Informe

Analisis del Rendimiento Estudiantil en Egipto

Introduccion a la ciencia de datos

Fatima Sanchez Dominguez

Profesor: Jaime Alejandro Romero Sierra

Lunes, 25 de noviembre del 2024

INTRODUCCIÓN

Objetivo del proyecto

El objetivo principal del proyecto es analizar y predecir el rendimiento estudiantil a partir de un conjunto de datos que incluyen calificaciones, factores demograficos y otros aspectos. Además, el objetivo era crear un dashboard interactivo para facilitar la interpretación de los resultados y apoyar la toma de decisiones

Justificación y Contexto

El rendimiento estudiantil es un indicador clave del éxito académico y refleja las oportunidades y desafíos que enfrentan los estudiantes. Identificar los factores que influyen en su utilidad nos permite desarrollar estrategias efectivas para mejorar los resultados educativos, eliminar desigualdades y personalizar los métodos de enseñanza. Este análisis también es relevante para la planificación académica y administrativa.

Fuentes de datos

- Origen: Base de datos extraída de kaggle.com
- Cantidad de datos: La base de datos cuenta con 19905 filas y 17 columnas
- Principales características: Variables, como las calificaciones en diversas materias("Subjects" dentro de la base), edad, nivel educativo de los padres y las categorias de rendimiento (alto, medio, bajo).

METODOLOGIA

LIMPIEZA DE DATOS

Proceso de limpieza de datos:
Primero se comenzó con un análisis inicial de la base de datos, esta contaba con 58717 filas y 16 columnas

```
import pandas as pd
df=pd.read_csv('https://raw.githubusercontent.com/fluvwonie/World-/refs/heads/main/df_sucio.csv')

df
```

✓ 7.4s

	Student Name	Student Age	Student year	Father Degree	Mother Degree	Education Type	Subject_1	Subject_2	Subject_3	Subj
0	Allison Lang	18	Year 9	High School	NaN	IB	72.23673953300614	85.931149	65.69851271532883	84.1
1	Jaclyn Mcneil	14	Year 9	Bachelor	NaN	IB	91.60307770488868	73.186427	64.24023932483671	47.7
2	Melissa Lee	16	Year 10	NaN	NaN	IB	100.0	83.985923	89.37816212685267	82.4
3	Timothy Maxwell	14	Year 10	Bachelor	NaN	Thanweya	91.08213123885534	100.000000	NaN	78.4
4	Eric Steele	17	Year 11	PhD	NaN	IGCSE	74.90614373713879	69.518146	65.28484123859717	56.3
...	...	...	...	...	...	...	...	...	...	...
58712	James Keller DDS	NaN	Year 12	bbb	NaN	IB	NaN	75.364662	53.68002843577125	89.5
58713	David Gray	14	Year 9	NaN	NaN	IB	53.46423003012429	64.215446	79.76916592237677	84.6
58714	Madeline Craig	17	Year 10	Master	NaN	IB	66.0345725185885	62.075037	64.59572115651984	78.0
58715	Joshua Castillo	16	Year 12	PhD	NaN	IGCSE	69.9623531312404	100.000000	75.5069568660999	99.7
58716	Marie Smith	14	Year 12	High School	NaN	IB	bbb	84.768347	75.27755926444465	66.4

58717 rows × 16 columns

Después de eso realice un análisis estadístico de los datos y pude identificar valores nulos en la columna 'Mother Degree', y que había algunas inconsistencias en los Subjects.

```
df.describe()
```

✓ 0.0s

	Mother Degree	Subject_2	Subject_4	Subject_6	Subject_7	Subject_9
count	0.0	56369.000000	56369.000000	56369.000000	56369.000000	56369.000000
mean	NaN	74.716320	74.766456	74.727108	74.742337	74.732483
std	NaN	14.379012	14.373929	14.405412	14.326311	14.350040
min	NaN	20.000000	20.000000	20.000000	20.000000	20.000000
25%	NaN	64.785208	64.857749	64.845467	65.025966	65.015658
50%	NaN	74.959388	75.017521	75.050251	75.041457	75.050616
75%	NaN	85.187657	85.224727	85.211527	85.111905	85.169166
max	NaN	100.000000	100.000000	100.000000	100.000000	100.000000

+ Code + Mark

Add Mark

Calcule el porcentaje de los valores faltantes por columnas y cree un DataFrame aparte para que pudiera visualizar los resultados, pude darme cuenta que la columna con mayor porcentaje de nulos era "Mother Degree" con un 100%.

```
# Calcular el porcentaje de valores faltantes por columna
porcentaje_nulos = df.isnull().mean() * 100

# Crear un DataFrame con los resultados
tabla_nulos = pd.DataFrame({
    'Columna': df.columns,
    'Porcentaje de Nulos': porcentaje_nulos
})

# Mostrar la tabla
tabla_nulos
```

	Columna	Porcentaje de Nulos
Student Name	Student Name	3.998842
Student Age	Student Age	3.998842
Student year	Student year	3.998842
Father Degree	Father Degree	23.238585
Mother Degree	Mother Degree	100.000000
Education Type	Education Type	3.998842
Subject_1	Subject_1	3.998842
Subject_2	Subject_2	3.998842
Subject_3	Subject_3	3.998842
Subject_4	Subject_4	3.998842
Subject_5	Subject_5	3.998842
Subject_6	Subject_6	3.998842
Subject_7	Subject_7	3.998842
Subject_8	Subject_8	3.998842
Subject_9	Subject_9	3.998842
Subject_10	Subject_10	3.998842

Revise si había valores duplicados en las columnas y pedí un resumen descriptivo del DataFrame para identificar los tipos de datos con los que estaría trabajando, y pude notar que algunas de las características no coincidan con su tipo de dato.

```
df.duplicated()
```

```
0        False
1        False
2        False
3        False
4        False
         ...
58712    False
58713    False
58714    False
58715    False
58716    False
Length: 58717, dtype: bool
```

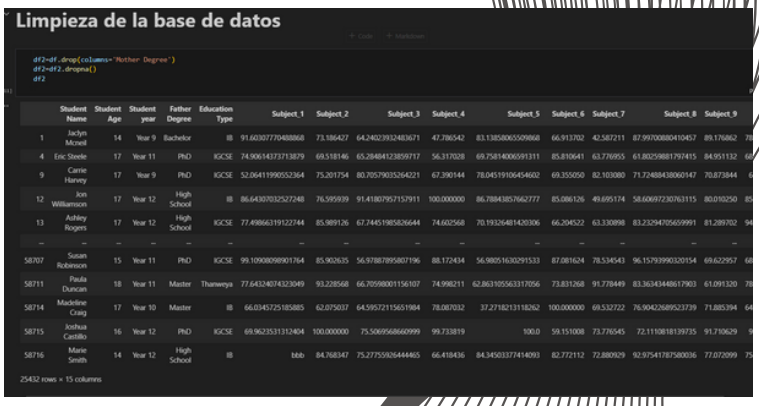
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58717 entries, 0 to 58716
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Student Name    56369 non-null  object
 1   Student Age     56369 non-null  object
 2   Student year    56369 non-null  object
 3   Father Degree   45072 non-null  object
 4   Mother Degree   0 non-null      float64
 5   Education Type  56369 non-null  object
 6   Subject_1       56369 non-null  object
 7   Subject_2       56369 non-null  float64
 8   Subject_3       56369 non-null  object
 9   Subject_4       56369 non-null  float64
 10  Subject_5       56369 non-null  object
 11  Subject_6       56369 non-null  float64
 12  Subject_7       56369 non-null  float64
 13  Subject_8       56369 non-null  object
 14  Subject_9       56369 non-null  float64
 15  Subject_10      56369 non-null  object
dtypes: float64(6), object(10)
```

METODOLOGIA

Procedí a la limpieza de datos, comencé por eliminar la columna "Mother Degree" debido que todos los datos de la columna eran valores nulos, e igual elimine los demás valores nulos de las otras columnas.

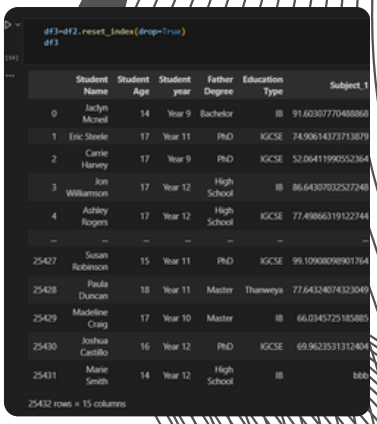
Al final nos quedan 25,432 filas y 15 columnas.




```
df2.isnull().sum()
```

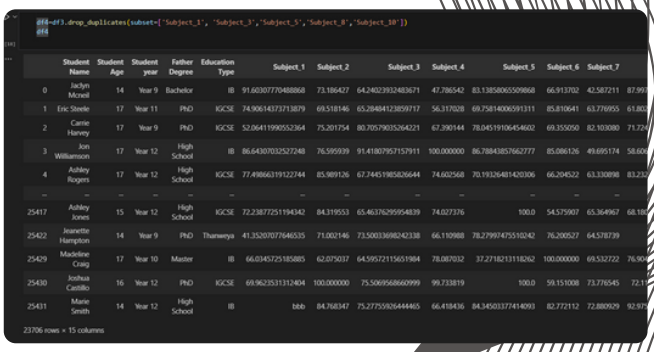
```
Student Name       0
Student Age        0
Student year       0
Father Degree      0
Education Type     0
Subject_1          0
Subject_2          0
Subject_3          0
Subject_4          0
Subject_5          0
Subject_6          0
Subject_7          0
Subject_8          0
Subject_9          0
Subject_10         0
dtype: int64
```

Comprobamos que los datos nulos se hayan eliminado correctamente.



Hice un reseteo del índice para poder trabajar mucho mejor con los datos

Volvi a comprobar si habia datos duplicados, esta vez seleccione las caracteristicas Subject 1, 3, 5, 8 y 10 y habia en total 1726 duplicados, procedi a eliminarlos, y al final obtuve 23, 706 filas



Al hacer la limpieza pude notar que había datos basura, en este caso los "bbb". Lo que hice para poderlos eliminar fue remplazarlos por NaN, esto para el caso de las variables de tipo object, para las de tipo numérico utilice pd.to_numeric con errors='coerce' para convertir todo lo que no sea numérico en NaN, y elimine los valores nulos.

```
remplazo=['Student Name', 'Student year','Father Degree','Education Type']
df4[remplazo] = df4[remplazo].replace('bbb', '', regex=True)


print(df4[remplazo])
```

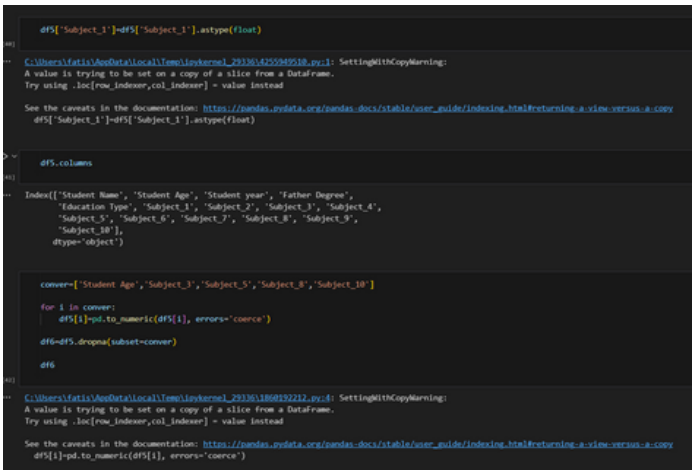
```
          Student Name Student year Father Degree Education Type
0        Jaclyn Mcneil      Year 9      Bachelor             IB
1          Eric Steele     Year 11           PhD          IGCSE
2        Carrie Harvey      Year 9           PhD          IGCSE
3      Jon Williamson     Year 12   High School             IB
4        Ashley Rogers     Year 12   High School          IGCSE
...                ...         ...           ...            ...
25417     Ashley Jones     Year 12   High School          IGCSE
25422  Jeanette Hampton      Year 9           PhD       Thanweya
25429    Madeline Craig     Year 10        Master             IB
25430    Joshua Castillo     Year 12           PhD          IGCSE
25431       Marie Smith     Year 12   High School             IB

[23706 rows x 4 columns]
```

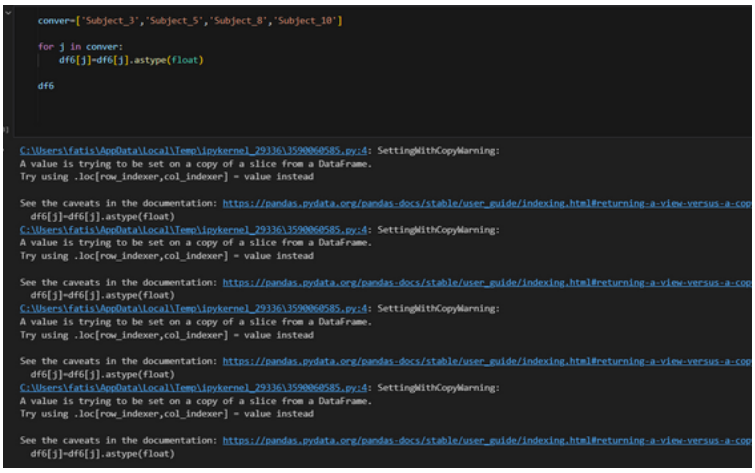
```
# Usar pd.to_numeric con errors='coerce' para convertir todo lo que no sea numérico en NaN
df4['Subject_1'] = pd.to_numeric(df4['Subject_1'], errors='coerce')

# Eliminar las filas que contienen NaN
df5 = df4.dropna(subset=['Subject_1'])


df5
```

```
C:\Users\fatis\AppData\Local\Temp\ipykernel_29336\1275435976.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df4['Subject_1'] = pd.to_numeric(df4['Subject_1'], errors='coerce')
```

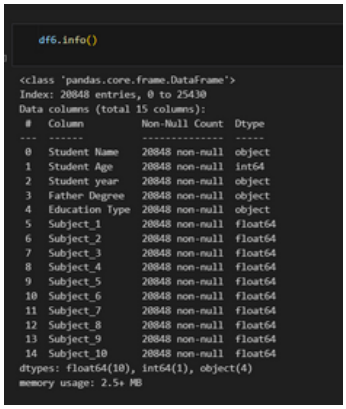
METODOLOGIA

Pase convertir los datos a su tipo de dato, a los “Subjects” que estaban como tipo de dato “object”, los convertí a decimal o “float”, igualmente con los datos de tipo “object“ los cambie a lo que les correspondía .

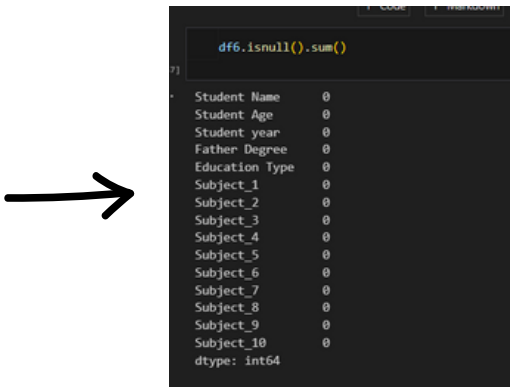




Comprobé que los datos hayan cambiado a su tipo de dato, verifique que no hubieran nuevamente valores nulos, volví a calcular el porcentaje de los valores nulos, ya no había ninguno, y por ultimo guarde la base de datos limpia, para proceder a realizar el EDA o Análisis Exploratorio.



```
df6.info()

<class 'pandas.core.frame.DataFrame'>
Index: 20848 entries, 0 to 25430
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Student Name    20848 non-null  object
 1   Student Age     20848 non-null  int64
 2   Student year    20848 non-null  object
 3   Father Degree   20848 non-null  object
 4   Education Type  20848 non-null  object
 5   Subject_1       20848 non-null  float64
 6   Subject_2       20848 non-null  float64
 7   Subject_3       20848 non-null  float64
 8   Subject_4       20848 non-null  float64
 9   Subject_5       20848 non-null  float64
 10  Subject_6       20848 non-null  float64
 11  Subject_7       20848 non-null  float64
 12  Subject_8       20848 non-null  float64
 13  Subject_9       20848 non-null  float64
 14  Subject_10      20848 non-null  float64
dtypes: float64(10), int64(1), object(4)
memory usage: 2.5+ MB
```



```
df6.isnull().sum()

Student Name      0
Student Age       0
Student year      0
Father Degree     0
Education Type    0
Subject_1         0
Subject_2         0
Subject_3         0
Subject_4         0
Subject_5         0
Subject_6         0
Subject_7         0
Subject_8         0
Subject_9         0
Subject_10        0
dtype: int64
```

```
porcentaje_nulos = df6.isnull().mean() * 100

tabla_nulos = pd.DataFrame({
    'Columna': df6.columns,
    'Porcentaje de Nulos': porcentaje_nulos
})

tabla_nulos
```

	Columna	Porcentaje de Nulos
Student Name	Student Name	0.0
Student Age	Student Age	0.0
Student year	Student year	0.0
Father Degree	Father Degree	0.0
Education Type	Education Type	0.0
Subject_1	Subject_1	0.0
Subject_2	Subject_2	0.0
Subject_3	Subject_3	0.0
Subject_4	Subject_4	0.0
Subject_5	Subject_5	0.0
Subject_6	Subject_6	0.0
Subject_7	Subject_7	0.0
Subject_8	Subject_8	0.0
Subject_9	Subject_9	0.0
Subject_10	Subject_10	0.0

```
df6.to_csv('base_de_datos_limpia.csv', index=False)
df6
```

Analisis Exploratorio (EDA)

EDA

Descripción general de los datos

Numero total de registros:
20848 filas × 15 columnas

Tipos de variables:

Numericas: 'Student Age', 'Subject_1', 'Subject_2', 'Subject_3', 'Subject_4', 'Subject_5', 'Subject_6', 'Subject_7', 'Subject_8', 'Subject_9', 'Subject_10'

Categoricas:'Student Name', 'Student year', 'Father Degree', 'Education Type'

Estadisticas descriptivas:

Media: La media se encuentra entre 74 y 75

Desviación estándar (std): Esta es de 14

Minimo y maximo: el mínimo de calificaciones es de 20, y en edad la edad mínima es de 14, mientras que el máximo en calificaciones es de 100 y en edad del estudiante es de 18

Mostrar los datos estadisticos

+ Code

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Student Age	20848.0	16.012279	1.419104	14.000000	15.000000	16.000000	17.000000	18.0
Subject_1	20848.0	74.810691	14.313873	23.115667	65.009898	75.203132	85.069442	100.0
Subject_2	20848.0	74.675569	14.366906	20.000000	64.693881	74.963166	85.122421	100.0
Subject_3	20848.0	74.750117	14.360173	20.349550	64.961283	75.146877	85.107413	100.0
Subject_4	20848.0	74.641910	14.357677	20.000000	64.777327	74.853444	85.116447	100.0
Subject_5	20848.0	74.551177	14.332123	20.000000	64.748061	74.798934	85.031051	100.0
Subject_6	20848.0	74.606782	14.478923	20.000000	64.572515	74.912569	85.164899	100.0
Subject_7	20848.0	74.642770	14.322241	20.000000	64.857336	75.058072	84.872813	100.0
Subject_8	20848.0	74.710475	14.346535	20.000000	64.912285	74.907386	85.111145	100.0
Subject_9	20848.0	74.642344	14.334672	22.138571	64.915281	74.972165	85.051653	100.0
Subject_10	20848.0	74.588469	14.406369	20.000000	64.725069	74.892297	84.917098	100.0

Visualización y Distribución de las variables

Primero comprobar los valores únicos en cada columna

```
#Comprobar valores unicos en cada columna
for column in df.columns:
    print(column, df[column].nunique())
```

[6] ✓ 0.0s

```
Student Name 18214
Student Age 5
Student year 4
Father Degree 4
Education Type 3
Subject_1 19862
Subject_2 19885
Subject_3 19838
Subject_4 19860
Subject_5 19853
Subject_6 19826
Subject_7 19890
Subject_8 19853
Subject_9 19897
Subject_10 19828
```

EDA

Al ver los valores, tenia como objetivo encontrar cuales eran los nombres mas frecuentes, por lo que realice un conteo de los nombres.

```
nombres_similares = df['Student Name'].value_counts()[:30]

nombres_similares
```

✓ 0.0s

```
Student Name
Michael Davis        11
Christopher Smith    10
Michael Smith        10
James Smith           8
William Johnson       8
James Jones           8
Matthew Miller        8
Michael Johnson       7
Nicholas Davis        7
David Johnson         7
Michael Brown         7
Anthony Smith         7
James Johnson         7
Michael Jones         7
Lisa Williams         6
David Williams        6
Jennifer Clark        6
Robert Williams       6
Samantha Smith        6
William Smith         6
John Williams         6
Robert Garcia         6
Jennifer Williams     6
Robert Wilson         6
...
Jason Miller          5
Robert Martinez       5
```

```
#Filtrado del nombre
df[df['Student Name']=='Michael Davis']
```

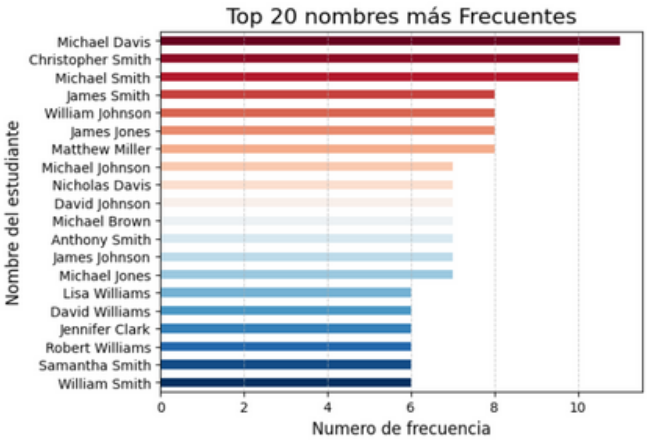
✓ 0.0s

	Student Name	Student Age	Student year	Fath Degr
6963	Michael Davis	15	Year 9	Pl
9236	Michael Davis	14	Year 10	High Scho
9294	Michael Davis	14	Year 11	Mast
9762	Michael Davis	18	Year 12	High Scho
10220	Michael	18	Year 9	Mast

El nombre de Michael Davis era el mas común, por lo que realice un filtrado con su nombre, para ver las características que este tenia

```
nombres_frecuentes = df['Student Name'].value_counts().nlargest(20)

colors = plt.cm.RdBu(np.linspace(0, 1, len(nombres_frecuentes)))

nombres_frecuentes.plot(kind='barh', color=colors)

plt.title('Top 20 nombres más Frecuentes', fontsize=16)
plt.xlabel('Numero de frecuencia', fontsize=12)
plt.ylabel('Nombre del estudiante', fontsize=12)

plt.grid(True, which='both', axis='x', linestyle='--', linewidth=0.7, alpha=0.7)

plt.gca().invert_yaxis()

plt.show()
```

Por ultimo realice un grafico que me permitiera visualizar mejor los datos.



Por ultimo, tenemos que Michael Davis es el nombre mas frecuente, y el menos frecuente es Wiliam Smith.

EDA

Realice un análisis de la edad del estudiante y el tipo de educación, para ver la distribucion.

Analisis de la edad del estudiante y del tipo de educacion

```
df['Student Age'].value_counts()
```

✓ 0.0s

```
Student Age
18    4268
17    4168
15    4150
14    4149
16    4113
Name: count, dtype: int64
```

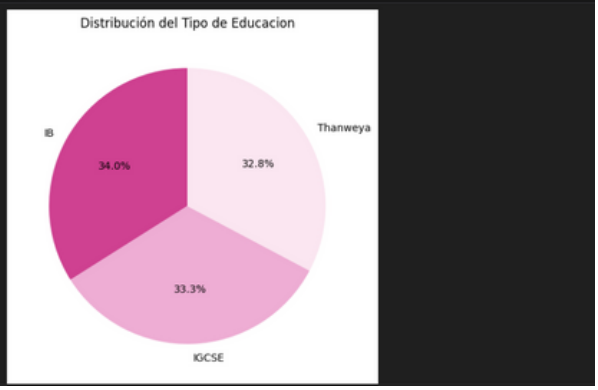
```
student_counts = df['Student Age'].value_counts()

# Gráfico de pastel
color = sns.color_palette('BrBG')

plt.figure(figsize=(6,6))
plt.pie(student_counts, labels=student_counts.index, autopct='%1.1f%%', startangle=90, colors=color)
plt.title('Distribución por Edad del Estudiante')
plt.show()
```

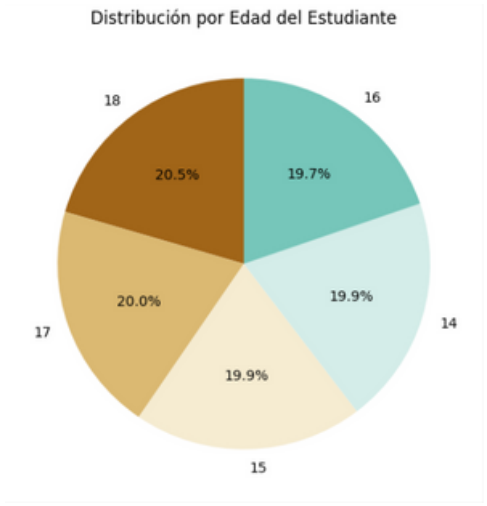
✓ 0.1s

cree dos gráficos de pastel para cada categoría para ver su distribución por separado.

```
edu_counts = df['Education Type'].value_counts()

# Gráfico de pastel
color = sns.color_palette('PiYG')

plt.figure(figsize=(6,6))
plt.pie(edu_counts, labels=edu_counts.index, autopct='%1.1f%%', startangle=90, colors=color)
plt.title('Distribución del Tipo de Educacion')
plt.show()
```

✓ 0.0s






```
df['Student Age'].groupby(df['Education Type']).value_counts()
```

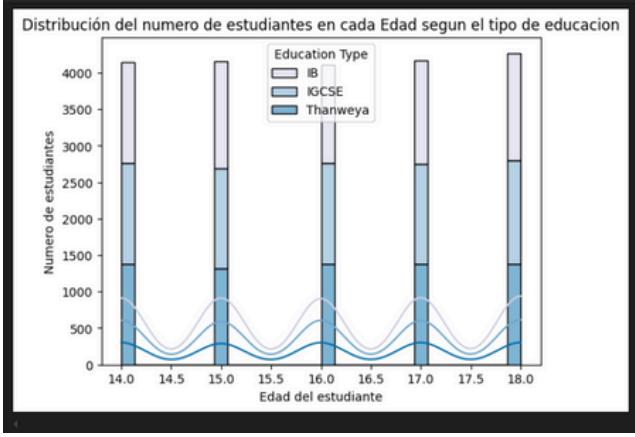
✓ 0.0s

```
Education Type  Student Age
IB              18             1465
                15             1463
                17             1419
                14             1385
                16             1352
IGCSE           18             1422
                16             1388
                14             1382
                17             1373
                15             1370
Thanweya        14             1382
                18             1381
                17             1376
                16             1373
                15             1317
Name: count, dtype: int64
```

Las agrupe y note las edades mas frecuentes en cada tipo de educación:
IB: 18 como la edad mas frecuente
IGSE: Igualmente con 18 como la edad mas frecuente
Thanweya: Con 14 con la edad mas frecuente

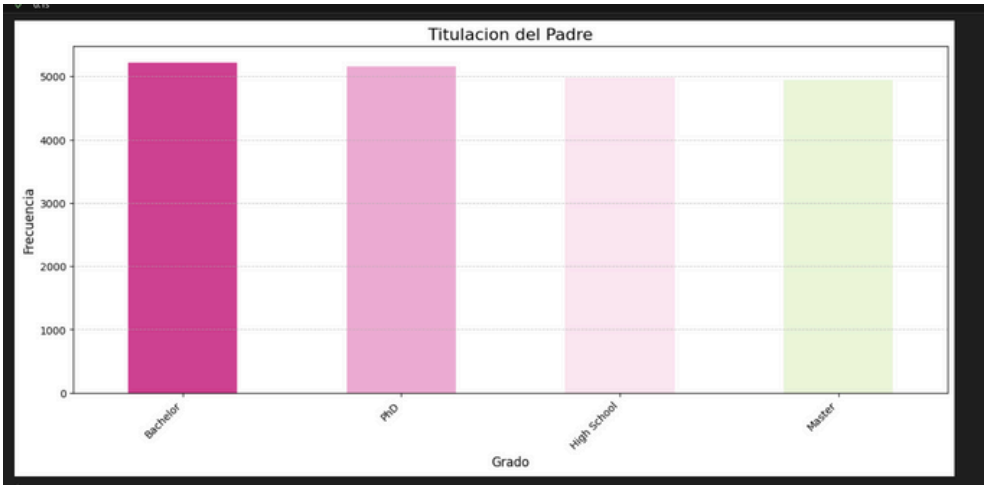
```
sns.histplot(data=df, x='Student Age',hue ='Education Type', multiple="stack",palette="PuBu",kde=True, bins=30)

# Mostrar el gráfico
plt.title('Distribución del numero de estudiantes en cada Edad segun el tipo de educacion')
plt.xlabel('Edad del estudiante')
plt.ylabel('Numero de estudiantes')
plt.show()
```

✓ 0.7s



Creando un hitsplot se puede observar mejor la distribución.

Procedí a realizar un análisis del grado del padre:



Grafique para poder visualizar la frecuencia del grado, siendo la mas frecuente "Bachelor" o "Licenciatura" y la menos frecuente "Master" o "Maestria"

Analisis de notas

Analisis de las notas

```
#Calculamos la media aritmetica de los subject en relacion con las otras columnas
df[['Subject_1','Subject_2','Subject_3','Subject_4','Subject_5',
    'Subject_6','Subject_7','Subject_8','Subject_9','Subject_10']].groupby(df['Student Age']).mean()
```

✓ 0.0s

Student Age	Subject_1	Subject_2	Subject_3	Subject_4	Subject_5	Subject_6	Subject_7	Subject_8	Subject_9	Subject_10
14	74.493784	74.851820	74.537487	74.681918	74.884124	74.783601	74.651548	74.674427	74.411788	74.667066
15	74.631563	74.713539	74.804485	74.619519	74.898197	74.433482	74.626489	74.780726	74.541414	74.302572
16	74.993764	74.408468	74.868541	74.274138	74.351128	74.755203	74.712016	74.819406	74.779088	74.629330
17	74.776957	74.467506	74.817660	74.828607	74.447431	74.512632	74.757473	74.719174	74.657201	74.816119
18	75.149458	74.927900	74.723871	74.796884	74.184184	74.552316	74.471320	74.563739	74.818324	74.528364

```
df[['Subject_1','Subject_2','Subject_3','Subject_4','Subject_5','Subject_6','Subject_7',
    'Subject_8','Subject_9','Subject_10']].groupby(df['Education Type']).mean()
```

✓ 0.0s

Education Type	Subject_1	Subject_2	Subject_3	Subject_4	Subject_5	Subject_6	Subject_7	Subject_8	Subject_9	Subject_10
IB	74.948562	74.486677	74.657852	74.445896	74.581052	74.599309	74.500502	74.572163	74.554942	74.537455
IGCSE	74.599856	74.812279	74.655707	74.854704	74.611066	74.515238	74.646482	74.642873	74.750787	74.607488
Thanweya	74.881780	74.732683	74.941704	74.629147	74.459367	74.707499	74.786581	74.922603	74.622883	74.622074

Calcule la media aritmetica de subject, en relacion con otras columnas, para detrminar si estas tenian que ver con el promedio de las materias

EDA

```
df[['Subject_1','Subject_2','Subject_3','Subject_4','Subject_5','Subject_6',
    'Subject_7','Subject_8','Subject_9','Subject_10']].groupby(df['Father Degree']).mean()
```

[21] ✓ 0.0s

Father Degree	Subject_1	Subject_2	Subject_3	Subject_4	Subject_5	Subject_6	Subject_7	Subject_8	Subject_9	Subject_10
Bachelor	74.903958	74.574100	74.768129	74.710740	74.216090	74.501609	74.681606	74.823176	74.669835	74.711770
High School	74.446821	74.483792	74.747233	74.663475	74.888706	74.899667	74.415868	74.655217	74.521568	74.790704
Master	74.991576	74.829853	74.836120	74.442313	74.590777	74.434830	74.999734	74.437547	74.647804	74.193369
PhD	74.860269	74.812272	74.612726	74.826713	74.488546	74.626893	74.471782	74.872019	74.701075	74.661017

La media aritmética es de casi 75 y es la misma dependiendo, la edad, el grado del padre y el tipo de educación

```
df['Student Name'][Top_10_students.index]
```

[24] ✓ 0.0s

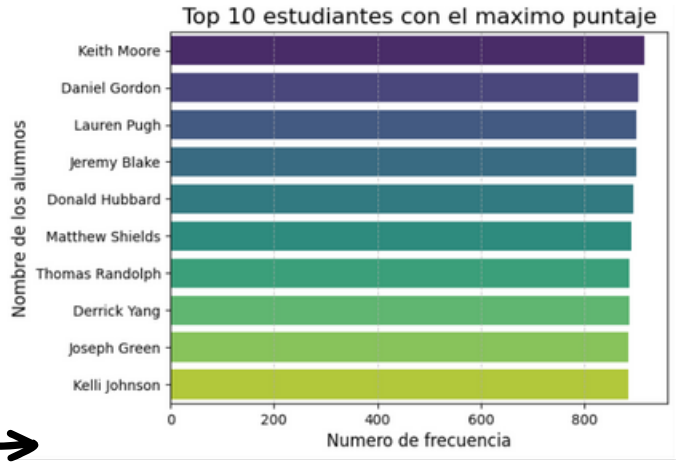
```
2081        Keith Moore
1769       Daniel Gordon
14372       Lauren Pugh
15813      Jeremy Blake
19326     Donald Hubbard
12525    Matthew Shields
11955    Thomas Randolph
11040      Derrick Yang
15903      Joseph Green
3443       Kelli Johnson
Name: Student Name, dtype: object
```

```
sns.barplot(x=Top_10_students.values, y=df['Student Name'][Top_10_students.index] ,
            palette='viridis')


plt.title('Top 10 estudiantes con el maximo puntaje', fontsize=16)
plt.xlabel('Numero de frecuencia', fontsize=12)
plt.ylabel('Nombre de los alumnos', fontsize=12)

plt.grid(True, which='both', axis='x', linestyle='--', linewidth=0.7, alpha=0.7)


plt.show()
```

[25] ✓ 0.2s

```
C:\Users\fatis\AppData\Local\Temp\ipykernel_36420\3154232.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y`

  sns.barplot(x=Top_10_students.values, y=df['Student Name'][Top_10_students.index] ,
```

Por ultimo mostraremos a los top 10 estudiantes con mejores notas, creando un grafico con las caracteristicas:

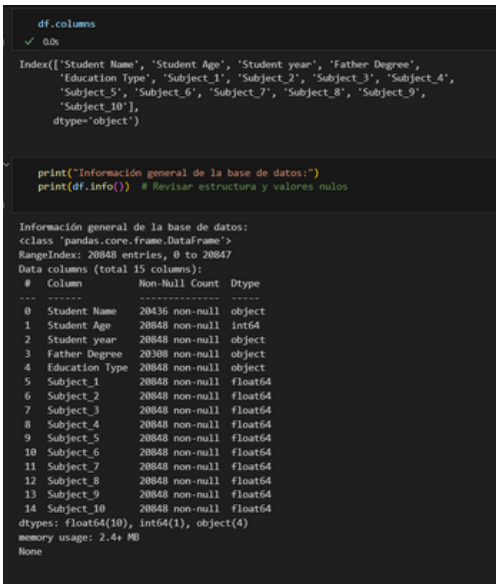


Siendo Keith Moore con el mejor puntaje de todos.

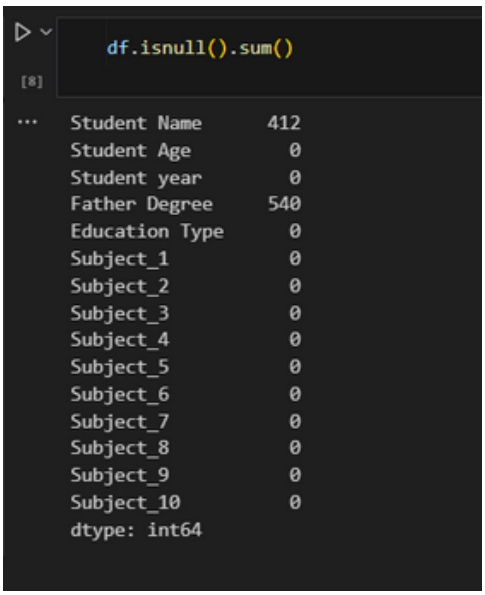
PREDICCION DEL RENDIMIENTO ACADEMICO

Después del análisis de datos, pude finalmente seguir al siguiente paso, entrenar a un modelo que hiciera la predicción del rendimiento estudiantil.

- Descargamos las librerías y cargamos la base de datos



- Revisamos las estructuras y valores nulos



- Volvimos a encontrar valores nulos en 'Student Name' y 'Father Degree', eso quiere decir que no se hizo correctamente la limpieza en un principio

- Hacemos la eliminación de los valores nulos.

```
df.dropna(inplace=True)
df
```

```
df.dropna(subset=[f'Subject_{i}' for i in range(1, 11)], inplace=True)
df
```

	Student Name	Student Age	Student year	Father Degree	Education Type	Subject_1	Subject_2	Subject_3	Subject_4	Subject_5	Subje
0	Jaclyn Mcneil	14	Year 9	Bachelor	IB	91.603078	73.186427	64.240239	47.786542	83.138581	66.91
1	Eric Steele	17	Year 11	PhD	IGCSE	74.906144	69.518146	65.284841	56.317028	69.758140	85.81
2	Carrie Harvey	17	Year 9	PhD	IGCSE	52.064120	75.201754	80.705790	67.390144	78.045191	69.35
3	Jon Williamson	17	Year 12	High School	IB	86.643070	76.595939	91.418080	100.000000	86.788439	85.08
4	Ashley Rogers	17	Year 12	High School	IGCSE	77.498663	85.989126	67.744520	74.602568	70.193265	66.20
...	...	...	...	...	...	...	...	...	...	...	...
20843	Karen Adams	15	Year 12	High School	Thanweya	93.959985	95.687938	66.221163	85.957726	69.853142	49.66
20844	Ashley Jones	15	Year 12	High School	IGCSE	72.238773	84.319553	65.463763	74.027376	100.000000	54.57
20845	Jeanette Hampton	14	Year 9	PhD	Thanweya	41.352071	71.002146	73.500337	66.110988	78.279975	76.20
20846	Madeline Craig	17	Year 10	Master	IB	66.034573	62.075037	64.595721	78.087032	37.271821	100.00
20847	Joshua Castillo	16	Year 12	PhD	IGCSE	69.962353	100.000000	75.506957	99.733819	100.000000	59.15

19905 rows × 15 columns

- Verificamos que ya no haya mas valores nulos

```
df.isnull().sum()
```

```
Student Name      0
Student Age       0
Student year      0
Father Degree     0
Education Type    0
Subject_1         0
Subject_2         0
Subject_3         0
Subject_4         0
Subject_5         0
Subject_6         0
Subject_7         0
Subject_8         0
Subject_9         0
Subject_10        0
dtype: int64
```

- Para poder hacer la predicción del rendimiento estudiantil en egipto, cree una nueva columna llamada "Average_Grade" o "Promedio" esto para poder categorizar a los estudiantes, con rendimiento, alto, medio y bajo, a la hora de hacer la predicción.

```
df['Average_Grade'] = df[[f'Subject_{i}' for i in range(1, 11)]].mean(axis=1)
df
```

	Student Name	Student Age	Student year	Father Degree	Education Type	Subject_1	Subject_2	Subject_3	Subject_4	Subject_5	Subject_6
0	Jaclyn Mcneil	14	Year 9	Bachelor	IB	91.603078	73.186427	64.240239	47.786542	83.138581	66.913702
1	Eric Steele	17	Year 11	PhD	IGCSE	74.906144	69.518146	65.284841	56.317028	69.758140	85.810641
2	Carrie Harvey	17	Year 9	PhD	IGCSE	52.064120	75.201754	80.705790	67.390144	78.045191	69.355050
3	Jon Williamson	17	Year 12	High School	IB	86.643070	76.595939	91.418080	100.000000	86.788439	85.086126
4	Ashley Rogers	17	Year 12	High School	IGCSE	77.498663	85.989126	67.744520	74.602568	70.193265	66.204522
...	...	...	...	...	...	...	...	...	...	...	...
20843	Karen Adams	15	Year 12	High School	Thanweya	93.959985	95.687938	66.221163	85.957726	69.853142	49.664365
20844	Ashley Jones	15	Year 12	High School	IGCSE	72.238773	84.319553	65.463763	74.027376	100.000000	54.575907
20845	Jeanette Hampton	14	Year 9	PhD	Thanweya	41.352071	71.002146	73.500337	66.110988	78.279975	76.200527
20846	Madeline Craig	17	Year 10	Master	IB	66.034573	62.075037	64.595721	78.087032	37.271821	100.000000
20847	Joshua Castillo	16	Year 12	PhD	IGCSE	69.962353	100.000000	75.506957	99.733819	100.000000	59.151008

19905 rows × 16 columns

- Guarde la base de datos para poder trabajar con el modelo de predicción

```
df.to_csv('basep_limpia2.csv', index=False)
```

Con ayuda de inteligencia artificial, pude realizar este modelo de regresión lineal, proporcionándome los siguientes datos:

Prediccion del rendimiento academico

+ Code + Markdown

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
import numpy as np
```

✓ 1.8s

```
df=pd.read_csv('https://raw.githubusercontent.com/fluvwonie/World-/refs/heads/main/basep_limpia2.csv')
df
```

✓ 1.9s

- Importamos las librerías y cargamos la base de datos anteriormente guardada.

```
# Preparación inicial: Separar las características y la variable objetivo
target = 'Average_Grade'
categorical_columns = ['Student year', 'Father Degree', 'Education Type']
```

✓ 0.0s

```
# Codificar variables categóricas
label_encoders = {col: LabelEncoder() for col in categorical_columns}
for col in categorical_columns:
    df[col] = label_encoders[col].fit_transform(df[col])
```

✓ 0.0s

```
# Separar las variables predictoras (X) y la objetivo (y)
X = df.drop(columns=['Student Name', target])  # Excluir el nombre del estudiante y la variable objetivo
y = df[target]
```

✓ 0.0s

- Para poder entrenar al modelo, primero se separan las caracteristicas y la variable objetivo (“Average_Grade”).
- Codificamos las variables categóricas.
- Lo siguiente es separar las variables predictoras y las del objetivo (X, Y).

```
# Dividir los datos en entrenamiento (80%) y prueba (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

X_train.head(), y_train.head()
```

✓ 0.0s

```
(       Student Age  Student year  Father Degree  Education Type  Subject_1  \
11387           14             1              2               2  65.010854
13611           14             2              0               0  81.130965
9639            18             1              2               2  59.831672
6389            16             3              1               2  60.767487
664             17             0              1               2  82.272240

       Subject_2  Subject_3   Subject_4  Subject_5  Subject_6  Subject_7  \
11387  89.868695  84.663048  100.000000  93.666098  94.538260  93.956288
13611  57.592092  56.788232   61.989218  64.700351  67.848071  75.868380
9639   73.631310  61.442064   72.734982  78.267454  64.954200  74.248515
6389   52.263592  67.828425   80.727926  69.184381  80.551642  77.048687
664    86.785630  73.097587   83.944685  79.258456  78.669466  83.796634

       Subject_8  Subject_9  Subject_10
11387  68.067512  66.857177   75.490723
13611  70.489821  95.790699   87.494940
9639   65.125343  86.230789   75.690847
6389   47.587374  74.755987   78.466621
664    55.563034  80.865122   64.708996  ,
11387   83.211866
13611   71.969277
9639    71.215718
6389    68.918212
664     76.896185
Name: Average_Grade, dtype: float64)
```

- Se hace una división con los datos de prueba y con los datos de entrenamiento

```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Entrenar un modelo de regresión lineal
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)

# Predicción en el conjunto de prueba
y_pred = linear_model.predict(X_test)

# Evaluación del modelo
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

mse, r2
```

✓ 0.0s

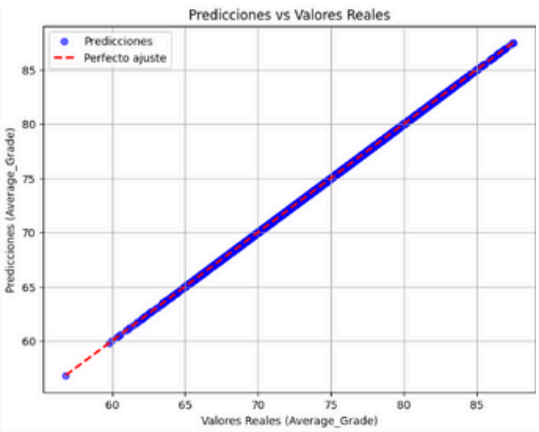
```
(np.float64(1.510935153949763e-28), 1.0)
```

- Se hace el entrenamiento del modelo de regresión lineal
- hace la predicción en el conjunto de prueba
- Se evalúa el modelo.

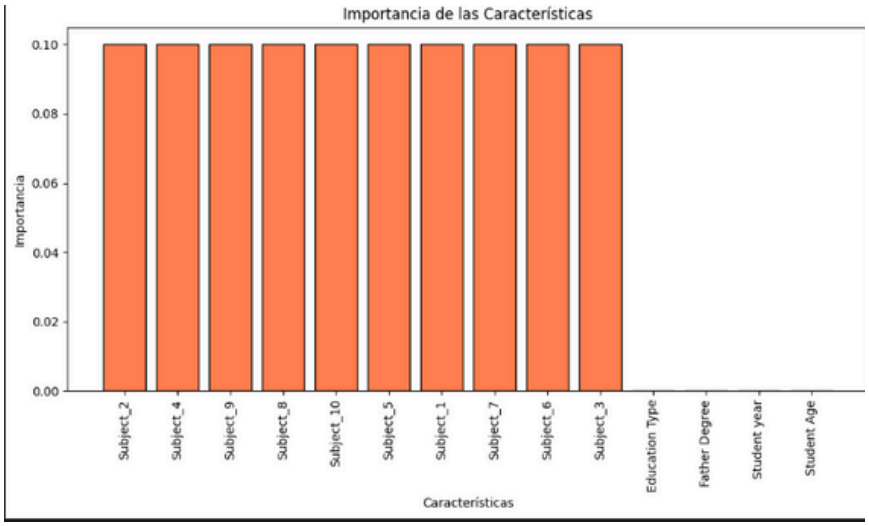
```python
import matplotlib.pyplot as plt
import numpy as np

# 1. Gráfico: Predicciones vs Valores Reales
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.6, color='blue', label='Predicciones')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2, label='Perfecto ajuste')
plt.title('Predicciones vs Valores Reales')
plt.xlabel('Valores Reales (Average_Grade)')
plt.ylabel('Predicciones (Average_Grade)')
plt.legend()
plt.grid(True)
plt.show()

# 2. Gráfico: Importancia de las características (para Random Forest o modelos similares)
if hasattr(linear_model):
    importances = linear_model.feature_importances_
    feature_names = X.columns
else:
    importances = np.abs(linear_model.coef_)  # Para modelos lineales
    feature_names = X.columns

# Ordenar características por importancia
indices = np.argsort(importances)[::-1]
plt.figure(figsize=(10, 6))
plt.bar(range(X.shape[1]), importances[indices], align='center', color='coral', edgecolor='black')
plt.xticks(range(X.shape[1]), feature_names[indices], rotation=90)
plt.title('Importancia de las Características')
plt.xlabel('Características')
plt.ylabel('Importancia')
plt.tight_layout()
plt.show()
```

✓ 0.4s

- Creo un grafico de las predicciones y los valores reales.
- Realizo otro grafico para ordenar las características y su importancia



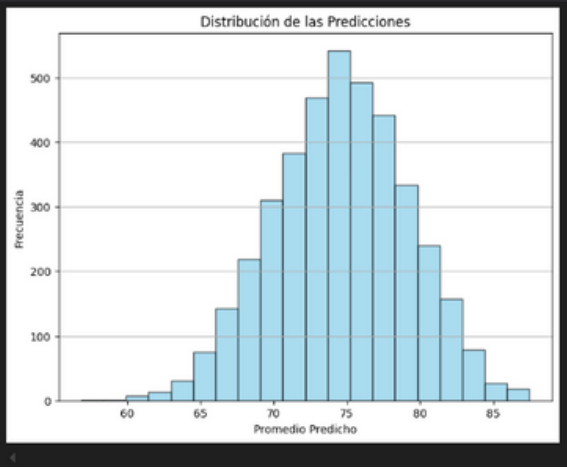
El modelo tiene un buen desempeño, ya que las predicciones son muy cercanas a los valores reales, lo cual sugiere un ajuste casi perfecto.



El modelo considera todas las características de manera equitativa, lo que implica que el rendimiento del estudiante (Average_Grade) depende de una combinación equilibrada de factores.

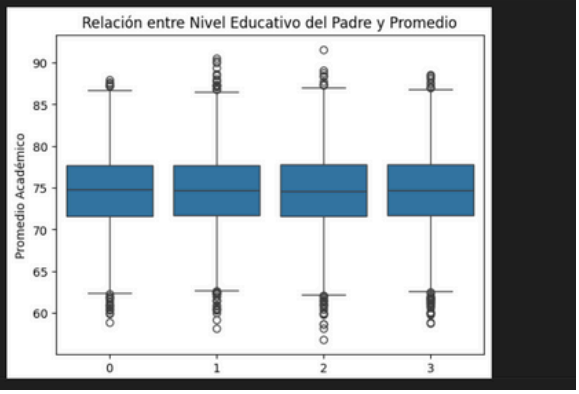
```
plt.figure(figsize=(8, 6))
plt.hist(y_pred, bins=20, color='skyblue', edgecolor='black', alpha=0.7)
plt.title('Distribución de las Predicciones')
plt.xlabel('Promedio Predicho')
plt.ylabel('Frecuencia')
plt.grid(axis='y')
plt.show()
```

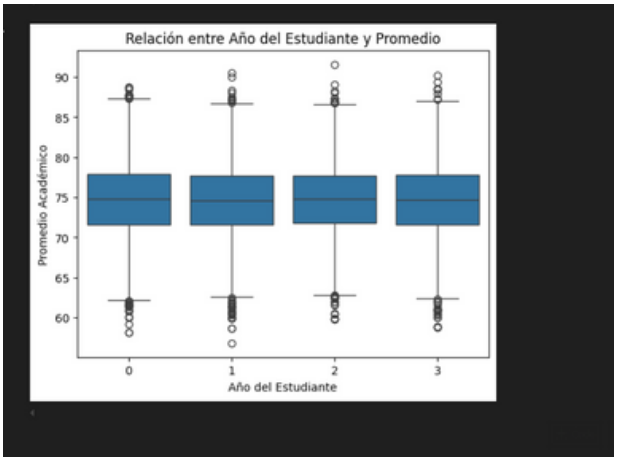
✓ 0.1s



- El grafico que muestra la distribución de las predicciones, vemos que el promedio tiene un pico mas alto en 75.

```
import seaborn as sns

# Comparar el nivel educativo del padre con el rendimiento
sns.boxplot(x='Father Degree', y='Average_Grade', data=df)
plt.title('Relación entre Nivel Educativo del Padre y Promedio')
plt.xlabel('Nivel Educativo del Padre')
plt.ylabel('Promedio Académico')
plt.show()

# Comparar rendimiento por año del estudiante
sns.boxplot(x='Student year', y='Average_Grade', data=df)
plt.title('Relación entre Año del Estudiante y Promedio')
plt.xlabel('Año del Estudiante')
plt.ylabel('Promedio Académico')
plt.show()
```





Creación del DashBoard

Creacion del Dashboard

```
import dash    Import "dash" could not be resolved
from dash import dcc, html    Import "dash" could not be resolved
import plotly.express as px    Import "plotly.express" could not be resolved
import pandas as pd    Import "pandas" could not be resolved from source
```

✓ 0.0s

```
data=pd.read_csv('https://raw.githubusercontent.com/fluvwonie/World-/refs/heads/main/prediccion_rendimiento.csv')
```

```
# Definir el orden de las categorías
data['Prediccion_Rendimiento'] = pd.Categorical(
    data['Prediccion_Rendimiento'],
    categories=['Rendimiento bajo', 'Rendimiento medio', 'Alto rendimiento'],
    ordered=True)
```

✓ 0.0s

```
rendimiento_padres = data.groupby(['Father Degree', 'Prediccion_Rendimiento']).size().unstack(fill_value=0)
```

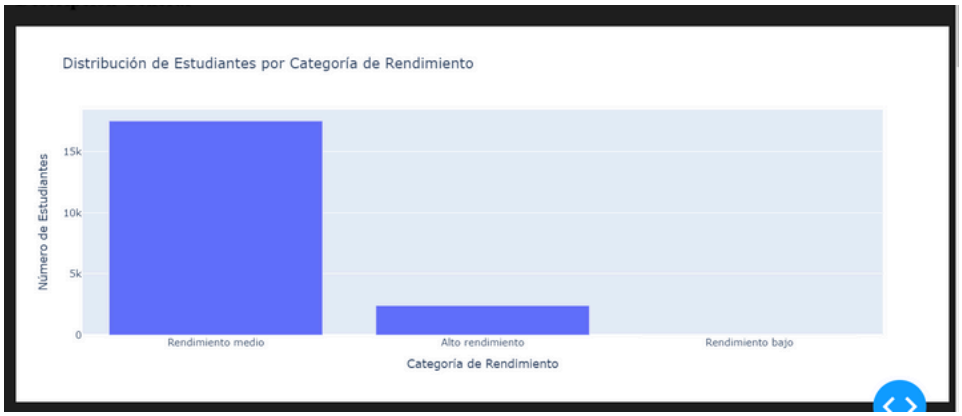
```
# Gráfico 1: Distribución de estudiantes por categoría de rendimiento
rendimiento_counts = data['Prediccion_Rendimiento'].value_counts()
fig_rendimiento = px.bar(
    rendimiento_counts.reindex(['Rendimiento bajo', 'Rendimiento medio', 'Alto rendimiento'], fill_value=0),
    x=rendimiento_counts.index,
    y=rendimiento_counts.values,
    labels={'x': 'Categoría de Rendimiento', 'y': 'Número de Estudiantes'},
    title='Distribución de Estudiantes por Categoría de Rendimiento',
)


# Gráfico 2: Boxplots de calificaciones por materia
fig_boxplot = px.box(
    data,
    y=data.columns[5:15],
    title='Distribución de Calificaciones por Materia',
    labels={'variable': 'Materias', 'value': 'Calificación'}
)

# Gráfico 3: Histograma del promedio general
fig_histograma = px.histogram(
    data,
    x='Average_Grade',
    nbins=20,
    title='Distribución del Promedio General',
    labels={'Average_Grade': 'Promedio General'},
)

# Gráfico 4: Relación entre nivel educativo de los padres y rendimiento
rendimiento_padres = data.groupby(['Father Degree', 'Prediccion_Rendimiento']).size().unstack(fill_value=0)

fig_rendimiento_padres = px.bar(
    rendimiento_padres,
    barmode='stack',
    title='Relación entre Nivel Educativo de los Padres y Rendimiento',
    labels={'value': 'Número de Estudiantes', 'Father Degree': 'Nivel Educativo de los Padres'},
)
```

```python
# Gráfico 5: Relación entre edad del estudiante y promedio general
fig_dispersion = px.scatter(
    data,
    x='Student Age',
    y='Average_Grade',
    title='Relación entre Edad del Estudiante y Promedio General',
    labels={'Student Age': 'Edad del Estudiante', 'Average_Grade': 'Promedio General'},
)

# Crear la app Dash
app = dash.Dash(__name__)

# Diseño del dashboard
app.layout = html.Div([
    html.H1('Dashboard Informativo de Rendimiento Estudiantil', style={'text-align': 'center'}),

    html.Div([
        html.H2('Descripción General'),
        dcc.Graph(figure=fig_rendimiento),
    ]),

    html.Div([
        html.H2('Análisis de Calificaciones'),
        dcc.Graph(figure=fig_boxplot),
        dcc.Graph(figure=fig_histograma),
    ]),

    html.Div([
        html.H2('Factores Relacionados con el Rendimiento'),
        dcc.Graph(figure=fig_rendimiento_padres),
        dcc.Graph(figure=fig_dispersion),
    ]),
])

# Ejecutar el servidor
if __name__ == '__main__':
    app.run_server(debug=True)
```

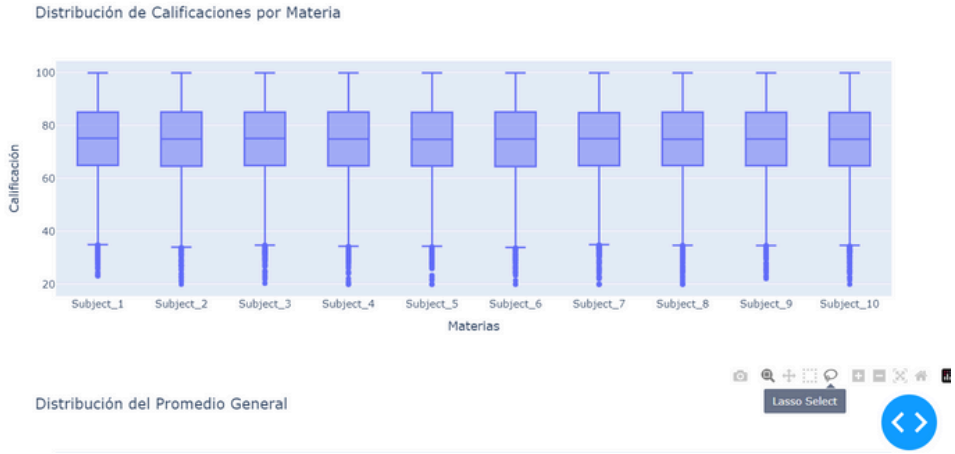
✓ 0.4s

Este código crea un dashboard interactivo que visualiza:

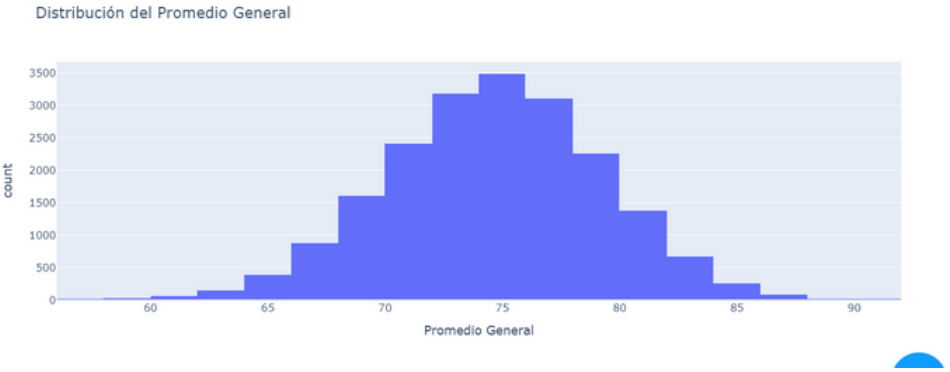
- La distribución de estudiantes según su rendimiento.
- Las calificaciones por materia mediante boxplots.
- La distribución del promedio general con un histograma.
- La relación entre el nivel educativo de los padres y el rendimiento de los estudiantes.



Distribución de estudiantes por categoría de rendimiento:
Observamos que la mayoría se encuentra en un rendimiento medio, caso contrario al rendimiento alto y bajo.



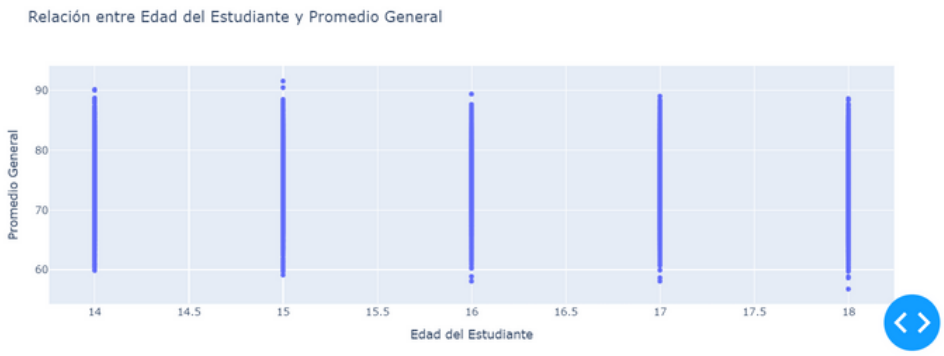
- Las cajas de cada materia son bastante similares en tamaño, lo que indica que la dispersión de las calificaciones es relativamente uniforme entre las diferentes materias.
- La mediana de las calificaciones parece estar alrededor de 80 en la mayoría de las materias, lo que sugiere un buen rendimiento promedio.
- Hay varios puntos debajo, indicando que existen estudiantes con calificaciones significativamente más bajas en cada materia.
- La mayoría de las calificaciones están entre 60 y 100, lo que sugiere que hay pocos estudiantes con calificaciones extremadamente bajas.



Este histograma sugiere que la mayoría de los estudiantes tienen un rendimiento promedio bueno (alrededor de 75), con pocos estudiantes en los extremos. La distribución es bastante centrada, lo que indica que no hay grandes desviaciones hacia promedios muy bajos o muy altos.

Esta gráfica sugiere que el nivel educativo de los padres podría no ser un factor determinante en el rendimiento académico de los estudiantes, al menos en los niveles medio y alto, ya que la distribución es similar en todas las categorías.

La ausencia de barras azules sugiere que muy pocos estudiantes se encuentran en la categoría de rendimiento bajo, lo que podría indicar un enfoque exitoso en mantener los niveles de rendimiento por encima de lo básico.



Cada punto azul en la gráfica representa un estudiante individual, con su respectiva edad y promedio general.

Parece haber una dispersión significativa en los promedios para cada grupo de edad, lo que significa que dentro de cada edad hay estudiantes con promedios muy altos (cerca de 90) y promedios más bajos (cerca de 60).
No parece haber un cambio evidente en los promedios generales entre diferentes edades, ya que las distribuciones verticales parecen similares.

Uso y beneficio

Este dashboard es una herramienta útil para maestros, o docentes en la toma de decisiones informadas sobre el desempeño estudiantil.

C O N C L U S I O N E S

Dentro de la base no se presentan como tal los factores que pudieran estar afectando al rendimento de los estudiantes, un factor que pudiera influir es el rendimiento academico de los padres, aun que como ya se mostro, podria no ser un factor determinate.

Observamos que la mayoría de los estudiantes están en un rendimiento académico bastante bueno, lo que podría indicar que la calidad de educación en Egipto podría ser buena.

M e j o r a s

- Ampliar el dataset con más variables, como el tiempo de estudio de los estudiantes, para poder determinar si eso podría influir en sus calificaciones.
- Probar modelos avanzados como Arboles de decisión.
- Incluir funcionalidades adicionales en el dashboard.

A n e x o s

https://www.kaggle.com/datasets/mohamedalabasy/education-in-egyp
https://www.kaggle.com/code/fatmamuhsen/egypt-education-dataanalysist
https://raw.githubusercontent.com/fluvwonie/World-/refs/heads/main/basep_limpia2.csv
https://raw.githubusercontent.com/fluvwonie/World-/refs/heads/main/base_de_datos_limpia.csv
https://raw.githubusercontent.com/fluvwonie/World-/refs/heads/main/prediccion_rendimiento.csv
Librerias usadas: pandas, dash, matplotlib, scikit-learn

L i n k d e l v i d e o d e l p r o y e c t o