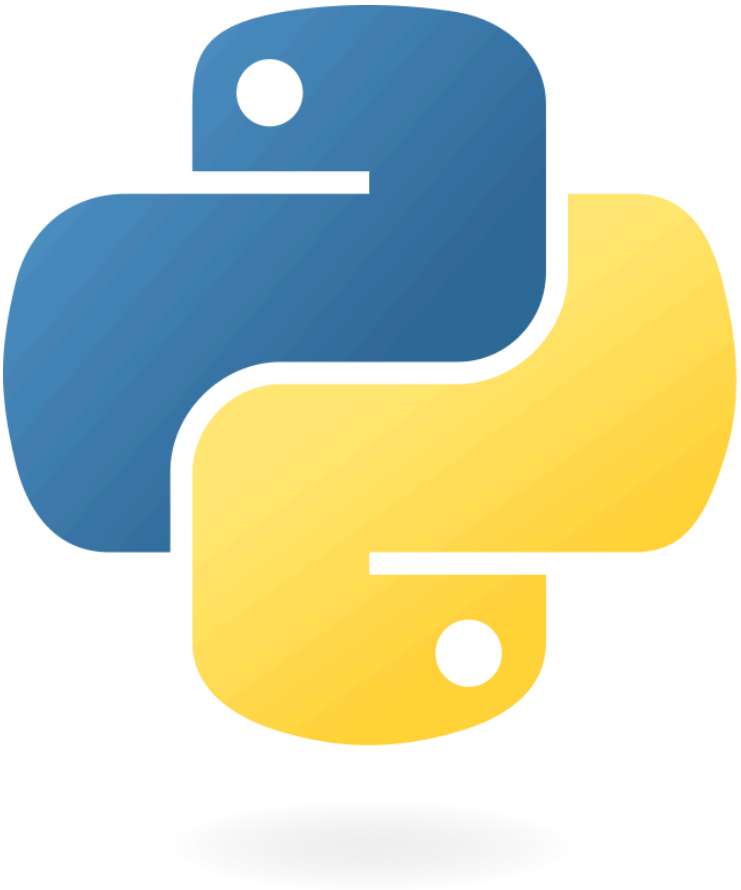
Introducción al Análisis de Datos con Pandas y Matplotlib en Python



Iván Arana Hernández

ivanrnhdez

ivanaranahdez

ivan.aranahdez@gmail.com

Índice

1. Introducción..... 2

2. Descripción del proyecto..... 2

3. Objetivos de Aprendizaje..... 3

4. Estructura del Proyecto..... 3

5. Funciones y Códigos Implementados..... 4
5.1. Preprocesamiento de datos..... 4
5.1.1. Funciones creadas para el preprocesamiento de datos..... 4
5.1.2. Funciones de Pandas utilizadas..... 5
5.2. Análisis Estadístico..... 6
5.2.1. Funciones creadas para el análisis estadístico..... 6
5.2.2. Funciones de Pandas utilizadas..... 6
5.3. Visualización de Datos..... 8
5.3.1. Funciones creadas para la visualización de datos..... 8
5.3.2. Funciones de Matplotlib y NumPy utilizadas..... 8
5.4. Generación de Informes en PDF..... 10
5.4.1. Función creada en la generación de informes..... 10
5.4.2. Funciones de FPDF utilizadas..... 10

6. Mejoras y Posibles Extensiones..... 11

7. Conclusión..... 12

8. Uso de Inteligencia Artificial en el Desarrollo..... 12

1. Introducción

Este proyecto ha sido una primera toma de contacto con el análisis de datos en Python, utilizando herramientas como Pandas para la manipulación de datos y Matplotlib para la visualización. A través de un conjunto de datos sencillo, que incluye información sobre estudiantes y sus calificaciones, se han explorado funciones básicas de limpieza, análisis y representación gráfica de los datos.

El principal propósito ha sido aprender a manejar estas bibliotecas de manera práctica, entendiendo cómo cargar, limpiar y visualizar datos de forma estructurada. Se han aplicado conceptos como la eliminación de valores nulos, el cálculo de estadísticas básicas y la creación de distintos tipos de gráficos.

Este informe resume las funciones implementadas y los resultados obtenidos, destacando cómo se han utilizado gráficos de barras, dispersión y circulares para representar la información de manera más comprensible. Además, se ha experimentado con la generación de informes en PDF, lo que permite documentar los análisis realizados de manera clara y estructurada.

2. Descripción del proyecto

El proyecto se basa en el uso de dos datasets que contienen información sobre estudiantes, sus asignaturas, calificaciones, edad, género y ciudad. Antes de comenzar el análisis, fue necesario realizar una unión de estos datasets, combinando la información de los alumnos con sus respectivas calificaciones de manera estructurada. Como primer paso en el procesamiento de datos, se aplicó una limpieza básica, eliminando valores nulos y duplicados, además de manejar casos específicos como estudiantes con calificaciones faltantes o no presentados.

Una vez los datos estuvieron preparados, se desarrollaron funciones para calcular estadísticas clave, como la media, la varianza y la desviación estándar de las calificaciones por asignatura, la matriz de correlación entre la edad y las calificaciones, etc. Para representar visualmente estos datos de manera clara, se implementaron diversos gráficos con Matplotlib, como gráficos de barras para mostrar la media de calificaciones por asignatura, gráficos combinados que muestran la cantidad de estudiantes y la media de calificaciones por ciudad, gráficos de dispersión para analizar la relación entre la edad y la calificación (incorporando una línea de tendencia para observar posibles correlaciones) y gráficos circulares para visualizar la distribución de género en la muestra de estudiantes.

Finalmente, para documentar los resultados de forma estructurada, se implementó una función que genera informes en PDF, permitiendo al usuario seleccionar qué gráfico analizar, ingresar un título y redactar un análisis detallado sobre las visualizaciones generadas. De esta manera, el proyecto no solo facilita la exploración y visualización de los datos, sino que también proporciona una herramienta para presentar los hallazgos. Este proyecto ha servido como una introducción práctica al análisis de datos con Pandas y Matplotlib, ayudando a comprender cómo unir datasets, limpiar información, calcular métricas clave y representar visualmente los resultados. Aunque el alcance ha sido introductorio, ha sentado una base sólida para aplicar estos conocimientos en proyectos más avanzados en el futuro.

3. Objetivos de Aprendizaje

Los objetivos de este proyecto se centraron en aprender y aplicar herramientas clave para el análisis y visualización de datos en Python.Los principales aprendizajes fueron:

- **Uso de Pandas:** Manipulación, limpieza y unión de datasets.
- **Visualización con Matplotlib:** Creación de gráficos como barras, dispersión y circulares.
- **Generación de informes:** Uso de FPDF para exportar análisis en PDF.
- **Estructuración de código:** Mejora en la organización y buenas prácticas en Python.

4. Estructura del Proyecto

El proyecto está organizado de manera modular para facilitar su gestión y escalabilidad. La carpeta data almacena los conjuntos de datos, diferenciando entre raw (datos sin procesar) y processed (datos listos para análisis). Los resultados generados se guardan en outputs, donde graficos contiene las visualizaciones creadas y analisis almacena los informes en PDF. Finalmente, el código modular se encuentra en scripts, permitiendo una separación clara de responsabilidades y facilitando la reutilización y ampliación del código.

5. Funciones y Códigos Implementados

5.1. Preprocesamiento de datos

En esta sección se detallan las funciones creadas para limpiar, procesar y combinar los datasets utilizados en el proyecto. El preprocesamiento de datos es un paso fundamental para garantizar que la información sea precisa, consistente y lista para su análisis.

5.1.1. Funciones creadas para el preprocesamiento de datos

- **cargar_datos_sin_duplicados(path):** Esta función carga los datos desde un archivo CSV y elimina registros duplicados, asegurando que cada fila sea única.

- **tratar_datos_alumnos(df):** Se enfoca en limpiar la información de los alumnos eliminando filas sin nombre o ciudad y asegurando que no haya nombres repetidos en el dataset.

- **tratar_datos_calificaciones(df):** Gestiona la información de las calificaciones, asignando un valor de -1 a los estudiantes que no presentaron exámenes, eliminando valores nulos y asegurando que no haya registros duplicados.

- **combinar_datos(df1, df2, path):** Fusiona los dos datasets a través de la columna "Nombre", asignando nuevos identificadores (ID_Alumno e ID_Calificacion), reordenando los datos y guardando el resultado en un nuevo archivo CSV.

5.1.2. Funciones de Pandas utilizadas

Durante el preprocesamiento de datos, se emplearon diversas funciones de Pandas:

- **pd.read_csv(path):** Permite cargar datos desde un archivo CSV en un DataFrame, facilitando su manipulación y análisis.

- **drop_duplicates():** Elimina filas repetidas en un DataFrame, asegurando que cada registro sea único según los criterios especificados.

- **dropna(subset=[...]):** Filtra y elimina filas que contengan valores nulos en determinadas columnas, garantizando datos más completos.

- **fillna(valor):** Rellena los valores nulos con un valor específico, útil para tratar datos faltantes sin eliminarlos.

- **pd.merge(df1, df2, on=..., how=...)**: Une dos DataFrames basándose en una columna común, permitiendo la combinación y estructuración de datos provenientes de diferentes fuentes.

- **sort_values(by=[...], ascending=[...])**: Ordena un DataFrame en función de una o varias columnas, facilitando la organización de los datos.

- **to_csv(path, index=False):** Guarda un DataFrame en un archivo CSV, permitiendo su almacenamiento o uso posterior en otros procesos.

Estas funciones permiten organizar, limpiar y preparar los datos de manera eficiente para su análisis y visualización posterior.

5.2. Análisis Estadístico

En esta sección se presentan las funciones diseñadas para realizar un análisis estadístico de los datos, permitiendo identificar patrones y tendencias clave. Se han implementado cálculos de métricas relevantes, como distribuciones, promedios y relaciones entre variables, con el objetivo de extraer información útil para la interpretación de los datos.

5.2.1. Funciones creadas para el análisis estadístico

- **analisis_datos_calificaciones(df):** Esta función evalúa el desempeño académico de los estudiantes. Filtra aquellos que se presentaron a los exámenes, calcula el porcentaje de aprobados totales y por asignatura, y obtiene la media de calificaciones por materia.

- **analisis_datos_asignaturas(df):** Analiza la distribución de género en cada asignatura, encuentra a los tres estudiantes con mejores calificaciones por materia y calcula medidas de dispersión como la varianza y la desviación estándar.

- **exploracion_datos(df):** Proporciona un análisis general de los datos, incluyendo estadísticas descriptivas, correlaciones entre edad y calificación, distribución de calificaciones por rangos y un estudio de rendimiento por género y asignatura.

5.2.2. Funciones de Pandas utilizadas

Para realizar el análisis estadístico, se emplearon diversas funciones de Pandas:

- **df[df['Columna'] != valor] / df[df['Columna'] >= valor]:** Filtra las filas del DataFrame que cumplen una condición.

- **groupby('Columna').size():** Agrupa datos por una columna y cuenta la cantidad de elementos en cada grupo.

- **groupby('Columna')['Otra_Columna'].mean():** Calcula la media de una columna dentro de cada grupo.

- **groupby('Columna')['Otra_Columna'].agg([...])**: Aplica múltiples funciones estadísticas a los grupos (media, mediana, varianza, desviación estándar).

- **groupby('Columna').head(n):** Obtiene las primeras n filas de cada grupo ordenado.

- **unstack(fill_value=0):** Convierte una agrupación en formato tabla y rellena valores nulos con un valor específico.

- **fillna(valor)**: Reemplaza valores nulos con un valor específico.

- **sort_values(by=[...], ascending=[...]):** Ordena los datos según columnas específicas.

- **reset_index():** Restablece los índices del DataFrame tras operaciones de agrupación.

- **pd.cut(df['Columna'], bins=[...], labels=[...], right=False):** Clasifica valores numéricos en rangos.

- **df[['Col1', 'Col2']].describe().transpose():** Calcula estadísticas generales de las columnas seleccionadas.

- **df[['Col1', 'Col2']].corr():** Genera la matriz de correlación entre variables numéricas.

- **df['Columna'].value_counts().sort_index():** Cuenta la cantidad de ocurrencias de cada categoría y las ordena

Estas herramientas permiten obtener información clave sobre el rendimiento académico de los estudiantes, facilitando su análisis e interpretación.

5.3. Visualización de Datos

En esta sección se presentan las funciones creadas para visualizar la información de manera gráfica, facilitando la interpretación de los datos. Se han diseñado distintos tipos de gráficos para analizar distribuciones, relaciones y tendencias en el rendimiento académico de los estudiantes.

5.3.1. Funciones creadas para la visualización de datos

- **grafica_medias_asignaturas(df):** Genera un gráfico de barras con la media de calificaciones por asignatura, lo que permite comparar el rendimiento en diferentes materias.

- **grafico_combinado_por_ciudad(df):** Crea un gráfico combinado que muestra la cantidad de estudiantes por ciudad con un diagrama de barras y la media de calificaciones por ciudad con una línea de tendencia.

- **generar_graficos_edad_calificacion_genero(df):** Produce un gráfico de dispersión para visualizar la relación entre edad y calificación, añadiendo una línea de tendencia, y un gráfico circular que representa la distribución de género en los datos.

5.3.2. Funciones de Matplotlib y NumPy utilizadas

Para la generación de gráficos, se han empleado diversas funciones de Matplotlib:

- **plt.figure(figsize=(ancho, alto)):** Define el tamaño del gráfico.

- **plt.subplots(nrows, ncols, figsize=(ancho, alto)):** Crea múltiples subgráficos en una misma figura.

- **df.plot(kind='bar', color=..., edgecolor=...):** Crea un gráfico de barras con estilo personalizado.

- **plt.scatter(x, y, color=..., alpha=...):** Dibuja un gráfico de dispersión.

- **plt.plot(x, y, color=..., marker=..., linewidth=...):** Traza una línea, utilizada para la tendencia en gráficos de dispersión.

- **df.plot(kind='pie', autopct='%1.1f%%', colors=...):** Crea un gráfico circular con porcentajes.

- **plt.xlabel('Etiqueta') / plt.ylabel('Etiqueta'):** Agrega etiquetas a los ejes.

- **plt.title('Título')**: Establece el título del gráfico.

- **plt.xticks(rotation=..., ha=...):** Ajusta la rotación y alineación de las etiquetas del eje X.

- **plt.tight_layout():** Optimiza la distribución de los elementos en la figura.

- **plt.show():** Muestra el gráfico generado.

- **plt.savefig(ruta, dpi=...):** Guarda el gráfico en un archivo con una resolución específica.

- **ax.twinx():** Crea un segundo eje Y para superponer gráficos en una misma figura.

- **ax1.tick_params(axis=..., labelcolor=...)**: Ajusta los parámetros del eje, personalizando el color de las etiquetas.

- **fig.legend(loc=..., bbox_to_anchor=...):** Agrega una leyenda en la figura combinada.

Además de Matplotlib, se han empleado funciones de NumPy para realizar ajustes matemáticos en los gráficos:

- **np.polyfit(x, y, grado):** Ajusta una función polinómica de grado específico a un conjunto de datos.

- **np.poly1d(coeficientes):** Crea una función polinómica a partir de los coeficientes obtenidos en polyfit.

- **poly1d_fn(x):** Evalúa la función polinómica en los valores de x, generando una línea de tendencia en el gráfico de dispersión.

Estas funciones permiten representar visualmente los datos, ayudando a identificar patrones y tendencias en el rendimiento académico de los estudiantes.

5.4. Generación de Informes en PDF

En esta sección se presentan las funciones diseñadas para generar informes en formato PDF a partir de gráficos previamente creados. Estos informes permiten documentar los análisis realizados sobre los datos visualizados, facilitando su almacenamiento y distribución.

5.4.1. Función creada en la generación de informes

- **generar_pdf():** Solicita al usuario el nombre del gráfico a analizar y recopila un conjunto de observaciones sobre la imagen. Luego, genera un informe en PDF que incluye el gráfico, un título y una lista de análisis ingresados por el usuario.

5.4.2. Funciones de FPDF utilizadas

Para la creación y estructuración del informe en PDF, se han empleado las siguientes funciones de la biblioteca fpdf:

- **FPDF():** Inicializa un nuevo documento PDF.
- **pdf.set_auto_page_break(auto=True, margin=valor):** Configura el ajuste automático de saltos de página, con un margen específico.
- **pdf.add_page():** Agrega una nueva página al documento.
- **pdf.set_font("fuente", style='...', size=valor):** Establece la fuente, el estilo (B: negrita, I: cursiva, U: subrayado) y el tamaño del texto.
- **pdf.cell(ancho, alto, texto, ln=..., align='...'):** Crea una celda con el texto especificado, permitiendo definir su alineación y si se ubica en una nueva línea.
- **pdf.ln(espacio):** Agrega un salto de línea con el espacio especificado.
- **pdf.image("ruta", x=..., y=..., w=...)**: Inserta una imagen en el PDF con coordenadas y ancho definido.

- **pdf.multi_cell(ancho, alto, texto):** Crea una celda de varias líneas para manejar textos extensos.

- **pdf.output("ruta"):** Guarda el documento PDF en la ruta especificada.

Estas funciones permiten estructurar de manera ordenada los informes generados, asegurando su correcta presentación y legibilidad.

6. Mejoras y Posibles Extensiones

Dado que este es un proyecto de iniciación, existen múltiples mejoras y ampliaciones que podrían implementarse en futuras versiones:

- **Automatización del análisis:** Integrar opciones para generar automáticamente los informes en PDF tras la creación de un gráfico sin necesidad de intervención manual.

- **Interfaz gráfica (GUI):** Desarrollar una aplicación con una interfaz amigable para facilitar la interacción sin necesidad de ejecutar scripts manualmente.

- **Mayor personalización de gráficos**: Incluir opciones avanzadas de configuración, como colores personalizados, títulos dinámicos y ajustes de diseño.

- **Exportación a otros formatos:** Ampliar la generación de informes en formatos como Excel o HTML.

- **Análisis avanzado:** Implementar técnicas de machine learning para realizar predicciones o clasificaciones sobre los datos analizados.

- **Integración con bases de datos:** Permitir la carga y almacenamiento de datos en bases de datos SQL o NoSQL para mejorar la escalabilidad.

7. Conclusión

Este proyecto demuestra la importancia de la modularidad y la estructuración en el desarrollo de herramientas de análisis de datos. Mediante la combinación de bibliotecas como pandas, matplotlib, numpy y fpdf, se logra un flujo de trabajo eficiente para la exploración, visualización y documentación de información.

La implementación de funciones organizadas y reutilizables facilita la expansión del código sin comprometer su legibilidad. Además, la estructura del proyecto permite un mantenimiento sencillo y una clara separación de responsabilidades, lo que lo hace adecuado para futuras mejoras.

En conclusión, este código representa una sólida base para proyectos de análisis de datos, con múltiples posibilidades de expansión y optimización en futuras iteraciones.

8. Uso de Inteligencia Artificial en el Desarrollo

Durante el desarrollo de este proyecto, la inteligencia artificial ha sido una herramienta clave para optimizar el código, mejorar su legibilidad y corregir errores en la lógica de programación. Además, ha facilitado la generación de documentación, estructurando el contenido de manera clara y comprensible. También ha sido útil para proponer mejoras y posibles ampliaciones del proyecto, permitiendo adoptar buenas prácticas desde el inicio. Sin embargo, es fundamental revisar y comprender cada sugerencia antes de implementarla, asegurando que se adapte correctamente a los objetivos del proyecto y manteniendo el control total sobre el desarrollo del código.