

- Ximena Silva Bárcena A01785518
- Rodrigo Martínez Vallejo A00573055

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
data = pd.read_csv('data.csv')
```

```
numeric_data = data.select_dtypes(include='number')
```

✓ Data Description

In this section I review the type of data each columns holds, as long as what it represents and the upper and lower limits of those columns.

```
print(data.dtypes)
```

In total there are 15 variables and a total number of 1000 rows

- **student_id** *object* it is the unique identifier of each student
- **age** *int64* Age of student. Goes from 17 to 24
- **gender** *object* Male/Female/Other.
- **study_hours_per_day** *float64* Avg. daily study time. Goes from 0 to 8.3
- **social_media_hours** *float* Daily social media time. Goes from 0 to 7.2
- **netflix_hours** *float64* Avg. daily Netflix/binging time. Goes from 0 to 5.4
- **part_time-job** Yes/No.
- **attendance_percentage** *float64* Class attendace (0-100%).
- **sleep_hours** *float64* Avg. daily sleep
- **diet_quality** *object* Poor/Fair/Good. Goes form 3.2 to 10
- **exercise_frequency** *int64* Times per week. Goes from 0 to 7.
- **parental_education_level** *object* HighSchool/Bachellor/Other
- **internet_quality** *object* Good/Average/Other
- **mental_health_rating** *int64* Scale of 1 to 10

- **extracurricular_participation** *object* Yes/No
- **exam_scores** *float64* Final exam score (0-100)

✓ Ignored Variables

- Student_id: is a unique identifier and does not contribute to understanding the relationships between study habits and academic success.
- Gender: is a categorical variable.
- Sleep-hour: show low correlation with the exam_score and other key performance indicators.
- Diet_quality: is a categorical variable.
- Extracurricular_participation: low correlation with the exam_score.
- Internet_quality: low correlation with exam_score.
- Parental_education_level: low correlation with exam_score.

✓ Smart Objectives

1. Rodrigo Martínez:

Using the current dataset, determine whether study_hours_per_day has a stronger correlation and is the only relevant variable to predict exam scores than other variables. This will be measured by recreating the model only using this variable to evaluate the predictive power of study time on academic performance.

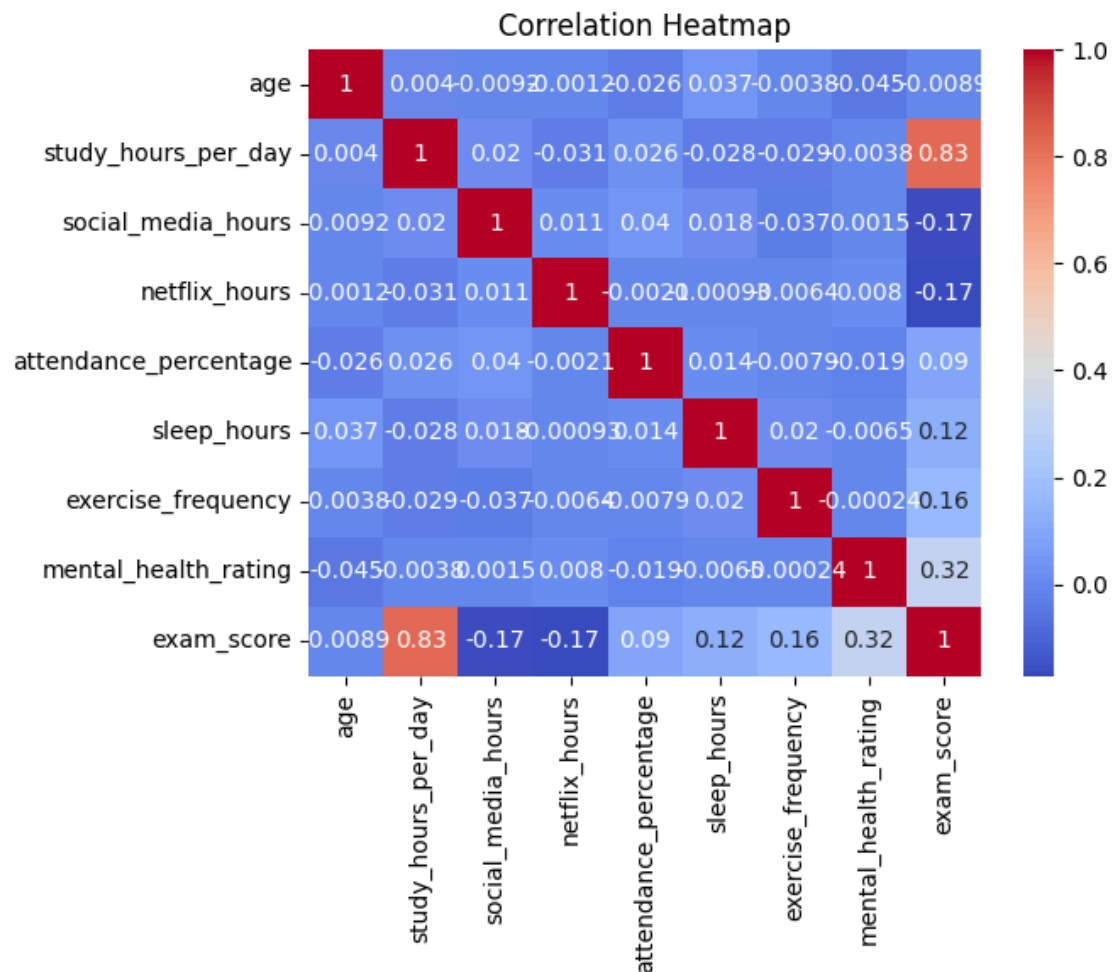
2. Ximena Silva:

To improve the predictive accuracy of academic performance models for students by focusing on key mental health-related variables, including mental health rating, sleep hours, exercise frequency, diet quality, and screen time (social media and Netflix hours), while eliminating less relevant factors like age, gender, parental education level, internet quality, part-time job, and extracurricular participation. This approach aims to achieve at least a 20% improvement in model accuracy by reducing unnecessary noise, enhancing the clarity and relevance of the predictive features.

✓ Heatmap

```
sns.heatmap(num_data_corr, annot=True, cmap='coolwarm')
```

```
plt.title("Correlation Heatmap")
plt.show()
```



✓ K-Means with chosen K value

```
# Matriz de entrada
X = np.array(dataframe_selected)
print("Shape:", X.shape)
```

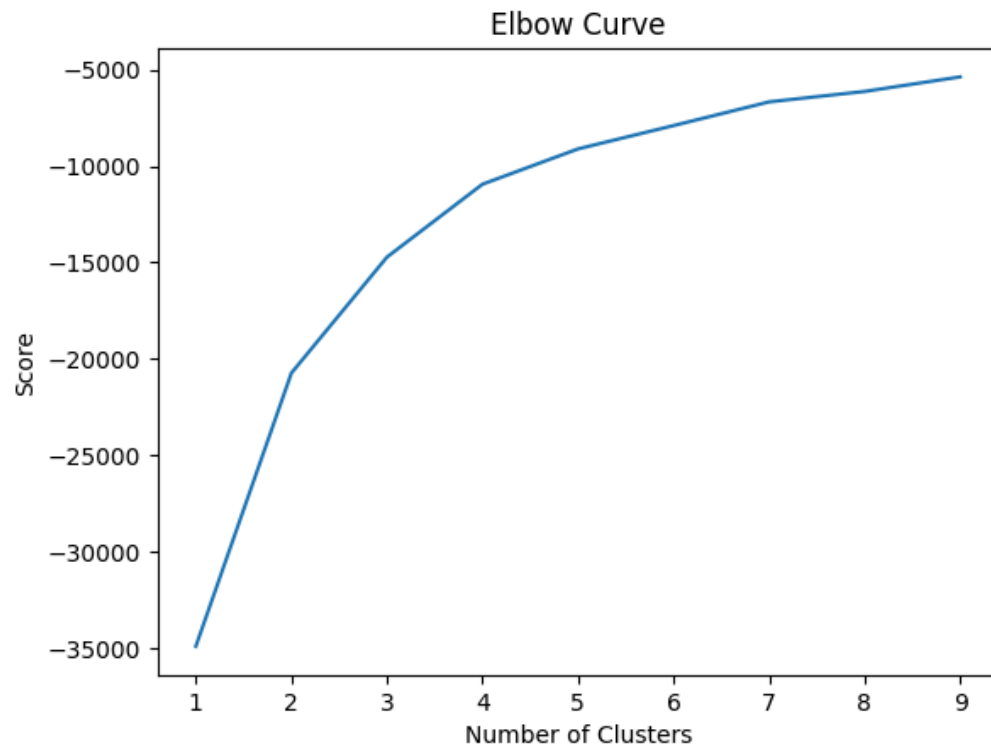
```
# Elbow curve
Nc = range(1, 10)
```

```

kmeans_models = [KMeans(n_clusters=i, n_init=10) for i in Nc]
scores = [model.fit(X).score(X) for model in kmeans_models]
plt.plot(Nc, scores)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()

```

↔ Shape: (100, 9)



Haz doble clic (o pulsa Intro) para editar

- **Value for k: 3**

The value of K was deceived upon the review of the Elbow Curve, from which it was decided to stick with a value of 3.

```

# Elegir K = 4
kmeans = KMeans(n_clusters=3, n_init=10).fit(dataframe_selected)
labels = kmeans.predict(dataframe_selected)
centroids = kmeans.cluster_centers_

```

```
# Asignar colores
colores = ['red','green', 'blue']
asignar = [colores[i] for i in labels]

for i in range(len(selected_columns)):
    for j in range(len(selected_columns)):
        if i != j:
            plt.scatter(X[:, i], X[:, j], c=asignar, s=70)
            plt.scatter(centroids[:, i], centroids[:, j], marker="*", c=colores, s=1000)
            plt.xlabel(selected_columns[i])
            plt.ylabel(selected_columns[j])
            plt.show()
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

```
df = pd.read_csv('student_habits_performance.csv')
```

```
df.info()
```

```
>>> <class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype
---  -
 0   student_id            1000 non-null   object
 1   age                   1000 non-null   int64
 2   gender                1000 non-null   object
 3   study_hours_per_day   1000 non-null   float64
 4   social_media_hours    1000 non-null   float64
 5   netflix_hours         1000 non-null   float64
 6   part_time_job         1000 non-null   object
 7   attendance_percentage 1000 non-null   float64
 8   sleep_hours           1000 non-null   float64
 9   diet_quality          1000 non-null   object
10   exercise_frequency    1000 non-null   int64
11   parental_education_level 909 non-null    object
12   internet_quality      1000 non-null   object
13   mental_health_rating  1000 non-null   int64
```

```
14 extracurricular_participation 1000 non-null object
15 exam_score                    1000 non-null float64
dtypes: float64(6), int64(3), object(7)
memory usage: 125.1+ KB
```

```
selected_columns = ["age", "study_hours_per_day", "social_media_hours", "netflix_hours", "attendance_percentage", "sleep_hours", "exercise_frequen
```

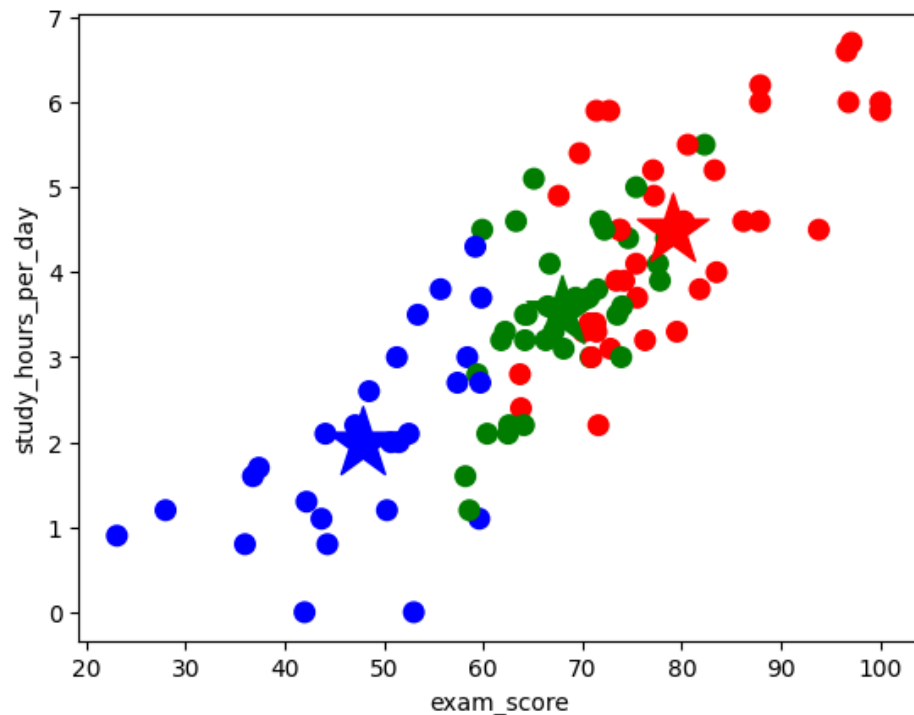
```
#threshold = df['study_hours_per_day'].quantile(0.75)
#filtered_df = df[df['study_hours_per_day'] >= threshold]
#dataframe_selected = filtered_df[selected_columns]
```

```
filtered_df = df.sample(frac=0.1, random_state=42)
dataframe_selected = filtered_df[selected_columns]
```

✓ Centroids for K-Means

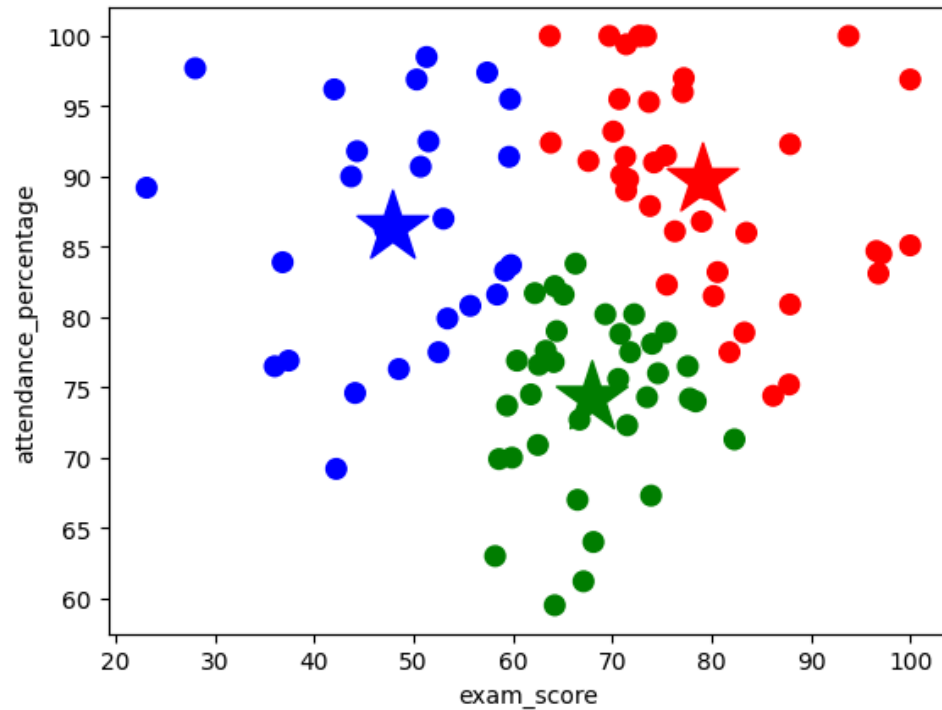
Exam scores vs Study hours per day

This scatter plot reveals a strong positive correlation between the number of hours students dedicate to studying and their exam performance. The red cluster is concentrated around 4.5 study hours per day and plus 80 in the exam scores, suggesting that increased study time pays off significantly. The green cluster falls around 3.5 hours of study and 65 as result, while the blue cluster shows the least study time at 2 hours of study and scores around 45 and 50.



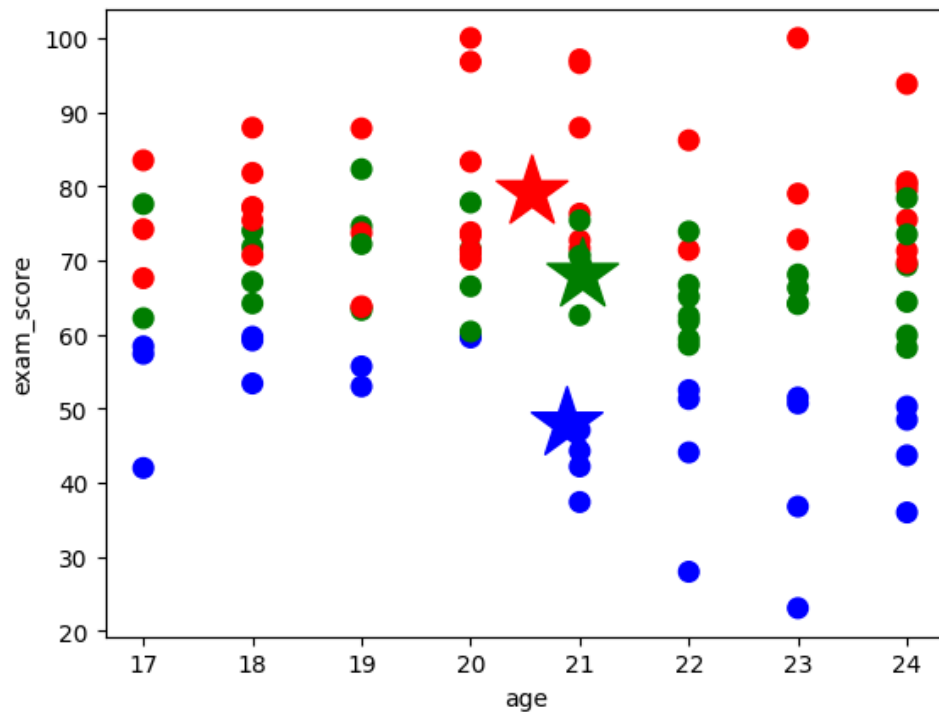
Study hours per day vs Attendance percentage

The clusters of this second graph correlate perfectly to the information obtained from the first one, where the red cluster corresponds to both the ones who studied the most and the ones who attended the majority of the classes. This revelation reinforces the idea that discipline corresponds to higher scores. On the other hand, the green cluster balances moderate attendance with moderate study time, and the blue cluster, despite decent attendance, studies far less, showing that attendance alone isn't enough without active study time.



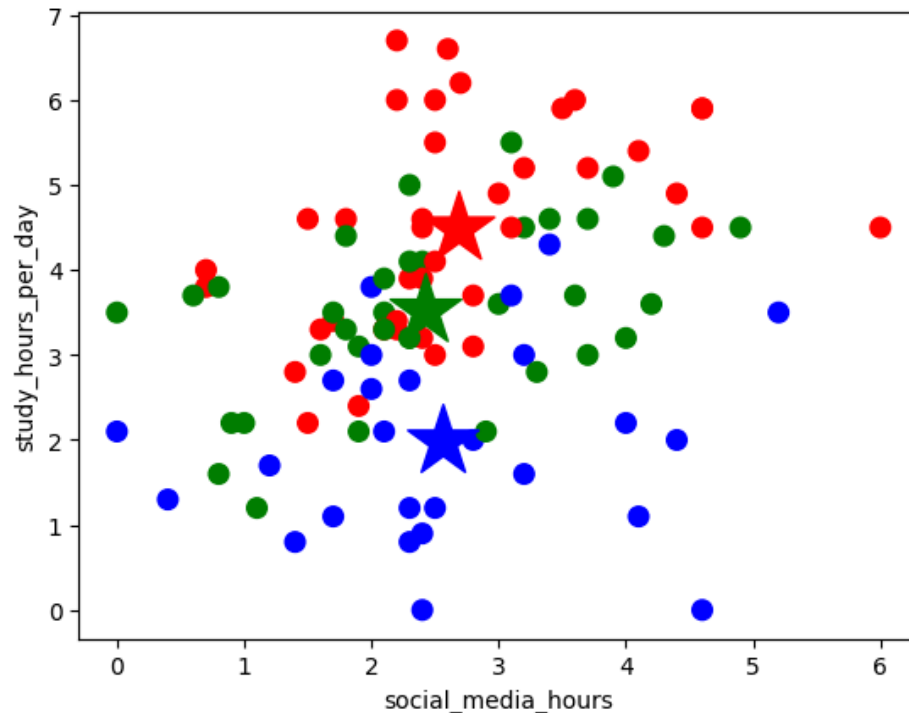
Exam scores vs Age

The centroids in this graph reveal that age is not a factor to take into account regarding academic performance, as the age gap is distributed pretty much equally across the three groups.



Social media hours vs Study hours per day

In this chart it can be observed that the blue cluster spends a similar amount of time on social media despite studying less. The red cluster balances study and screen time more effectively, clustering around 4.5 study hours of study with pretty much the same social media time as the other two clusters. From this it could be said that social media hours of use don't appear to play a significant role in this case regarding the ability to spend the same time studying or more.



- **Cluster 1:** These students show high academic performance with consistently high study hours, high attendance, and moderate screen time. They likely have strong time management and learning strategies.
- **Cluster 2:** This group maintains average study time, moderate attendance, and average exam performance. They may benefit from improved study habits or targeted academic support.
- **Cluster 3:** Students in this cluster attend class fairly regularly but study significantly less and perform poorly. Their screen time is pretty much the same as the other clusters.

✓ Data Mining

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
from wordcloud import WordCloud
from collections import Counter
```

```
df = pd.read_csv('student_habits_performance.csv')

text_columns = ['gender', 'part_time_job', 'diet_quality', 'parental_education_level',
                'internet_quality', 'extracurricular_participation']

for col in text_columns:
    df[col] = df[col].astype(str).apply(lambda x: re.sub(r'^a-zA-Z ', '', x.lower().strip()))

# Distribution plots
numeric_cols = ['study_hours_per_day', 'social_media_hours', 'netflix_hours',
                'attendance_percentage', 'sleep_hours', 'exercise_frequency',
                'mental_health_rating', 'exam_score']

# Word clouds from categorical text
for col in text_columns:
    text = " ".join(df[col])
    wordcloud = WordCloud(background_color="white").generate(text)
    plt.figure(figsize=(6, 4))
    plt.imshow(wordcloud, interpolation="bilinear")
    plt.axis("off")
    plt.title(f"Word Cloud: {col}")
    plt.show()
```



Word Cloud: gender

male
female

Word Cloud: part_time_job

yes

Word Cloud: diet_quality

poor good
fair

Word Cloud: parental_education_level

nan bachelor high
school bachelor
high school
master bachelor

Word Cloud: internet_quality

good poor
average

Word Cloud: extracurricular_participation

yes

✓ Prediction

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('student_habits_performance.csv')
```

```
df.head()
```

↗

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part_time_job	attendance_percentage	sleep_hours	diet_quality
0	S1000	23	Female	0.0	1.2	1.1	No	85.0	8.0	
1	S1001	20	Female	6.9	2.8	2.3	No	97.3	4.6	
2	S1002	21	Male	1.4	3.1	1.3	No	94.8	8.0	
3	S1003	23	Female	1.0	3.9	1.0	No	71.0	9.2	
4	S1004	19	Female	5.0	4.4	0.5	No	90.9	4.9	

```
df["internet_quality"].value_counts()
```

↗

	count
internet_quality	
Good	447
Average	391
Poor	162

```
gender_mapping = {'Male': 1, 'Female': 0}
df['gender'] = df['gender'].map(gender_mapping).fillna(-1).astype(int)
```

```
diet_mapping = {'Poor': 0, 'Fair': 1, 'Good': 2}
```

```

df['diet_quality'] = df['diet_quality'].map(diet_mapping).fillna(-1).astype(int)

parental_mapping = {'High School': 0, 'Bachelor': 1, 'Master': 2}
df['parental_education_level'] = df['parental_education_level'].map(parental_mapping).fillna(-1).astype(int)

job_mapping = {'Yes': 1, 'No': 0}
df['part_time_job'] = df['part_time_job'].map(job_mapping).fillna(-1).astype(int)

extracurricular_mapping = {'Yes': 1, 'No': 0}
df['extracurricular_participation'] = df['extracurricular_participation'].map(extracurricular_mapping).fillna(-1).astype(int)

internet_mapping = {'Good': 2, 'Average': 1, 'Poor': 0}
df['internet_quality'] = df['internet_quality'].map(internet_mapping).fillna(-1).astype(int)

# Save the modified dataset
df.to_csv('student_habits_performance_modified.csv', index=False)

df_m = pd.read_csv('student_habits_performance_modified.csv')

df_m.info()

```

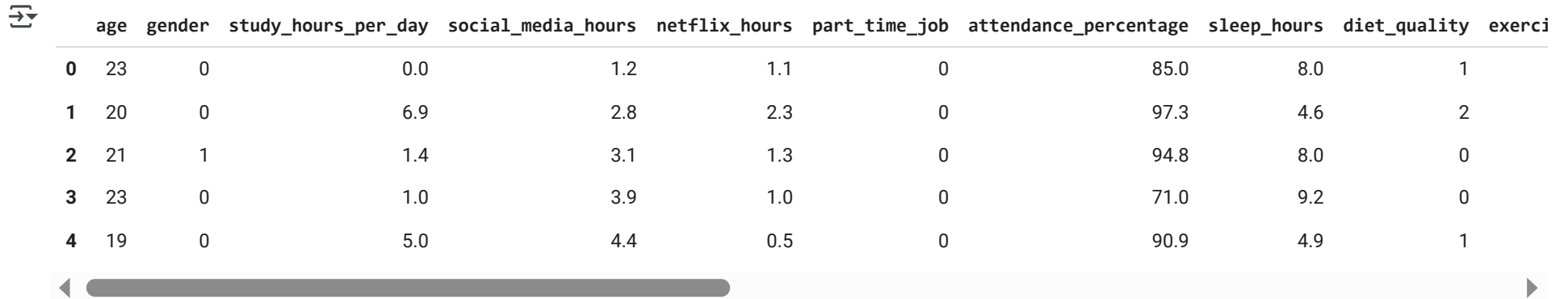
```

↔ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 16 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   student_id                            1000 non-null   object
1   age                                    1000 non-null   int64
2   gender                                1000 non-null   int64
3   study_hours_per_day                   1000 non-null   float64
4   social_media_hours                    1000 non-null   float64
5   netflix_hours                         1000 non-null   float64
6   part_time_job                         1000 non-null   int64
7   attendance_percentage                 1000 non-null   float64
8   sleep_hours                          1000 non-null   float64
9   diet_quality                          1000 non-null   int64
10  exercise_frequency                   1000 non-null   int64
11  parental_education_level              1000 non-null   int64
12  internet_quality                      1000 non-null   int64
13  mental_health_rating                 1000 non-null   int64
14  extracurricular_participation         1000 non-null   int64
15  exam_score                           1000 non-null   float64
dtypes: float64(6), int64(9), object(1)
memory usage: 125.1+ KB

```

```
df = df_m.iloc[:,1:16]
```

```
df.head()
```



	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part_time_job	attendance_percentage	sleep_hours	diet_quality	exerci
0	23	0	0.0	1.2	1.1	0	85.0	8.0	1	
1	20	0	6.9	2.8	2.3	0	97.3	4.6	2	
2	21	1	1.4	3.1	1.3	0	94.8	8.0	0	
3	23	0	1.0	3.9	1.0	0	71.0	9.2	0	
4	19	0	5.0	4.4	0.5	0	90.9	4.9	1	

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




	0
age	0
gender	0
study_hours_per_day	0
social_media_hours	0
netflix_hours	0
part_time_job	0
attendance_percentage	0
sleep_hours	0
diet_quality	0
exercise_frequency	0
parental_education_level	0
internet_quality	0
mental_health_rating	0
extracurricular_participation	0
exam_score	0



df.columns



```
Index(['age', 'gender', 'study_hours_per_day', 'social_media_hours',
       'netflix_hours', 'part_time_job', 'attendance_percentage',
       'sleep_hours', 'diet_quality', 'exercise_frequency',
       'parental_education_level', 'internet_quality', 'mental_health_rating',
       'extracurricular_participation', 'exam_score'],
      dtype='object')

x = df[['age', 'gender', 'study_hours_per_day', 'social_media_hours',
       'netflix_hours', 'part_time_job', 'attendance_percentage',
       'sleep_hours', 'diet_quality', 'exercise_frequency',
       'parental_education_level', 'internet_quality', 'mental_health_rating',
       'extracurricular_participation']].values # variables independientes
y = df['exam_score'].values # variable dependiente
```

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
```

```
y_test
```

```
array([ 70.9,  36.8,  91.3,  88. ,  69.4,  64.5,  71.1,  73.1, 100. ,
        83.6,  73.9,  60.1,  91.3,  52.5,  77.2,  87.2,  43.4,  65.3,
        78.9,  57.2,  51.3,  84.6,  96.3,  67. ,  75.5,  82.8,  75.1,
        50.4,  30.2, 100. ,  85.4,  58.1,  53.9,  31.5,  34.3,  72.8,
        49.1,  94.8,  71.9,  53.3, 100. ,  65.9,  45.8,  71.5,  65.2,
        77.9,  67.3,  96.2,  76.1,  71.4, 100. ,  27.6,  49.1,  70.7,
        93.7,  61. ,  76.2,  63. ,  56.2,  59.4, 100. ,  49.4,  65.4,
        85.1,  60.3,  64.5,  51.5,  47.4,  85.3,  61.7,  76. ,  93.8,
        79. ,  90. ,  58.8,  73.7,  79.6,  33.6,  80.2,  71.5,  65.1,
        68.7,  45.2,  88. ,  71.8,  88.7,  46.2,  74.9,  80.3,  96.6,
        89.3,  66.5,  53.4,  85.9,  99.4,  77.5,  89.8,  60. ,  99.3,
        35.3,  68.4,  84.9,  69.6,  66.7,  80.4,  58.4,  50.4,  78. ,
        31.1, 100. ,  80.9,  70.5,  59.7,  61.7,  85. ,  64.2,  54.1,
        96.8, 100. ,  62.5,  89.4,  73. ,  84.8,  55. ,  61.2,  47.2,
        68.4,  47.1,  71. ,  61.1,  49.9,  50.3,  66.1,  83.3,  59.2,
        83.2,  70.1,  78.9,  65.7,  83.1,  74. ,  43.7,  62.7,  58.4,
        93.9,  73. ,  59.8,  77.8,  68.1,  51.2,  73.6,  70.6,  74.9,
        44.5,  66.7,  63.6,  58.9,  67.9,  65.6,  76. ,  73. ,  59.6,
        92.9,  53.3,  43.5,  63.6,  94.2,  50.2,  70.4,  54.7, 100. ,
        89.7,  46.7,  87.8,  67.8,  96.2,  51.2,  54.6,  65.6,  74. ,
        63.7,  74.9,  98.1,  98.8,  47.6,  44.3,  66.3,  59. ,  57.1,
        78.4,  81.1,  98.4,  58.3, 100. ,  98.8,  64.2,  58.6,  67.5,
        82.6,  64.2])
```

```
from sklearn.linear_model import LinearRegression
model_regression = LinearRegression()
```

```
model_regression.fit(x_train, y_train)
```

```
LinearRegression
```

```
x_labels = ['age', 'gender', 'study_hours_per_day', 'social_media_hours',
            'netflix_hours', 'part_time_job', 'attendance_percentage',
            'sleep_hours', 'diet_quality', 'exercise_frequency',
            'parental_education_level', 'internet_quality', 'mental_health_rating',
```

```
'extracurricular_participation']
c_label = ['Coeficientes']
```

```
coeff_df = pd.DataFrame(model_regression.coef_, x_labels, c_label)
coeff_df
```



	Coeficientes
age	0.026763
gender	-0.067679
study_hours_per_day	9.572787
social_media_hours	-2.642771
netflix_hours	-2.249510
part_time_job	0.200632
attendance_percentage	0.158908
sleep_hours	2.125172
diet_quality	-0.159697
exercise_frequency	1.446809
parental_education_level	0.071022
internet_quality	-0.441883
mental_health_rating	1.984944
extracurricular_participation	0.050241

```
y_pred = model_regression.predict(x_test)
```

```
residuals = pd.DataFrame({'Real': y_test, 'Predicción': y_pred, 'Residual': y_test - y_pred})
residuals = residuals.sample(n = 24)
residuals = residuals.sort_values(by='Real')
residuals
```



	Real	Predicción	Residual
185	44.3	44.925484	-0.625484
131	50.3	51.050991	-0.750991
106	50.4	60.728004	-10.328004
196	58.6	57.606569	0.993431
146	59.8	57.752282	2.047718
142	62.7	61.256671	1.443329
57	63.0	68.218135	-5.218135
65	64.5	68.590468	-4.090468
138	65.7	61.425151	4.274849
91	66.5	71.684932	-5.184932
103	66.7	69.045013	-2.345013
174	67.8	70.640317	-2.840317
100	68.4	75.329892	-6.929892
151	70.6	75.927637	-5.327637
145	73.0	80.638992	-7.638992
7	73.1	72.973752	0.126248
150	73.6	68.427387	5.172613
26	75.1	72.071976	3.028024
14	77.2	77.682350	-0.482350
189	78.4	78.778973	-0.378973
190	81.1	83.858093	-2.758093
3	88.0	87.090649	0.909351
120	89.4	78.420646	10.979354
183	98.8	90.516333	8.283667

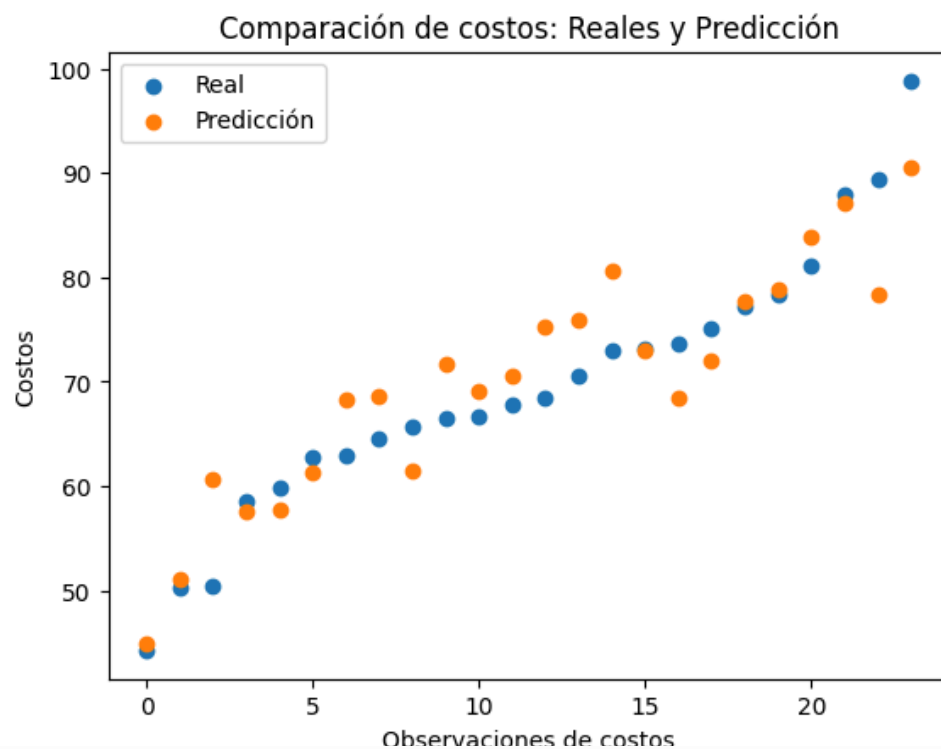
```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

↔ 0.9122682017265076

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

```
%matplotlib inline
```

```
plt.scatter(np.arange(24), residuals['Real'], label = "Real")
plt.scatter(np.arange(24), residuals['Predicción'], label = "Predicción")
plt.title("Comparación de calificaciones reales: Reales y Predicción")
plt.xlabel("Observaciones de calificaciones")
plt.ylabel("calificaciones")
plt.legend(loc='upper left')
plt.show()
```



✓ First Smart Objective

✓ First Smart Objective

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

df = pd.read_csv('student_habits_performance.csv')

x = df[['study_hours_per_day']].values
y = df['exam_score'].values

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

model_regression = LinearRegression()
model_regression.fit(x_train, y_train)

x_labels = ['study_hours_per_day']
c_label = ['Coeficientes']
coeff_df = pd.DataFrame(model_regression.coef_, x_labels, columns=c_label)
print("Coeficientes:")
print(coeff_df)

y_pred = model_regression.predict(x_test)

residuals = pd.DataFrame({
    'Real': y_test,
    'Predicción': y_pred,
    'Residual': y_test - y_pred
})

residuals = residuals.sample(n=24, random_state=0).sort_values(by='Real')
```

```
print("R² Score:", r2_score(y_test, y_pred))

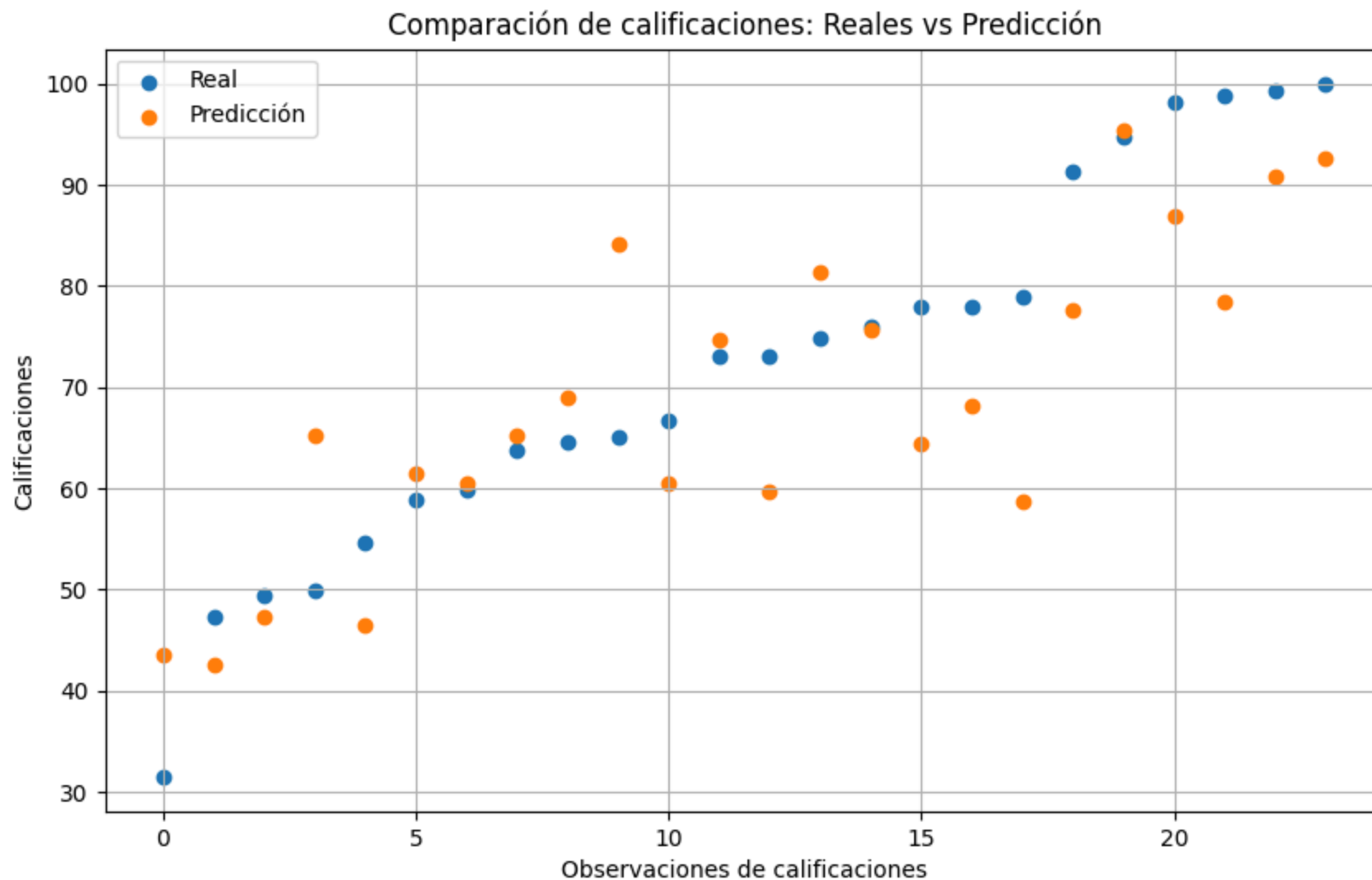
plt.figure(figsize=(10,6))
plt.scatter(np.arange(24), residuals['Real'], label="Real")
plt.scatter(np.arange(24), residuals['Predicción'], label="Predicción")
plt.title("Comparación de calificaciones: Reales vs Predicción")
plt.xlabel("")
plt.ylabel("Calificaciones")
plt.legend(loc='upper left')
plt.grid(True)
plt.show()
```



Coeficientes:

Coeficientes

study_hours_per_day 9.436972

R² Score: 0.7439682335232483

The results obtained shed light on the SMART objective, which aimed to determine whether study hours alone were the most relevant variable for predicting exam scores.

Contrary to initial intuition, the findings reveal that a model based solely on study hours is significantly less accurate than the original model, which considered multiple variables. While the original model achieved an R^2 score of 0.91, the simplified model dropped to 0.74 an 18% decrease in explanatory power. This also reflects a drop in predictive accuracy from 91% to 74%.

The insight taken here coupled with the K-mean analysis done shows clusters where the people who studied the most were the ones who attended most of the classes, creating a correlation more with the compromise the students have with the course in general, and not only attending / not attending or only studying.

In conclusion, although study hours show a strong positive correlation with exam performance, they are not sufficient on their own. Including additional variables is essential for building a more precise and reliable predictive model.

✓ Second Smart Objective

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

df = pd.read_csv('student_habits_performance_modified.csv')

x = df[['social_media_hours',
        'netflix_hours', 'attendance_percentage',
        'sleep_hours', 'diet_quality', 'exercise_frequency',
        'mental_health_rating']].values # variables independientes
y = df['exam_score'].values # variable dependiente

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

model_regression = LinearRegression()
model_regression.fit(x_train, y_train)

x_labels = ['social_media_hours',
```

```

    'netflix_hours', 'attendance_percentage',
    'sleep_hours', 'diet_quality', 'exercise_frequency',
    'mental_health_rating']
c_label = ['Coeficientes']
coeff_df = pd.DataFrame(model_regression.coef_, x_labels, columns=c_label)
print("Coeficientes:")
print(coeff_df)

y_pred = model_regression.predict(x_test)

residuals = pd.DataFrame({
    'Real': y_test,
    'Predicción': y_pred,
    'Residual': y_test - y_pred
})

residuals = residuals.sample(n=24, random_state=0).sort_values(by='Real')

print("R² Score:", r2_score(y_test, y_pred))

plt.figure(figsize=(10,6))
plt.scatter(np.arange(24), residuals['Real'], label="Real")
plt.scatter(np.arange(24), residuals['Predicción'], label="Predicción")
plt.title("Comparación de calificaciones: Reales vs Predicción")
plt.xlabel("")
plt.ylabel("Calificaciones")
plt.legend(loc='upper left')
plt.grid(True)
plt.show()

```



Mostrar salida oculta

The SMART objective provided is flawed because it assumes that reducing the feature set to only mental health-related variables, such as `mental_health_rating`, `sleep_hours`, `exercise_frequency`, `diet_quality`, and screen time (`social_media_hours` and `netflix_hours`), will significantly improve predictive accuracy. However, the regression analysis results indicate the opposite. The focused model, which included only these mental health variables, achieved an R^2 score of just 0.1756, meaning it explains only 17.56% of the

variance in student performance. This is substantially lower than the full model, which included a broader set of factors like age,