UNIVERSIDAD INTERNACIONAL DEL ECUADOR

FACULTAD DE CIENCIAS TÉCNICAS

ESCUELA DE INGENIERÍA MECATRÓNICA

ENDOMORFISMO II

PROYECTO 2

Integrantes

Adrian Ochoa
Wilhelm Montalvo

Quito, 14 de mayo del 2024
Marzo – Junio 2024

Proyecto 2: Clasificacion de dataset

Proyect 2: Dataset classification

[1]Escuela de Ingeniería Mecatrónica, Universidad Internacional del Ecuador, Quito, Ecuador
*Autor Principal/Corresponding author, e-mail: adochoada@uide.edu.ec

***Abstract*—**
Classifying datasets in Visual Studio Code is crucial for efficiently organizing information, enhancing accessibility, and manipulating relevant data in software projects. It fosters team collaboration by promoting a coherent structure and data reuse across different project components. In summary, this practice significantly contributes to efficient data management in software development.

***Keywords*—Classification, datasets, Visual Studio Code, efficiency, collaboration**

***Resumen*—**
La clasificación de datasets en Visual Studio Code es esencial para organizar la información de manera eficiente, mejorando la accesibilidad y manipulación de datos relevantes en proyectos de software. Facilita la colaboración entre equipos al promover una estructura coherente y la reutilización de datos en diferentes partes del proyecto. En resumen, esta práctica contribuye significativamente a una gestión eficiente de datos en el desarrollo de software.

***Palabras Clave*—Clasificación, conjuntos de datos, Visual Studio Code, eficiencia, colaboración.**

I. Introducción

La clasificación de datasets en Visual Studio Code (VSCode) es una práctica esencial en el desarrollo de software moderno. Con la creciente complejidad de las aplicaciones y la enorme cantidad de datos que se manejan, la capacidad de organizar y acceder fácilmente a conjuntos de datos relevantes se vuelve crucial. VSCode ofrece herramientas y funcionalidades que permiten a los desarrolladores estructurar y organizar sus datasets de manera efectiva, lo que simplifica el proceso de desarrollo y mejora la colaboración entre equipos.

Esta clasificación no se limita únicamente a la organización de archivos, sino que abarca una variedad de estrategias para categorizar y etiquetar los datos de manera significativa. Desde la segmentación por tipo de datos hasta la clasificación por funcionalidad o contexto de uso, los desarrolladores pueden adoptar enfoques flexibles según las necesidades específicas de sus proyectos. Esta capacidad de clasificación proporciona una visión clara y estructurada del conjunto de datos disponible, facilitando la navegación y la localización de información relevante en cualquier etapa del desarrollo.

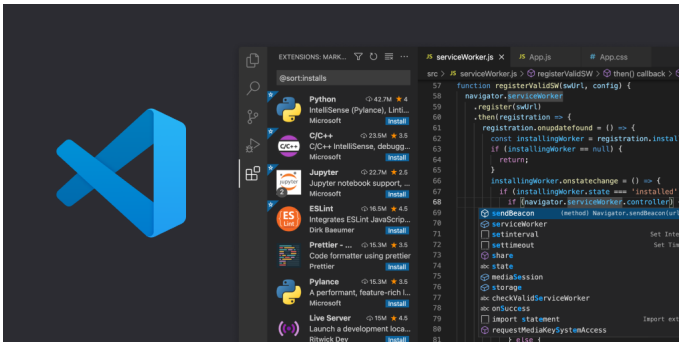

Fig. 1. Logo VSCode

II. Metodología

A. *Variables*

Las variables categóricas y numéricas son dos tipos fundamentales de datos utilizados en análisis estadístico y aprendizaje automático. Las variables categóricas representan distintas categorías o grupos que no tienen un orden inherente entre ellos. Por ejemplo, el color de los ojos o la marca de un automóvil son variables categóricas, ya que no se pueden ordenar de manera significativa. En contraste, las variables numéricas representan cantidades numéricas y pueden ser continuas o discretas. Estas variables pueden tener un orden y se pueden operar matemáticamente, como la edad de una persona o el precio de un producto.

Una diferencia clave entre variables categóricas y numéricas radica en la forma en que se analizan y se utilizan en modelos estadísticos. Las variables categóricas suelen requerir técnicas especiales de codificación, como la codificación one-hot, para poder ser utilizadas en algoritmos de aprendizaje automático. Por otro lado, las variables numéricas se pueden utilizar directamente en la mayoría de los modelos, ya que se pueden interpretar y operar matemáticamente. Además, las variables numéricas suelen proporcionar una mayor cantidad de información y flexibilidad en el análisis, permitiendo una exploración más detallada de los datos y la identificación de patrones y tendencias.

Al tener en cuenta estos conceptos, se utiliza la función LabelEncoder de la biblioteca scikit.learn para poder transformar las variables categóricas a numéricas y así poder trabajar de manera más efectiva con todos los datos. [1]

B. Imputación de datos

La imputación de datos representa una tarea crucial en el análisis de datos, especialmente cuando se enfrentan valores faltantes o incompletos. En esencia, implica rellenar esos vacíos mediante el uso de técnicas estadísticas o algoritmos específicos. Estos métodos pueden variar desde enfoques simples, como sustituir los valores faltantes con la media o la mediana de la variable, hasta estrategias más complejas que se basan en modelos predictivos.

Dentro de un entorno de programación dedicado al análisis de datos, la imputación de datos adquiere una importancia vital para garantizar la integridad y fiabilidad de los conjuntos de datos empleados. La habilidad para llevar a cabo esta tarea de manera efectiva puede tener un impacto significativo en la precisión y validez de los análisis y modelos construidos a partir de esos datos. Sin embargo, es crucial seleccionar cuidadosamente el método de imputación más adecuado, teniendo en cuenta la naturaleza de los datos y el contexto específico del problema en cuestión, con el fin de evitar sesgos o distorsiones en los resultados finales. En el caso pertinente, se utiliza la función SimpleImputer de scikitlearn y se reemplazan los datos erroneos por la mediana de todos los datos medidos.

III. Desarrollo

El código empieza ingresando la ubicación del archivo CSV que contiene los datos a analizar. Posteriormente, se utiliza la biblioteca pandas para cargar estos datos en un DataFrame llamado 'data', lo que permite su manipulación y procesamiento en Python. Luego, se emplea un codificador de etiquetas (LabelEncoder) para transformar las variables categóricas del DataFrame en variables numéricas. Este proceso es esencial para que los algoritmos de aprendizaje automático puedan trabajar con estas variables de manera efectiva, ya que muchos de ellos requieren entradas numéricas en lugar de categóricas.

En la siguiente sección del código, se lleva a cabo un proceso de imputación de valores cero en ciertas columnas donde estos valores no deberían estar presentes. Esto se realiza mediante la creación de un imputador simple (SimpleImputer) que reemplaza los valores faltantes (NaN) con la mediana de cada columna correspondiente.

Después, se realiza una transformación adicional en la columna 'SP' para convertirla en una variable binaria, donde 'Si' se representa como 1 y 'No' como 0. Esta transformación es útil cuando se desea utilizar esta columna como objetivo en un modelo de clasificación binaria, facilitando así el proceso de entrenamiento y evaluación del modelo. En conjunto, estos pasos preparan los datos para su posterior análisis y modelado, asegurando que estén en un formato adecuado y coherente para su uso en aplicaciones de aprendizaje automático.

Se preparan y se dividen los datos en un conjunto de entrenamiento y un conjunto de prueba) utilizando la función train_test_split. Aquí, X representa las características o atributos de los datos, mientras que y representa las etiquetas o clases que se están prediciendo. El parámetro test_size = 0.3 indica que el 30 por ciento de los datos se reservarán para el conjunto de prueba, mientras que random_state=40 asegura que la división de los datos sea reproducible.

Una vez divididos los datos, se procede a crear y entrenar un modelo de clasificación, en este caso, un clasificador de bosques aleatorios (RandomForestClassifier). Este modelo se entrena utilizando el conjunto de entrenamiento, donde se especifican ciertos hiperparámetros como el número de estimadores y la profundidad máxima del árbol. Posteriormente, el modelo entrenado se utiliza para predecir las etiquetas correspondientes al conjunto de prueba, generando así las predicciones.

Finalmente, se evalúa el rendimiento del modelo utilizando diversas métricas de evaluación, como la exactitud (accuracy), la matriz de confusión, la precisión (precision), la sensibilidad (recall) y el F1-Score. Estas métricas proporcionan una visión completa del desempeño del modelo en la clasificación de las clases. Al imprimir estas métricas en la consola, los desarrolladores pueden analizar y comprender mejor cómo se comporta el modelo y si es adecuado para su propósito previsto.

IV. Pruebas y resultados

Los resultados muestran que el modelo de clasificación logra una exactitud del 83 %, lo que significa que el 83 % de las predicciones realizadas por el modelo son correctas. Sin embargo, al observar la matriz de confusión, se evidencia que hay un total de 6681 predicciones correctas de la clase

negativa (verdaderos negativos) y 750 predicciones correctas de la clase positiva (verdaderos positivos), pero también hay 1231 falsos positivos y 338 falsos negativos. Esta discrepancia sugiere que el modelo podría estar teniendo dificultades para clasificar correctamente las instancias de la clase positiva.

Además, al analizar las métricas de precisión, sensibilidad y F1-Score, se observa que la precisión del modelo es del 69 %, lo que indica la proporción de predicciones positivas que fueron correctas. Sin embargo, la sensibilidad, que mide la proporción de instancias positivas que se clasificaron correctamente, es del 38 %, lo que sugiere que el modelo puede estar teniendo dificultades para identificar correctamente las instancias de la clase positiva. El F1-Score, que combina la precisión y la sensibilidad en una sola métrica, es del 49 %, lo que indica un equilibrio entre ambas medidas y refleja el rendimiento general del modelo en términos de la precisión y el recall. En resumen, aunque el modelo muestra una alta exactitud, es importante considerar estas métricas adicionales para comprender su desempeño completo y tomar decisiones informadas sobre su uso y mejora.

V. Conclusiones

- La clasificación de datasets enriquece significativamente la eficiencia en el desarrollo de proyectos al organizar la información de manera sistemática y accesible. Esta estructuración facilita la manipulación y el análisis de datos, permitiendo a los equipos trabajar de manera más fluida y productiva.

- Asimismo, esta práctica fomenta la colaboración entre equipos al establecer una estructura coherente para compartir y reutilizar los conjuntos de datos. Al tener una clasificación clara, los miembros del equipo pueden colaborar de manera más efectiva, compartiendo recursos y optimizando el flujo de trabajo.

- En un entorno de desarrollo como Visual Studio Code, la clasificación de datasets se convierte en un componente esencial para optimizar la gestión de datos. Al adoptar esta práctica, los equipos pueden mantener una organización coherente de los datos, lo que contribuye a una mayor eficiencia y calidad en el desarrollo de software.

Referencias

[1] IBM. Tipos de variables en spss statistics, 2023.