Revista esprint Investigación

https://rei.esprint.tech

ISSN: 2960-8317

Vol. 2 N° 2, julio-diciembre 2023 (26-34)

Artículo de investigación

Calidad de datos con Python: un enfoque práctico

Data quality with Python: a practical approach

Lady Marieliza Espinoza Tinoco*
Universidad Nacional de Chimborazo. Riobamba-Ecuador.
lespinoza@unach.edu.ec
https://orcid.org/0000-0001-6569-3686

Ana Elizabeth Congacha Aushay
Universidad Nacional de Chimborazo. Riobamba-Ecuador.
acongacha@unach.edu.ec
http://orcid.org/0000-0002-2488-9110

Juan Carlos Díaz Ordóñez
Escuela Superior Politécnica de Chimborazo Riobamba-Ecuador
jcdiaz@espoch.edu.ec
https://orcid.org/0000-0001-7926-484X

***Correspondencia:**
lespinoza@unach.edu.ec

Cómo citar este artículo:
Espinoza, L., Congacha, A., & Díaz, J. (2023). Calidad de datos con Python: un enfoque práctico. *Esprint Investigación*, 2(2), 26-34. https://doi.org/10.61347/ei.v2i2.55

***Recibido**: 4 de agosto de 2023*
***Aceptado:** 7 de septiembre de 2023*
***Publicado:** 18 de septiembre de 2023*

Resumen: Si bien la calidad de los datos en el análisis y toma de decisiones resulta de vital importancia, escasos estudios proporcionan pasos claros para ejecutarlos a través del lenguaje de programación Python. En consecuencia, el objetivo de la presente investigación se relaciona con diseñar una guía para evaluar y mejorar la calidad de los datos utilizando el lenguaje de programación Python. Esta investigación con enfoque cualitativo se aplica en un caso práctico medido a través de las características de calidad: Exactitud, Integridad, Libre de Errores y Valor Añadido. Los resultados indican que, mediante la aplicación de la metodología propuesta basada en 12 pasos a través de Python, los datos cumplen con las características de calidad requeridas.

Palabras clave: Calidad de datos, características de calidad, metodología, Python.

Abstract: *Although the quality of the data in the analysis and decision-making is of vital importance, few studies provide clear steps to execute them through the Python programming language. Consequently, the objective of this research is related to designing a guide to evaluate and improve the quality of data using the Python programming language. This research with a qualitative approach is applied in a practical case measured through the quality characteristics: Accuracy, Completeness, Free of Errors and Added Value. The results indicate that by applying the proposed methodology based on 12 steps through Python, the data meets the required quality characteristics.*

***Keywords**: Data quality, methodology, Python, quality characteristics.*

1. Introducción

En la era de la información, los datos generados y almacenados se han disparado exponencialmente, al emplearse en diversos campos científicos y empresariales para obtener información valiosa, tomar decisiones informadas y extraer conocimientos significativos. Sin embargo, tener acceso a grandes volúmenes de datos no garantiza automáticamente la calidad y fiabilidad de la información obtenida a partir de ellos.

La mala calidad de los datos puede afectar el rendimiento de una organización (Ridzuan & Zainon, 2019), al influir directamente en la validez y confiabilidad de los análisis y resultados. Los datos de baja calidad pueden contener errores, valores inconsistentes, registros duplicados, valores faltantes o incluso información irrelevante, lo que conduce a conclusiones erróneas o decisiones equivocadas. Además, la presencia de datos faltantes deviene un problema común que enfrentan los investigadores y científicos de datos (Jadhav et al., 2019). Por lo tanto, debe efectuarse un proceso de revisión exhaustiva de calidad de datos que garantice la integridad y precisión de los conjuntos de datos utilizados en cualquier análisis o estudio.

Ilyas & Chu (2019) afirman que los datos pueden verse afectados por los errores previamente mencionados, lo cual representa un problema para quienes trabajan con ellos. Estos errores han costado un promedio de 12.9 millones de dólares cada año a varias organizaciones (Sakpal, 2021). Para abordar este problema, Ilyas & Chu (2019) proponen dos fases para el proceso de limpieza de datos: la detección de errores, donde identifican y validan errores y la segunda, donde se reparan errores en mediante la aplicación de actualizaciones adecuadas a los datos para posteriores usos. En la fase de detección de errores se pueden usar técnicas cuantitativas o cualitativas. Las primeras emplean métodos estadísticos para identificar valores anormales, mientras las segundas se basan en enfoques descriptivos para especificar patrones o restricciones consistentes en los datos, identificando como errores aquellos datos que violan dichos patrones o restricciones.

En el estudio de Lentini (2021), además de incluir las fases mencionadas, encierra una fase previa denominada "definición de metadatos". Estos metadatos, fundamentales para comprender y mejorar la calidad de los datos, pueden ser establecidos manualmente por expertos del dominio o descubiertos automáticamente mediante un análisis automático del conjunto de datos, o una combinación de ambos métodos. Cada enfoque tiene sus ventajas y limitaciones, y la elección dependerá del alcance y la complejidad del conjunto de datos que se busca limpiar y mejorar. Por otro lado, autores como Müller & Freytag (2003) establecen una fase de "auditoría de datos" inicial y una fase final de "post procesamiento y control". Si durante esta última fase se encuentran tuplas no corregidas inicialmente, se inicia un nuevo ciclo en el proceso de limpieza de datos, comenzando por auditar los datos y buscando características en datos excepcionales.

La calidad de los datos, en esencia, se entiende como el grado en que los datos de interés satisfacen los requisitos, están libres de defectos y son adecuados para el propósito previsto (Hassenstein & Vanella, 2022). Los datos de calidad son claves para el análisis de datos, en escenarios prácticos se asocian generalmente con el preprocesamiento (Ehrlinger & Wöß, 2022). El objetivo de la preparación de datos es proporcionar datos de alta calidad (West et al., 2021).

Según estos últimos autores los criterios para evaluar la calidad de datos son Exactitud, grado en que los datos describen correctamente un objeto o evento en cuestión; Integridad, grado en que los datos están completos y tiene un carácter de suficiente; Libre de Errores, grado en que los datos proporcionados son correctos y confiables y Valor Añadido, grado en que los datos permiten obtener beneficios mediante su uso analítico.

Las técnicas que incluyen el uso de varios programas como Excel demuestran complejidad de uso, consumen más tiempo y demandan un amplio conocimiento del empleo de fórmulas (Dasari & Varma, 2022). En muchas ocasiones no son replicables, es decir, no servirán para otro conjunto de datos. Por lo que las técnicas de limpieza de datos usadas en el lenguaje de programación Python resultan más eficientes y rápidas en comparación con las de los métodos clásicos. Python como lenguaje de programación se caracteriza por su versatilidad y su código abierto que cuenta con una amplia variedad de librerías y herramientas especializadas en el procesamiento y manipulación de datos. Al automatizar tareas de limpieza, validación y transformación de datos, facilita el trabajo de los analistas y mejora la calidad de los resultados.

De acuerdo con McKinney (2011), Python ha experimentado un significativo avance en accesibilidad, al ser ampliamente utilizado en lugar de R, Matlab, Stata, SAS y otras aplicaciones científicas. Esto se ha logrado gracias a la madurez y escalabilidad de las bibliotecas fundamentales. Específicamente la librería Pandas, que de acuerdo con el Equipo de Desarrollo de Pandas (2023) es un paquete que proporciona estructuras de datos rápidas, flexibles y expresivas, diseñadas para facilitar el trabajo con datos de manera fácil e intuitiva. Ofrece, además, un manejo sencillo de datos faltantes,

la capacidad de insertar o eliminar columnas en DataFrames y alineación automática y explícita de datos mediante etiquetas para facilitar cálculos. Además, esta librería presenta una funcionalidad poderosa para la agrupación y transformación de conjuntos de datos, así como para la conversión de datos desiguales e indexados en objetos DataFrame. También sobresale en la segmentación inteligente basada en etiquetas, indexación sofisticada y combinación de conjuntos de datos. Además de que es flexible para transformar y reorganizar los datos, facilita la carga y el guardado de información desde diversos formatos. Es especialmente útil para el etiquetado jerárquico de ejes y el manejo de series de tiempo, además de tener la capacidad de combinar y unir datos provenientes de múltiples fuentes.

En la literatura científica escasos estudios documentan los comandos necesarios para el proceso de calidad de datos en el lenguaje de programación Python, como, por ejemplo, la búsqueda realizada en julio de 2023 con la cadena de búsqueda "(TITLE(python) AND TITLE("dataquality"))" en SCOPUS a nivel de título, únicamente arroja 4 documentos, de los cuales se puede rescatar el trabajo realizado por Dasari & Varma (2022) que explora las técnicas y herramientas disponibles en el mercado para la limpieza de datos, que se resumen junto con su aplicación utilizando Python. La investigación considera los diversos problemas surgidos durante la aplicación de estas técnicas y métodos, y cómo superar tales obstáculos. Sin embargo, los comandos ocupados son escasos y no representan los diferentes problemas que se pueden encontrar en el proceso de calidad de datos.

El objetivo del presente estudio es diseñar una guía para evaluar y mejorar la calidad de los datos utilizando el lenguaje de programación Python, con el fin de garantizar la utilidad de los datos utilizados en el análisis y toma de decisiones. Al desarrollar este enfoque práctico, se proporcionará el proceso de aplicación de las técnicas y herramientas de Python disponibles para abordar los desafíos comunes de calidad de datos. Esto beneficiará a profesionales, investigadores y estudiantes involucrados en proyectos de análisis de datos, ya que podrán aplicar estas técnicas de manera eficiente y efectiva para asegurar la calidad de sus datos. Además, al promover el uso de Python como herramienta principal, se contribuirá a un análisis más riguroso y a una toma de decisiones más sólida.

2. Metodología

La presente investigación tiene un enfoque mixto debido a que describe las diferentes técnicas aplicadas en el proceso de calidad de datos, además de ejecutar un caso de estudio experimental con una base de datos real de bienes raíces de 3692 registros recabados de diferentes fuentes de información web inmobiliaria de la ciudad de Riobamba, Ecuador, en el periodo comprendido desde 2015 a 2022. Se empleó la estadística descriptiva para comprobar que los datos resultantes cumplieran con las características de calidad de datos: Exactitud, Integridad, Libre de Errores y Valor Añadido. Se utilizó la plataforma Anaconda y del *Integrated Development Environment* (IDE) Spyder para la programación de los comandos. Se decidió el siguiente procedimiento para el proceso de calidad de datos:

1. Cargar las librerías necesarias.
2. Cargar e integrar datos.
3. Eliminar columnas duplicadas e inservibles.
4. Eliminar filas duplicadas.
5. Eliminar filas inservibles.
6. Estandarizar las categorías de cada columna.
7. Verificar errores en datos numéricos.
8. Realizar el tratamiento de datos vacíos.

9. Verificar que no existan valores nulos.
10. Remover duplicados.
11. Retirar o corregir valores atípicos.
12. Exportar datos.

3. Resultados

3.1. *Cargar las librerías necesarias*

Se procede a la carga de librerías, se invoca a la librería Pandas, la cual permite obtener información de archivos como CSV, Excel, SQL, HTML, entre otros; así como la manipulación, análisis e integración de datos. La línea de código es la siguiente: "*import pandas as pd*". Adicionalmente, para la detección de valores atípicos se cargó "import numpy as np" y "import matplotlib.pyplot as plt". NumPy es una biblioteca fundamental para la computación numérica en Python. Proporciona una amplia gama de funciones y herramientas para trabajar con matrices multidimensionales, realizar operaciones matemáticas y cálculos científicos eficientes. Al importar NumPy como np, se establece un alias para acceder a las funciones y objetos de NumPy de una manera más concisa en el código. Matplotlib es una biblioteca de visualización de datos en Python que se utiliza ampliamente para crear gráficos, diagramas y visualizaciones interactivas. Al importar pyplot de Matplotlib como plt, se establece un alias para acceder a las funciones y métodos de trazado de Matplotlib de una manera más conveniente en el código.

3.2. *Cargar e integrar datos*

Para la carga de datos de los archivos del caso de estudio se empleó la línea de código "*df=pd.read_excel('bienes_raices.xlsx')"*, donde *df* corresponde a la variable que obtiene los datos de un archivo de Excel con nombre de "*bienes_raices.xlsx"*, variable de tipo *DataFrame.* Así también, se puede utilizar el comando *"print(df.head())"*, con la finalidad de imprimir las primeras 5 filas, para asegurar que los datos han sido cargados.

3.3. *Eliminar columnas duplicadas e inservibles*

Antes de la eliminación de filas se debe realizar un análisis de las columnas que pueden poseer los mismos datos, o también que estas no tengan una utilidad de análisis, en muchas ocasiones es posible que columnas como teléfono, número de identificación, código postal o similares deban eliminarse debido a que no aportarán mayormente al análisis. Para eliminar columnas en Python se utiliza el comando *"df=df.drop('Nombre_Columna',axis=1)"*. También puede utilizarse para remover varias columnas a la vez *"df=df.drop(colums=(['col1','col2']))"*. En este proceso es recomendable utilizar también el comando *"print(df['Tipo de propiedad'].describe())*" que proporciona una estadística descriptiva por columnas, y el comando *"print(df.isnull().sum())"* que permite verificar el número total de valores nulos en cada columna del *DataFrame.* La base de datos presentaba 32 columnas; sin embargo, dos han sido eliminadas por presentarse duplicadas: Ciudad de la propiedad, debido a que la columna Ciudad tenía la misma información; y la Dirección exacta de la propiedad, puesto que no era posible categorizarla debido a los múltiples errores de escritura presentados.

3.4. *Eliminar filas duplicadas*

La línea de código utilizada para eliminar filas duplicadas es *"df=df.drop_duplicates()"*, adicionalmente para reiniciar el índice del *DataFrame*¸ se utiliza el siguiente comando "*df=df.reset_index(drop=True)"*, debido a que el comando de eliminación de filas duplicadas deja filas con valores no consecutivos, lo cual puede provocar confusión al observar los índices de los últimos registros.

3.5. *Eliminar filas inservibles*

La eliminación de filas inservibles dependerá de cada caso, sin embargo, los autores de este artículo científico consideran conveniente definir un porcentaje que se considere como mínimo válido, es decir, si es inferior a igual a este valor la fila será eliminada, si es mayor se conservará para aplicar algún otro tratamiento que permita completar los valores faltantes. El procedimiento se define de la siguiente manera:

- Definir el porcentaje de columnas válidas deseado 80% *"porcentaje_validas=0.8"*.
- Se cuenta las columnas válidas para cada fila *"num_validas=df.notnull().sum(axis=1)"*.
- Calcular el porcentaje de columnas válidas para cada fila *"porc_validas=num_validas/df.shape[1]"*.
- Seleccionar las filas que tengan menos porcentaje que lo permitido *"filas_validas=porc_validas>=porcentaje_validas"*.
- Eliminar las filas que no cumplan con este criterio *"df=df[filas_validas]"*.
- Reiniciar el índice del dataframe *"df=df.reset_index(drop=True)"*.

3.6. *Estandarizar las categorías de cada columna*

Para estandarizar las categorías de cada columna debe emplearse el siguiente comando que permitirá contar cuántos valores existen de cada clase; por ejemplo: "*df['Tipo de propiedad'].value_counts()"*, donde "Tipo de propiedad" es la columna que se encuentran en la base de datos. El resultado indica una serie de valores que no corresponden a las clases únicas, tal es el caso de "Casa" que se encuentra también como "C", "Terreno" que se encuentra como "Terrenos" como "T" y como "Terrenows", esta última de escrita inclusive de forma incorrecta. Por esta razón se definen los siguientes ejemplos como los casos más habituales que se pueden encontrar:

- Reemplaza "C" por "Casa", *"df['Tipo de propiedad']=df['Tipo de propiedad'].str.replace(r'\bC\b','Casa')"*. En este caso se utiliza *"r'\bC\b'"*, porque se indica que debe encontrar exactamente el valor de "C", de otra manera estaría encontrando dentro del contenido de la celda la letra "C", es decir, en el ejemplo también devolvería los resultados en el caso de encontrar la palabra "Casa".
- Reemplaza "casa" por "Casa", *"df['Tipo de propiedad']=df['Tipo de propiedad'].str.replace('casa','Casa')"*.
- Reemplaza "Terrenow" por "Terreno", *"df['Tipo de propiedad']=df['Tipo de propiedad'].str.replace('Terrenow','Terreno')"*.

3.7. *Verificar errores en datos numéricos*

Para esta etapa se puede hacer uso del siguiente comando: *"print(df['Precio'].head(10))"*, esta línea de código indica los 10 primeros valores que se encuentran en la columna "Precio". Además, se puede utilizar el comando *"df.info()"* que proporciona una descripción concisa pero útil del *DataFrame*, incluyendo la estructura de columnas, tipos de datos y estadísticas básicas sobre los valores no nulos en cada columna. Entonces con este comando se puede observar, por ejemplo, que valores Python no reconoce como numéricos o en este caso el "Precio", el que fue reconocido como "object" en lugar de Float, esto indica que no todos los datos son numéricos y que deben corregirse. Además, se puede intentar forzar con el comando "*df['Precio']=df['Precio'].astype(float)"*, el cual indicará un error debido que al existir valores de tipo cadena no permitirá transformar a valores numéricos sin corregirlos. Para encontrar los valores de cadena se puede hacer uso de las líneas de código "*cadena_valores = df.loc[df['Precio'].apply(lambda x: isinstance(x, str)), 'Precio']"* y luego *"print(cadena_valores)"*.

Posteriormente, se puede ejecutar el siguiente proceso:

- Encontrando la ubicación del problema, *"mask=df['Precio'].astype(str).str.contains('usd',case=False)"*
- Reemplazando en toda la columna donde existen problemas relacionados a usd,

"df.loc[mask,'Precio']=df.loc[mask,'Precio'].str.replace('usd','')".

- Aquí se reemplaza los valores que contengan espacios en blanco, por ningún valor, mask=df['Precio'].astype(str).str.contains(' ',case=False)
- Posteriormente ya es posible formatear la variable inicialmente de tipo object, a un valor numérico de float *"df['Precio']=df['Precio'].astype(float)"*.

3.8. *Realizar el tratamiento de datos vacíos*

Existen varias formas para el tratamiento de datos vacíos, entre ellas la estrategia de la media para los valores numéricos y la mediana; y para los valores categóricos la moda. Aun así, es posible también aplicar técnicas de Machine Learning como redes neuronales artificiales o algún algoritmo de clasificación para rellenar estos valores. Se muestran dos líneas de código, la primera rellena con la media a la columna "Precio" y la segunda con la moda a la columna "Tipo de Propiedad":

"imputer_num=SimpleImputer(strategy='mean')

imputer_cat=SimpleImputer(strategy='most_frequent')

df['Precio']=imputer_num.fit_transform(df[['Precio']])

df['Tipo de propiedad']=imputer_cat.fit_transform(df[['Tipo de propiedad']])"

3.9. *Verificar que no existan valores nulos*

Es importante ahora verificar que no existan valores nulos en la base de datos; para ello se ejecuta la siguiente línea de código "*print(df.isnull().sum())"*, que indicará el conteo de valores nulos de todas las columnas de la base de datos, estos deben indicar un valor de cero.

3.10. *Remover duplicados*

Es necesario nuevamente utilizar esta línea de código debido a que nuevos valores pudieron completarse indicando nuevas filas duplicadas *"df=df.drop_duplicates()"*, adicionalmente para reiniciar el índice del DataFrame¸ se utiliza el comando *"df=df.reset_index(drop=True)".*

3.11. *Valores atípicos*

Esta etapa de la guía es la más extensa, aquí se analizará el precio para encontrar posibles datos que no concuerden con los precios del sector, para ello se asignará a la variable data el valor de la columna precio "*data=df['Precio'].values***",** posteriormente se puede crear un histograma para observar la distribución de los datos; para ello se emplearán las siguientes instrucciones:

plt.hist(data,bins=10,edgecolor='black')"

plt.xlabel('Precio')

plt.ylabel('Frecuencia')

plt.title('Histograma de Precios')

plt.show()

Otro gráfico que puede tomarse en cuenta es el de dispersión:

plt.scatter(df.index,df['Precio'], s=20,c='b',alpha=0.5)

Ahora se realizará el cálculo de los cuartiles y el Rango Intercuartil (IQR):

q1=df['Precio'].quantile(0.25)

q3=df['Precio'].quantile(0.75)

iqr=q3-q1

Se crean los límites superior e inferior:

*upper_bound=q3+1.5*iqr*

*lower_bound=q1-1.5*iqr*

Se seleccionan los valores atípicos sean menores al límite inferior y superiores al límite superior:

outliers=df[(df['Precio']>upper_bound)|(df['Precio']<lower_bound)]

Se muestra en un gráfico de dispersión:

plt.scatter(outliers.index,outliers['Precio'],s=100,c='r',alpha=0.5)

plt.xlabel('Índice')

plt.ylabel('Precio')

plt.title('Gráfico de dispersión de Precio')

plt.show()

Se puede además imprimir los valores atípicos antes de tomar la decisión de eliminarlos, o también se puede cargar a un archivo de Excel para su verificación, en la cual se tendrá dos caminos o analizarlos por separado si los datos son reales, para lo cual se necesita una investigación previa, o si los datos -como en este caso de estudio- son irreales, es decir, que pudo existir un error en el momento de recolección debido a que o son demasiado bajos o demasiado altos se procede a eliminarlos.

print(outliers)

outliers.to_excel('valores_atipicos.xlsx',index=True)

Para eliminar los valores atípicos se utiliza la siguiente línea de código:

df=df.drop(outliers.index)

3.12. *Exportar datos*

Para la exportación de datos se utiliza el comando df.to_excel('base_limpia.xlsx',index=False), donde "base_limpia.xlsx" representa el nombre del nuevo archivo que contiene la base de datos en formato Excel y que será utilizada posteriormente para los diferentes análisis, en esto puede incluirse la revisión de valores atípicos, escalamiento de valores y la creación de variables Dummy de ser el caso.

La Tabla 1 indica un resumen de los resultados de la aplicación de la guía para calidad de datos a través del lenguaje de programación Python, se puede observar que los datos iniciales son de 3692 filas por 32 columnas, y al final del proceso han permanecido 3242 filas por 30 columnas. Sin embargo, se ha podido resolver los problemas de calidad de datos relacionados con la integridad, libre de errores, exactitud y valor añadido, en este último se hicieron varias pruebas con algoritmos de Machine Learning para comprobar que los datos son útiles para el proceso de modelado y posteriormente la toma de decisiones.

Tabla 1

Resultados de la aplicación de la guía para calidad de datos con Python

Característica	Descripción	Datos iniciales	Datos finales
Integridad	15 % de los datos presentó problemas.	3692 filas 32 columnas	3338 filas 30 columnas
Libre de Errores	El 3,7 % de registros presentó errores. 125 errores corregidos.	3338 filas 30 columnas	3338 filas 30 columnas
Exactitud	2,8 % de registros presenta problemas de exactitud. 96 registros retirados.	3338 filas 30 columnas	3242 filas 30 columnas
Valor Añadido	Los datos fueron probados y permitieron realizar distintos análisis como: Clustering, clasificación, series temporales	3338 filas 30 columnas	3242 filas 30 columnas

4. Conclusiones

La calidad de los datos se concibe como un factor crucial en cualquier análisis o estudio, debido a que influye directamente en la validez y confiabilidad de los resultados. Las técnicas de limpieza de datos en Python son más eficientes, rápidas y brindan herramientas especializadas para el procesamiento y manipulación de datos. El uso de Python como lenguaje de programación proporcionó beneficios significativos con un enfoque eficiente y efectivo para abordar los desafíos comunes de calidad de datos. Los resultados del caso de estudio indicaron que la guía de calidad de datos proporcionada en este documento cubre los criterios de Integridad, Libre de Errores, Exactitud y Valor Añadido.

En términos de trabajos futuros, varias áreas podrían ser objeto de investigación y mejora en relación con la revisión de valores atípicos, la escala de valores y la creación de variables dummy. En cuanto a la revisión de valores atípicos, a pesar de los avances en las técnicas y métodos existentes, aún existen oportunidades para explorar en mayor medida. Se podrían investigar enfoques novedosos para la detección automática de valores atípicos en conjuntos de datos grandes y complejos, así como desarrollar métodos que permitan una gestión más efectiva de estos valores, como la imputación de valores faltantes o la selección de estrategias de tratamiento adecuadas.

Referencias

Dasari, D., & Varma, P. S. (2022). Employing Various Data Cleaning Techniques to Achieve Better Data Quality using Python. In *2022 6th International Conference on Electronics, Communication and Aerospace Technology* (pp. 1379-1383). IEEE. https://doi.org/10.1109/ICECA55336.2022.10009079

Ehrlinger, L., & Wöß, W. (2022). A survey of data quality measurement and monitoring tools. *Frontiers in Big Data*, *5*, 850611. https://doi.org/10.3389/fdata.2022.850611

Equipo de Desarrollo de Pandas. (2023). *Pandas-dev/pandas*. Github. https://github.com/pandas-dev/pandas/tree/v2.0.3

Hassenstein, M., & Vanella, P. (2022). Data Quality—Concepts and Problems. *Encyclopedia*, *2*(1), 498-510. https://doi.org/10.3390/encyclopedia2010032

Ilyas, I., & Chu, X. (2019). *Data cleaning*. Morgan & Claypool. https://doi.org/10.1145/3310205

Jadhav, A., Pramod, D., & Ramanathan, K. (2019). Comparison of performance of data imputation methods for numeric dataset. *Applied Artificial Intelligence*, *33*(10), 913-933. https://doi.org/10.1080/08839514.2019.1637138

Lentini, A. (2021). *Calidad de datos y aprendizaje automático: detección de errores semánticos en datos estructurados con esquema desconocido* [Tesis de especialización, Instituto Tecnológico de Buenos Aires]. Repositorio del Instituto Tecnológico de Buenos Aires. https://ri.itba.edu.ar/entities/trabajo%20final%20de%20especializaci%C3%B3n/1d04d92e-69bf-43cf-889f-a4acb13ab040

McKinney, W. (2011). Pandas: a foundational Python library for data analysis and statistics. *Python for high performance and scientific computing*, *14*(9), 1-9. https://www.researchgate.net/publication/265194455_pandas_a_Foundational_Python_Library_for_Data_Analysis_and_Statistics

Müller, H., & Freytag, J. (2003). *Problems, methods, and challenges in comprehensive data cleansing*. Professoren des Inst. Für Informatik. https://www.researchgate.net/publication/228929938_Problems_methods_and_challenges_in_comprehensive_data_cleansing

Ridzuan, F., & Zainon, W. (2019). A review on data cleansing methods for big data. *Procedia Computer Science*, 161, 731-738. https://doi.org/10.1016/j.procs.2019.11.177

Sakpal, M. (2021). *How to improve your data quality*. Gartner. https://www.gartner.com/smarterwithgartner/how-to-improve-your-data-quality

West, N., Gries, J., Brockmeier, C., Göbel, J. C., & Deuse, J. (2021). Towards integrated data analysis quality: criteria for the application of industrial data science. In *2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI)* (pp. 131-138). IEEE. https://doi.org/10.1109/IRI51335.2021.00024

Transparencia

Conflicto de interés

Los autores declaran que no existen conflictos de interés que influyan en la objetividad de este estudio.

Fuente de financiamiento

No se recibieron fondos financieros de ninguna organización que pudiera tener interés en los resultados presentados.

Contribución de autoría

Lady Marieliza Espinoza Tinoco: Conceptualización, metodología, análisis formal, investigación, visualización, redacción - preparación del borrador original, redacción - revisión y edición, financiamiento, administración del proyecto, recursos, supervisión.

Ana Elizabeth Congacha Aushay: Conceptualización, software, validación, gestión de datos, visualización, redacción - preparación del borrador original, redacción - revisión y edición, financiamiento, recursos, supervisión.

Juan Carlos Díaz Ordóñez: Investigación, redacción - preparación del borrador original, redacción - revisión y edición, financiamiento, recursos, supervisión.

Los autores contribuyeron activamente en el análisis de los resultados, revisión y aprobación del manuscrito final.