemplo, normalizar variables o crear nuevas combinaciones de variables para mejorar la capacidad del modelo de extraer información útil.
* **Manejo de valores perdidos:** Los datos incompletos pueden sesgar los resultados de un modelo. Las técnicas comunes incluyen eliminar filas incompletas o imputar valores faltantes usando la media, mediana o métodos más avanzados.
* **Codificación de variables categóricas:** Las variables categóricas deben transformarse en una representación numérica. Técnicas como One-Hot Encoding convierten las categorías en columnas binarias que los modelos pueden interpretar.
* **Reducción de dimensionalidad:** Técnicas como PCA o t-SNE ayudan a simplificar los datos sin perder información valiosa, lo que puede mejorar la eficiencia del modelo y reducir el ruido.
  + 1. **Estrategias de Entrenamiento**

Una parte crucial de ML es cómo se dividen los datos y se ajusta el modelo. El enfoque adecuado puede marcar la diferencia entre un modelo que generaliza bien y uno que simplemente se ajusta a los datos de entrenamiento.

* **Conjunto de entrenamiento:** Este es el conjunto de datos utilizado para que el modelo aprenda. Aquí el objetivo es que el modelo reconozca los patrones clave.
* **Conjunto de validación:** Es fundamental para ajustar los hiperparámetros y prevenir el overfitting. Al validar el rendimiento durante el entrenamiento, podemos asegurarnos de que el modelo no se ajuste demasiado a los datos de entrenamiento, permitiendo que se generalice mejor.
* **Conjunto de prueba:** Finalmente, el conjunto de prueba es un subconjunto que no se ha usado ni en el entrenamiento ni en la validación, y que nos da una medida realista de cuán bien funcionará el modelo con datos completamente nuevos.
* **Validación cruzada:** En lugar de utilizar una única división de los datos, la validación cruzada divide el conjunto de datos en múltiples partes. Así, el modelo se entrena y evalúa en diferentes subconjuntos, lo que mejora la capacidad de estimar su rendimiento real.

**2.1.6 Principales Algoritmos de Machine Learning**

* **Regresión Logística**: A pesar de su nombre, la **Regresión Logística** no se utiliza para problemas de regresión, sino para tareas de **clasificación**, especialmente para aquellas donde tienes que predecir entre dos opciones (por ejemplo, si un correo es spam o no). Este algoritmo trata de calcular la **probabilidad** de que una instancia (una fila de datos) pertenezca a una clase en particular. Utiliza una función matemática llamada **sigmoide** que transforma el resultado en un valor entre 0 y 1, lo que representa la probabilidad de pertenecer a la clase objetivo.
* **Máquinas de Soporte Vectorial (SVM)**: Las **SVM** son muy poderosas cuando necesitas clasificar datos que no son linealmente separables, es decir, cuando no puedes dividir las clases con una simple línea recta. Su objetivo es encontrar el **hiperplano** (una línea o plano en espacios más complejos) que mejor separe las clases. Si los datos son más complicados y no pueden separarse de manera simple, SVM utiliza algo llamado **kernel** para transformar los datos en un espacio de mayor dimensionalidad, donde pueda encontrar esa separación más fácilmente. Las SVM son eficaces para problemas de **clasificación binaria y multiclase**, y se utilizan en tareas como la clasificación de imágenes y reconocimiento de patrones.
* **Árboles de Decisión**: Este algoritmo es fácil de entender y se basa en una estructura similar a un árbol. Cada **nodo** en el árbol representa una pregunta o condición basada en los datos. A medida que avanzas por las ramas, vas tomando decisiones hasta llegar a una **hoja**, que es la predicción final. Los árboles de decisión se usan tanto para **clasificación** (por ejemplo, si un cliente comprará o no un producto) como para **regresión** (por ejemplo, predecir el precio de una casa).
* **Random Forest**: Es una mejora de los árboles de decisión. En lugar de depender de un solo árbol, el algoritmo crea un **bosque de árboles** a partir de diferentes subconjuntos de los datos y de las características. Cada árbol del bosque da una predicción y el Random Forest toma la clase que recibe más votos (en clasificación) o promedia las predicciones (en regresión). Esta técnica reduce el problema de sobreajuste y generalmente proporciona mejores resultados que un solo árbol.
* **Regresión Lineal**: Es uno de los algoritmos más sencillos y se usa para predecir valores **continuos**. En la regresión lineal, la idea es encontrar una línea recta que mejor se ajuste a los datos, de forma que minimice la diferencia entre las predicciones y los valores reales. Este algoritmo es útil cuando crees que hay una **relación lineal** entre las características y la variable objetivo, por ejemplo, predecir el precio de una casa en función de su tamaño.
* **K-Means**: El algoritmo **K-Means** se utiliza para **agrupamiento**, es decir, cuando quieres dividir tus datos en diferentes grupos o **clusters** según su similitud. K-Means funciona asignando aleatoriamente "centroides" (puntos que representan el centro de cada grupo) y luego ajusta esos centroides para que los puntos estén lo más cerca posible de su centroide correspondiente. Es útil cuando quieres explorar tus datos y agruparlos en grupos similares, por ejemplo, agrupar clientes en segmentos basados en sus comportamientos de compra. Sin embargo, necesitas especificar de antemano cuántos grupos quieres formar.
* **DBSCAN**: A diferencia de K-Means, el algoritmo **DBSCAN** no requiere que especifiques el número de clusters de antemano. En su lugar, agrupa puntos que están **densamente ubicados** y considera los puntos más dispersos como **ruido** o puntos atípicos. Esto lo hace ideal para encontrar clusters de **formas arbitrarias** y para manejar datos con ruido. Es particularmente útil en situaciones donde los datos no tienen una estructura regular o cuando no estás seguro de cuántos clusters debes formar.

**2.1.7 Métricas de Evaluación**

Las métricas nos permiten evaluar el rendimiento de los modelos y elegir la mejor opción para nuestros datos. Estas varían según el tipo de problema:

**Problemas de Clasificación:**

* Accuracy: Mide la proporción de predicciones correctas sobre el total de casos. Útil, pero puede ser engañoso en datasets desequilibrados.
* Precision y Recall: La precisión mide cuántas de las predicciones positivas fueron correctas, mientras que el recall mide cuántos de los verdaderos positivos fueron correctamente identificados.
* F1-Score: Combina precisión y recall en una sola métrica, útil cuando hay un desbalance en las clases.
* AUC y ROC: La curva ROC traza la tasa de verdaderos positivos frente a la tasa de falsos positivos. El AUC mide el área bajo esta curva y proporciona una evaluación de la capacidad del modelo para distinguir entre clases.
* Matriz de Confusión: Representa las predicciones correctas e incorrectas de manera más detallada. Las filas representan las clases verdaderas y las columnas las predicciones, lo que facilita identificar errores específicos.

**Problemas de Regresión:**

* MAE (Mean Absolute Error): Promedio de las diferencias absolutas entre las predicciones y los valores reales.
* RMSE (Root Mean Squared Error): Penaliza más los errores grandes, siendo útil cuando los errores extremos son problemáticos.
* R² (Coeficiente de determinación) Indica qué proporción de la variabilidad en la variable dependiente está explicada por las variables independientes.

**Problemas de Clusterización**:

* *Silhouette Score*: Mide la coherencia interna de los grupos. Un valor cercano a 1 indica una buena separación entre clusters.
* *Inercia*: Suma de las distancias al cuadrado de los puntos al centroide más cercano. Un valor bajo sugiere clusters más compactos.
  1. **Descripción del flujo de trabajo.** 
     1. **Clasificación sobre el Wine Dataset**

Se comenzó cargando las bibliotecas necesarias para manipulación de datos, visualización y machine learning, tales como pandas, numpy, matplotlib, y scikit-learn. Posteriormente, se cargó el Wine Dataset y se realizó una visualización inicial de las primeras filas, proporcionando un panorama general de los datos.

* **Análisis Exploratorio de Datos (EDA)**

A través de la exploración inicial del conjunto de datos, se revelaron varios aspectos importantes que ofrecen una comprensión más profunda del mismo. El dataset consta de 178 observaciones con 13 características numéricas y una variable objetivo que clasifica los vinos en tres categorías. No se encontraron valores faltantes, lo que evita la necesidad de aplicar técnicas de imputación. Sin embargo, las características presentan una amplia gama de valores, lo que sugiere la importancia de aplicar técnicas de escalado antes de proceder con los modelos. También se observó que la distribución de las clases de vino es bastante equilibrada, un factor favorable para el rendimiento de los algoritmos de clasificación. No obstante, algunas características, como el contenido de alcohol y la prolina, muestran distribuciones asimétricas, lo que podría influir en los resultados finales y subraya la necesidad de un preprocesamiento adecuado.

División del Dataset : El dataset fue dividido en conjunto de entrenamiento (80%) y conjunto de prueba (20%) utilizando la función train\_test\_split(). Se utilizó una división estratificada para garantizar que ambas particiones contengan una proporción equitativa de cada clase. El conjunto de prueba se mantuvo aislado para su evaluación final.

Matriz de Correlación: El análisis de la matriz de correlación mostró relaciones significativas entre varias características y la variable objetivo. Particularmente, las variables flavanoids y total\_phenols tienen una fuerte correlación negativa con el target (-0.870 y -0.714, respectivamente), lo que sugiere que son determinantes en la clasificación de los vinos. Además, se observó una alta correlación entre flavanoids y total\_phenols (0.85), lo cual es relevante, ya que ambas variables están relacionadas con la composición fenólica del vino. Esto indicó la necesidad de gestionar correctamente la multicolinealidad en el preprocesamiento.

Análisis de Outliers: Mediante gráficos de boxplot, se identificaron outliers en características como proline, magnesium, ash y color\_intensity. Estos valores atípicos podrían afectar negativamente los modelos de clasificación, especialmente aquellos sensibles a los valores extremos, como la regresión logística.

* **Preprocesamiento**

Ingeniería de Rasgos : Con base en el análisis exploratorio, se crearon dos nuevas variables: flavanoid\_ratio y nonflavanoid\_ratio, que representan la proporción de flavonoides y no flavonoides respecto a los fenoles totales. Esto ayudó a reducir la multicolinealidad entre las características altamente correlacionadas. Además las características originales total\_phenols, flavanoids y nonflavanoid\_phenols fueron eliminadas tras la creación de estas nuevas variables, con el objetivo de evitar redundancias en los modelos.

Manejo de Outliers : Para mitigar el impacto de los valores atípicos sin eliminarlos, se aplicó winsorización. Esta técnica limita los valores extremos a un rango específico (por ejemplo, el 95 percentil), lo que ayuda a reducir la influencia de los outliers en los modelos.

Escalado de Datos: Dado que varias características tenían rangos muy distintos (por ejemplo, proline y alcohol), se aplicó escalado mediante StandardScaler(). Este paso fue crucial, especialmente para modelos como SVM, que son sensibles a las escalas de las características.

Reducción de Dimensionalidad con PCA : Se aplicó Análisis de Componentes Principales (PCA) con el objetivo de reducir la dimensionalidad del dataset y facilitar la visualización.

Se concluyó en que las primeras 4-5 componentes principales capturan aproximadamente el 70% de la varianza total y la **proyección en 2D** mostró una separación clara entre las clases de vinos, indicando que PCA puede captar cierta estructura inherente de los datos. Sin embargo, la aplicación de esta técnica no mejoró significativamente el rendimiento de los modelos de clasificación, lo que sugiere que la información útil para la clasificación no estaba necesariamente concentrada en las primeras componentes principales.

* **Selección y Entrenamiento de Modelos**

Se entrenaron tres modelos de clasificación: **Regresión Logística**, **Random Forest** y **Support Vector Machine (SVM)**. Los resultados iniciales en el conjunto de entrenamiento mostraron una precisión del 100% para todos los modelos, lo cual sugiere un posible sobreajuste.

Validación cruzada: Para verificar estos resultados, se implementó **validación cruzada estratificada (Stratified K-Fold)**, lo que permitió una evaluación más robusta del rendimiento. Los tres modelos alcanzaron precisiones superiores al 97%, siendo SVM el más preciso con una media del 98.6% y baja desviación estándar.

Optimización de Hiperparámetros: Se utilizó **GridSearchCV** para optimizar los hiperparámetros de los modelos. Esto permitió ajustar los modelos de manera más precisa, equilibrando la complejidad y la regularización. Los modelos optimizados mostraron mejoras sustanciales en consistencia y generalización.

* **Evaluación del Modelo en el Conjunto de Prueba**

Los modelos optimizados fueron evaluados en el conjunto de prueba, obteniendo los siguientes resultados:

**Regresión Logística**: Precisión del 97%, con buen balance entre recall y F1-score.

**Random Forest**: Clasificó correctamente todas las muestras del conjunto de prueba, aunque esto podría ser una señal de sobreajuste.

**SVM:** Desempeño consistente con una precisión cercana al 98%.

Se analizaron las curvas ROC y las áreas bajo la curva (AUC) para evaluar el rendimiento de los modelos. **Random Forest** mostró el mejor desempeño, con AUC cercanas a 1.0 para todas las clases.**Regresión Logística** y **SVM** también lograron AUC elevadas, lo que indica una excelente capacidad discriminativa de los modelos.

* **Análisis de resultados**

**Rendimiento Predictivo:** El modelo de **Random Forest** se destacó por su capacidad para capturar relaciones no lineales entre las características, lo que resultó en un excelente rendimiento predictivo, clasificando correctamente todas las muestras del conjunto de prueba. Aunque mostró el mejor desempeño global, este resultado sugiere un posible sobreajuste, ya que también alcanzó una precisión del 100% en el conjunto de entrenamiento.

La **Regresión Logística**, por otro lado, fue un modelo eficiente en términos de tiempo de entrenamiento y mostró un buen rendimiento con una precisión del 97%. Sin embargo, su naturaleza lineal limitó su capacidad para manejar relaciones más complejas, lo que afectó ligeramente su recall en algunas clases.

El **SVM**, aunque más costoso computacionalmente debido al uso de kernels no lineales, demostró ser el más consistente, con una precisión cercana al 98% tanto en entrenamiento como en el conjunto de prueba, mostrando un excelente balance entre precisión y recall.

**Selección de Métricas:** Se utilizaron **precisión**, **recall**, **F1-score**, y **AUC-ROC** para evaluar los modelos, ya que ofrecen una visión completa del rendimiento en un problema multiclase. La precisión fue clave para evaluar la exactitud de las predicciones, mientras que el recall permitió medir cuántos verdaderos positivos fueron detectados correctamente. El **F1-score** fue esencial para balancear precisión y recall y la curva **ROC** y el **AUC** ayudaron a evaluar la capacidad discriminativa de los modelos

**Balance Eficiencia-Eficacia**: Si bien **Random Forest** fue el modelo con mejor capacidad predictiva, su tiempo de entrenamiento fue considerablemente mayor debido a la construcción de múltiples árboles. En contraste, la **Regresión Logística** fue mucho más rápida, convirtiéndose en una opción más adecuada cuando el tiempo de entrenamiento es crucial. El **SVM**, aunque competitivo en rendimiento, fue el modelo menos eficiente en términos computacionales, lo que limitó su aplicabilidad en casos donde la eficiencia es una prioridad.

**Conclusión**  
En términos generales, **Random Forest** fue el mejor modelo para capturar la complejidad del dataset, pero su alto costo computacional puede no ser adecuado en todos los escenarios. **Regresión Logística** fue eficiente y fácil de interpretar, y **SVM** se posicionó como un modelo intermedio, ofreciendo un buen balance entre precisión y tiempos de entrenamiento, pero con un costo computacional elevado.

* + 1. **Regresión sobre el California Housing Dataset**
* **Análisis Exploratorio de Datos (EDA)**

Antes de aplicar cualquier modelo, se realizó un análisis exploratorio de los datos para comprender mejor las características del conjunto de datos. Este proceso permitió identificar patrones, distribuciones y posibles relaciones entre las variables que podrían ser útiles para la predicción del precio medio de las casas. Los principales pasos del EDA fueron:

Distribuciones de las Variables: Se analizaron las distribuciones de las principales variables, como median\_house\_value, median\_income, households y population. Esto permitió observar asimetrías en las distribuciones, como en el caso de median\_house\_value, donde se detectó una distribución sesgada a la derecha, indicando la presencia de valores extremos.

Relaciones entre Variables: Se investigaron las relaciones entre las diferentes variables a través de gráficos de dispersión y mapas de calor de correlación. Se observó, por ejemplo, que median\_income tiene una fuerte correlación positiva con median\_house\_value, lo que indica que a mayor ingreso medio de los hogares, mayor es el valor de las casas. Esta relación era esperada, pero otras correlaciones como la densidad poblacional mostraron un comportamiento más complejo y difícil de modelar con técnicas lineales.

Identificación de Valores Atípicos (Outliers): Se identificaron valores atípicos en algunas variables, como en households y population, donde algunos distritos presentaban poblaciones extremadamente altas en comparación con el resto. Estos valores atípicos podrían afectar negativamente el rendimiento de los modelos, por lo que fueron considerados en el preprocesamiento.

* **Preprocesamiento de Datos**

El preprocesamiento es una fase crítica para asegurar que los datos sean adecuados para alimentar a los modelos. En este caso, aplicamos diversas técnicas:

Codificación de Variables Categóricas: La variable ocean\_proximity es categórica, lo que significa que debe ser transformada en una forma numérica para ser utilizada por los algoritmos de machine learning. Para ello, se utilizó One-Hot Encoding, que crea una nueva columna para cada categoría, asignando un valor binario (0 o 1) dependiendo de si la observación pertenece o no a esa categoría.

Escalado de Datos: Algunas características como median\_income y population tienen escalas diferentes, lo que puede afectar el rendimiento de los modelos. Para normalizar estas variables, se utilizó el Standard Scaler, que estandariza los datos restando la media y dividiendo por la desviación estándar, asegurando que todas las características tengan la misma importancia.

Manejo de Datos Faltantes: Aunque el dataset era relativamente limpio, cualquier valor faltante fue gestionado mediante el método de imputación por media, reemplazando los valores nulos con el promedio de la columna. Esta técnica es sencilla pero efectiva cuando la cantidad de datos faltantes es pequeña.

* **Entrenamiento de los Modelos**

Dos de los algoritmos utilizados en este proyecto fueron Árboles de Decisión y Random Forest, los cuales son modelos basados en árboles de decisión. Ambos fueron entrenados y además se utilizó **GridSearchCV** para optimizar sus hiperparámetros.

* **Evaluación de los Modelos**

Error Cuadrático Medio (RMSE): El Random Forest obtuvo un RMSE de 36,419 en el conjunto de prueba, considerablemente menor que el RMSE del Árbol de Decisión (46,374). Esto indica que el Random Forest es más preciso en sus predicciones. En el conjunto de entrenamiento, el RMSE del Random Forest fue 13,905, mientras que el Árbol de Decisión mostró un RMSE de 37,764.

Error Absoluto Medio (MAE): De manera similar, el MAE del Random Forest en el conjunto de prueba fue de 25,248, en comparación con 32,769 del Árbol de Decisión. En los datos de entrenamiento, el MAE del Random Forest fue mucho menor (9,709) que el del Árbol de Decisión (27,199), lo que sugiere un mejor ajuste del modelo.

* **Análisis de Resultados**

Rendimiento del Modelo (RMSE y MAE)

En este proyecto, se observó que el modelo de Random Forest superó consistentemente al Árbol de Decisión, con un menor MAE y RMSE, demostrando su capacidad para modelar relaciones no lineales y manejar conjuntos de datos grandes y complejos.

Elección de las Métricas: RMSE y MAE

Se seleccionaron RMSE y MAE como las principales métricas debido a que proporcionan una visión clara de la precisión del modelo al predecir valores continuos. El RMSE penaliza más los errores grandes, mientras que el MAE es más robusto frente a valores atípicos. Ambas métricas permiten evaluar la precisión y la robustez del modelo.

Balance Eficacia-Eficiencia

Eficacia: El Random Forest demostró ser el modelo más preciso en términos de RMSE y MAE, lo que sugiere una mejor capacidad para modelar las relaciones no lineales en los datos.

Eficiencia: Sin embargo, el Random Forest también tomó mucho más tiempo para entrenarse debido a su complejidad. El Árbol de Decisión, aunque menos preciso, se entrenó mucho más rápido.

**Conclusiones**

Random Forest es el modelo más adecuado para este problema debido a su precisión superior en la predicción de valores.

El Árbol de Decisión ofrece una opción más rápida para escenarios donde el tiempo de entrenamiento es una limitación, pero es menos preciso.

En escenarios donde se requiere tanto precisión como tiempos de entrenamiento cortos, se debe considerar el uso de Random Forest con ajustes adicionales para equilibrar rendimiento y eficiencia.

### Clusterización

El flujo de trabajo en este proyecto sigue dos etapas principales: la generación de puntos aleatorios alrededor de centros predefinidos y la aplicación de algoritmos de clustering para agrupar dichos puntos. A continuación se describe cada una de estas etapas.

En la primera etapa, se utiliza un generador de puntos aleatorios que distribuye los puntos uniformemente alrededor de tres centros predefinidos. Cada punto se genera mediante un ángulo aleatorio y una distancia aleatoria dentro de un radio especificado en torno a cada centro. Esto garantiza que los puntos estén distribuidos en torno a los centros de manera uniforme. El proceso se repite hasta que se alcanzan 1000 puntos.

En la segunda etapa, se aplican los algoritmos K-Means y DBSCAN sobre los puntos generados. K-Means divide los puntos en un número predefinido de clusters, mientras que DBSCAN agrupa puntos que están densamente empaquetados y trata los puntos restantes como ruido. Los resultados de ambos métodos se visualizan y se comparan utilizando métricas de evaluación de clustering.

* **Análisis Crítico de los Resultados**

En este análisis, se compararon los algoritmos **K-Means** y **DBSCAN** para agrupar los puntos generados. A continuación, se presenta una comparación crítica de los resultados:

Eficacia y Precisión

Ambos algoritmos mostraron un buen rendimiento en términos de **homogeneidad** y **completitud**, obteniendo un **V-Measure** de 1.00, lo que indica una clasificación perfecta de los puntos generados. Sin embargo, K-Means presentó una **inercia** de 334.83, lo que sugiere que los puntos estaban más compactos alrededor de sus centroides. Este valor de inercia indica una alta precisión en la compactación de los clusters.

Por otro lado, el **Silhouette Score** para ambos algoritmos fue de 0.63, lo que sugiere una separación razonable entre los clusters, aunque no perfecta. Un Silhouette Score más alto indicaría que los clusters están mejor separados. En este aspecto, ambos algoritmos mostraron un rendimiento similar.

Eficiencia y Tiempos de Entrenamiento

En términos de eficiencia, **K-Means** es generalmente más rápido que **DBSCAN**, especialmente en conjuntos de datos grandes, ya que K-Means tiene una complejidad computacional de O(n), donde *n* es el número de puntos. En este caso, K-Means es más eficiente porque calcula los centroides en pocas iteraciones, ajustando los puntos rápidamente.

**DBSCAN**, por otro lado, tiene una complejidad de O(n log n), lo que puede ser más costoso en términos de tiempo de ejecución, especialmente si los clusters son densos o si hay muchos puntos ruidosos. Sin embargo, DBSCAN tiene la ventaja de detectar automáticamente el número de clusters y manejar bien los datos ruidosos, lo que K-Means no puede hacer sin predefinir el número de clusters.

Selección de Métricas

Se eligieron métricas como la **inercia**, el **Silhouette Score**, y la **V-Measure** debido a su capacidad para capturar diferentes aspectos del rendimiento de los algoritmos. La inercia es adecuada para K-Means ya que refleja cuán compactos están los clusters. Sin embargo, no se utilizó en DBSCAN porque este algoritmo no optimiza centroides, por lo que sería irrelevante.

El **Silhouette Score** fue seleccionado porque proporciona una visión sobre la separación de los clusters, lo que es crucial en un análisis comparativo. La **homogeneidad**, **completitud** y **V-Measure** fueron seleccionadas porque permiten evaluar la pureza de los clusters y si los puntos de una clase están correctamente agrupados. Estas métricas son más informativas que simplemente evaluar la distancia entre puntos.

**Conclusión**

En general, **K-Means** fue más eficiente en términos de tiempo de entrenamiento y compactación de clusters, pero **DBSCAN** fue superior en su capacidad para manejar ruido y descubrir clusters sin necesidad de predefinir *k*. La elección del algoritmo depende del contexto: si se necesita rapidez y se conoce el número aproximado de clusters, K-Means es una mejor opción. Sin embargo, si se trabaja con datos con ruido o de formas irregulares, DBSCAN es preferible debido a su flexibilidad.

En términos de las métricas seleccionadas, se buscó un equilibrio entre precisión y simplicidad, utilizando métricas como la inercia para capturar la compactación de K-Means y el Silhouette Score para evaluar la separación entre clusters. La evaluación crítica mostró que, aunque ambos algoritmos son efectivos en este conjunto de datos, el tiempo de entrenamiento y la capacidad para manejar ruido hacen que DBSCAN sea más adecuado para situaciones más complejas.

1. **CONCLUSIONES**

El desarrollo de este trabajo ha permitido un acercamiento al ámbito del aprendizaje automático, facilitando la familiarización con conceptos clave y la implementación de modelos prácticos.

Se estudiaron conceptos claves concernientes a los fundamentos del machine learning, incluyendo los distintos tipos de aprendizaje (supervisado, no supervisado y por refuerzo), así como las tareas específicas de clasificación, regresión y clusterización. Este conocimiento ha sido esencial para entender las aplicaciones y limitaciones de cada enfoque.

Se siguió un flujo de trabajo estructurado y se logró una organización eficiente del proceso, desde la recolección y preprocesamiento de datos hasta la evaluación de modelos. Este enfoque no solo facilitó la ejecución de las tareas, sino que también garantizó la reproducibilidad de los resultados.

Se realizaron evaluaciones de los modelos desarrollados para cada tarea, utilizando métricas adecuadas para medir su rendimiento. Esta etapa fue crucial para identificar áreas de mejora y optimización en los modelos, así como para entender la efectividad de diferentes algoritmos.

El desarrollo de este trabajo ha proporcionado un acercamiento al machine learning, permitiendo una comprensión integral del proceso y sus aplicaciones. Las lecciones aprendidas servirán como un pilar fundamental para futuras investigaciones y desarrollos en el campo del aprendizaje automático.

1. **BIBLIOGRAFÍA**

Albon, C. (2018). *Machine Learning with Python Cookbook.* Estados Unidos de América: O’Reilly Media.

Fenner, M. E. (2020). *Machine Learning with Python for Everyone.* Estados Unidos de América: Addison Wesley.

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow.* Estados Unidos de América: O’Reilly Media.