Thi Cuối Kỳ - Thực Hành

April 4, 2025

Nguyễn Thái Nguyên - 2274802010587

[156]:
```
import pandas as pd
import numpy as np

# Tạo DataFrame với dữ liệu từ bảng trong hình
data = {
    "Student_ID": ["S001", "S002", "S003", "S004", "S005", "S006", "S007",␣
 ↪"S008", "S009", "S010",
                   "S011", "S012", "S013", "S014", "S015", "S016", "S017",␣
 ↪"S018", "S019", "S020",
                   "S021", "S022", "S023", "S024", "S025", "S026", "S027",␣
 ↪"S028", "S029", "S030"],
    "Course": ["Math", "Physics", "Chemistry", "Math", "Physics", "Chemistry",␣
 ↪"Math", "Physics", "Chemistry", "Math",
               "Physics", "Chemistry", "Math", "Physics", "Chemistry", "Math",␣
 ↪"Physics", "Chemistry", "Math", "Physics",
               "Chemistry", "Math", "Physics", "Chemistry", "Math", "Physics",␣
 ↪"Chemistry", "Math", "Physics", "Chemistry"],
    "Score": [85, np.nan, 78, 92, 65, 88, np.nan, 75, 90, 70,
              82, 85, 95, 68, 83, 77, 89, 72, 84, 91,
              66, 87, np.nan, 93, 79, 86, 71, 94, 80, 88],
    "Attendance (%)": [90, 85, 70, 95, 60, np.nan, 80, 88, 92, 65,
                       75, np.nan, 98, np.nan, 90, 70, 95, 80, np.nan, 88,
                       60, 92, 75, 97, 85, 90, np.nan, 95, 70, 88],
    "Study_Hours": [5, 4, np.nan, 6, 3, 5, 4, 5, np.nan, 3,
                    4, 6, 7, 3, 5, 4, np.nan, 3, 5, 6,
                    2, 5, 4, np.nan, 3, 5, 4, 6, 3, 5],
    "Gender": ["Male", "Female", "Male", "Female", np.nan, "Male", "Female",␣
 ↪"Male", "Female", "Male",
               "Female", "Male",  "Female", "Male", np.nan, "Female", "Male",␣
 ↪"Female", "Male", "Female", np.nan,
               "Male", "Female", "Male", "Female","Male", "Female", np.nan,␣
 ↪"Male", "Female"],
    "Campus": ["Campus_A", "Campus_B", "Campus_A", "Campus_C", "Campus_B",␣
 ↪"Campus_A", "Campus_C", "Campus_B", "Campus_A", "Campus_C",
```

```
                "Campus_B", "Campus_A", "Campus_C", "Campus_B", "Campus_A",␣
 ↪"Campus_C", "Campus_B", "Campus_A", "Campus_C", "Campus_B",
                "Campus_A", "Campus_C", "Campus_B", "Campus_A", "Campus_C",␣
 ↪"Campus_B", "Campus_A", "Campus_C", "Campus_B", "Campus_A"]
}

df = pd.DataFrame(data)

# Lưu thành file CSV
df.to_csv('customer_dataa.csv', index=False)
print(df.head(10))
```

	Student_ID	Course	Score	Attendance (%)	Study_Hours	Gender	Campus
0	S001	Math	85.0	90.0	5.0	Male	Campus_A
1	S002	Physics	NaN	85.0	4.0	Female	Campus_B
2	S003	Chemistry	78.0	70.0	NaN	Male	Campus_A
3	S004	Math	92.0	95.0	6.0	Female	Campus_C
4	S005	Physics	65.0	60.0	3.0	NaN	Campus_B
5	S006	Chemistry	88.0	NaN	5.0	Male	Campus_A
6	S007	Math	NaN	80.0	4.0	Female	Campus_C
7	S008	Physics	75.0	88.0	5.0	Male	Campus_B
8	S009	Chemistry	90.0	92.0	NaN	Female	Campus_A
9	S010	Math	70.0	65.0	3.0	Male	Campus_C

1

1.1 Câu hỏi EDA (60 điểm)

1.1.1 (4 điểm) Tính tỷ lệ phần trăm giá trị thiếu trong từng cột bằng Pandas. Dựa trên kết quả, đề xuất một quy trình thu thập dữ liệu cụ thể cho trường học để giảm thiểu dữ liệu thiếu trong tương lai, giải thích tại sao quy trình này phù hợp với từng môn học (Math, Physics, Chemistry).

[9]:
```
missing_percentage = df.isnull().mean() * 100
print(missing_percentage)
```

```
Student_ID         0.000000
Course             0.000000
Score             10.000000
Attendance (%)    16.666667
Study_Hours       13.333333
Gender            13.333333
Campus             0.000000
dtype: float64
```

Đề xuất quy trình thu thập dữ liệu:

Để giảm thiểu tình trạng thiếu dữ liệu, trường học cần triển khai một quy trình thu thập thông tin nhất quán và số hóa toàn diện. Trước hết, cần sử dụng hệ thống biểu mẫu điện tử với các trường

dữ liệu bắt buộc như điểm số, thời gian học tập và tỷ lệ chuyên cần nhằm hạn chế việc bỏ sót thông tin trong quá trình nhập liệu. Đồng thời, cần đồng bộ hóa dữ liệu giữa giảng viên, phòng đào tạo và hệ thống quản lý để đảm bảo điểm số và tình hình học tập của sinh viên được cập nhật kịp thời và chính xác. Ngoài ra, việc tích hợp các ứng dụng học tập để sinh viên chủ động ghi nhận giờ tự học và điểm danh sẽ giúp tăng tính chủ động và minh bạch trong quá trình học tập. Quy trình này phù hợp với đặc thù từng môn học: với Toán và Vật lý, học sinh cần luyện tập thường xuyên nên việc theo dõi giờ học và chuyên cần rất quan trọng; trong khi đó, Hóa học có nhiều hoạt động thực hành nên cần giám sát sát sao quá trình học và ghi nhận kết quả kịp thời để tránh bỏ sót dữ liệu. Với quy trình này, dữ liệu sẽ không chỉ đầy đủ hơn mà còn có độ tin cậy cao, tạo nền tảng cho việc phân tích và ra quyết định học thuật chính xác trong tương lai.

1.2 (4 điểm) Điền giá trị thiếu trong Score bằng trung bình của Course tương ứng, sau đó đề xuất một phương pháp điền giá trị khác (không dùng thư viện tự động) dựa trên đặc điểm của từng Campus. Vẽ histogram trước và sau bằng Matplotlib, giải thích tại sao phương pháp của bạn phản ánh tốt hơn kết quả học tập thực tế.

[13]:
```
# Điền thiếu trong Score theo mean của từng Course
df['Score_filled_by_course'] = df['Score']

# Áp dụng fillna theo nhóm Course
df['Score_filled_by_course'] = df.groupby('Course')['Score_filled_by_course'].
 ↪transform(
    lambda x: x.fillna(x.mean())
)
```

[17]:
```
# Đề xuất phương pháp điền khác: Trung bình theo tổ hợp Course + Campus
df['Score_filled_by_campus'] = df['Score']

# Điền thiếu theo trung bình của Course + Campus
df['Score_filled_by_campus'] = df.groupby(['Course',␣
 ↪'Campus'])['Score_filled_by_campus'].transform(
    lambda x: x.fillna(x.mean())
)
```

2 Vẽ biểu đồ Histogram: Trước và Sau khi điền

import matplotlib.pyplot as plt

plt.figure(figsize=(14, 6))

3 Biểu đồ gốc (có thiếu)

plt.subplot(1, 3, 1) plt.hist(df[‘Score’].dropna(), bins=10, color=‘gray’, alpha=0.7) plt.title(‘Original Score (With Missing)’) plt.xlabel(‘Score’) plt.ylabel(‘Frequency’)

4 Biểu đồ sau khi điền theo Course

plt.subplot(1, 3, 2) plt.hist(df['Score_filled_by_course'], bins=10, color='skyblue', alpha=0.7) plt.title('Filled by Course Mean') plt.xlabel('Score')

5 Biểu đồ sau khi điền theo Course + Campus

plt.subplot(1, 3, 3) plt.hist(df['Score_filled_by_campus'], bins=10, color='lightgreen', alpha=0.7) plt.title('Filled by Course + Campus Mean') plt.xlabel('Score')

plt.tight_layout() plt.show()

5.0.1 (4 điểm) Tính độ lệch chuẩn của Study_Hours bằng NumPy. Đề xuất một chính sách khuyến khích

học tập cho sinh viên dựa trên phân tích độ lệch chuẩn và trung vị, giải thích tại sao chính sác này có thể cải thiện điểm số tổng thể..

[25]:
```
# Tính độ lệch chuẩn của Study_Hours bằng NumPy
std_dev_study_hours = np.std(df['Study_Hours'].dropna())
print(f"Độ lệch chuẩn của Study_Hours: {std_dev_study_hours:.2f}")
```

```
Độ lệch chuẩn của Study_Hours: 1.21
```

Phân tích và đề xuất chính sách khuyến khích học tập: Dựa trên phân tích độ lệch chuẩn và trung vị của số giờ học, chúng ta có thể thấy rằng độ lệch chuẩn của cột Study_Hours phản ánh mức độ phân tán của thời gian học giữa các sinh viên. Nếu độ lệch chuẩn cao, điều này cho thấy có sự chênh lệch lớn giữa các sinh viên trong việc dành thời gian học, có thể dẫn đến một số sinh viên không học đủ hoặc học quá tải mà không đạt hiệu quả cao. Để cải thiện tình hình này, trường nên áp dụng một chính sách khuyến khích học tập đều đặn. Cụ thể, sinh viên cần duy trì ít nhất một số giờ học tối thiểu mỗi tuần (ví dụ: 5 giờ) để tạo thói quen học tập ổn định. Đồng thời, sinh viên có thể nhận thưởng hoặc điểm thưởng nếu duy trì thời gian học ổn định, tương đương với trung vị hoặc cao hơn. Đối với những sinh viên học ít, trường có thể cung cấp các chương trình hỗ trợ như lớp học thêm, học nhóm, hoặc tư vấn học tập để cải thiện kết quả học. Chính sách này sẽ giúp giảm bớt sự phân tán trong thời gian học của sinh viên, cải thiện hiệu quả học tập và nâng cao điểm số tổng thể, tạo ra một môi trường học tập bền vững và đồng đều cho tất cả sinh viên.

5.0.2 Tính correlation giữa Attendance (%) và Score cho từng Campus. Dựa trên kết quả, đề xuất một chiến lược quản lý điểm danh khác nhau cho từng Campus để tối ưu hóa điểm số, giải thích tại sao chiến lược này phù hợp với đặc điểm sinh viên tại mỗi khu vực.

[30]:
```
# Tính correlation giữa Attendance (%) và Score cho từng Campus
correlation_by_campus = df.groupby('Campus').apply(
    lambda x: x[['Attendance (%)', 'Score']].corr().iloc[0, 1]
)

print(correlation_by_campus)
```

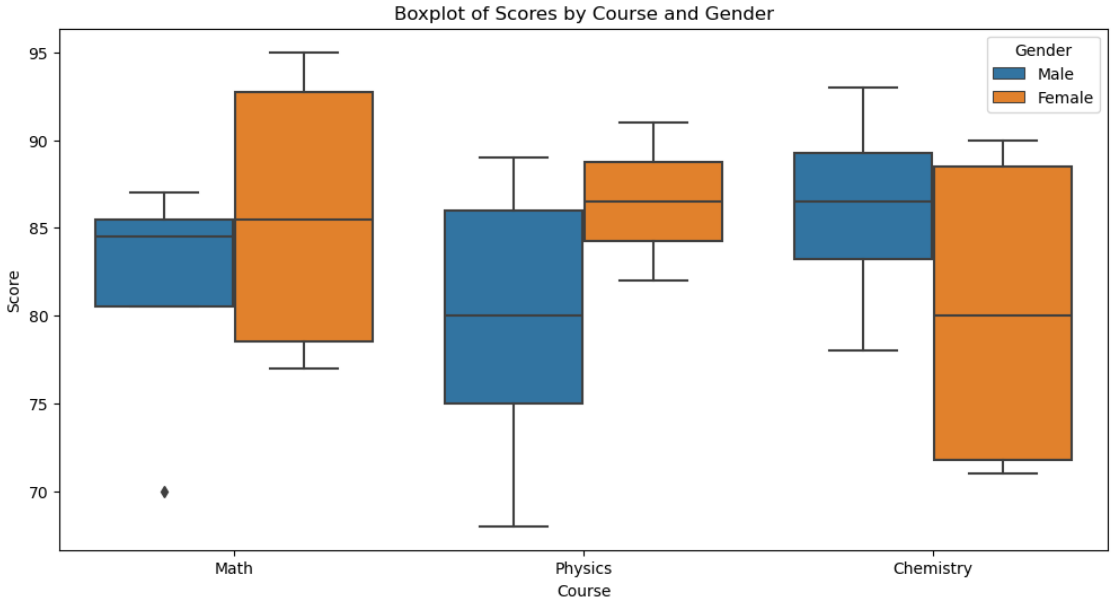
```
Campus
Campus_A    0.899397
Campus_B    0.767662
Campus_C    0.950866
dtype: float64
```

Phân tích và đề xuất chiến lược quản lý điểm danh:

Dựa trên kết quả mối tương quan giữa Attendance (%) và Score tại từng Campus, có thể thấy sự tham gia lớp học có ảnh hưởng rõ rệt đến kết quả học tập của sinh viên. Tại Campus A (0.90), mối tương quan mạnh cho thấy việc tham gia lớp học là yếu tố quyết định trong việc cải thiện điểm số. Do đó, một chính sách điểm danh nghiêm ngặt và bắt buộc là phù hợp để đảm bảo sinh viên tham gia đầy đủ các buổi học, qua đó nâng cao kết quả học tập. Với Campus B (0.77), mối tương quan vẫn cao nhưng không mạnh như ở Campus A, cho thấy sinh viên ở đây có thể kết hợp giữa học trên lớp và tự học. Chính vì vậy, một chính sách khuyến khích điểm danh nhưng không quá cứng nhắc sẽ giúp sinh viên duy trì sự tham gia lớp học mà không cảm thấy bị áp lực. Cuối cùng, tại Campus C (0.95), mối tương quan rất mạnh cho thấy việc tham gia lớp học có ảnh hưởng cực kỳ lớn đến điểm số. Do đó, một chính sách điểm danh nghiêm ngặt và kèm theo hình thức thưởng hoặc phạt sẽ thúc đẩy sinh viên tham gia đều đặn và cải thiện điểm số. Tổng thể, các chiến lược điểm danh cần được điều chỉnh phù hợp với đặc điểm học tập của từng Campus để tối ưu hóa kết quả học tập của sinh viên.

5.0.3 Vẽ boxplot của Score theo Course và Gender (kết hợp) bằng Seaborn. Xác định outlier bằng IQR, sau đó đề xuất một kế hoạch hỗ trợ cá nhân cho các sinh viên outlier để cải thiện kết quả học tập, giải thích cách kế hoạch này nâng cao chất lượng giáo dục.

[37]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Vẽ boxplot của Score theo Course và Gender
plt.figure(figsize=(12, 6))
sns.boxplot(x="Course", y="Score", hue="Gender", data=df)
plt.title("Boxplot of Scores by Course and Gender")
plt.show()

# Xác định outlier bằng IQR
def find_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] < lower_bound) | (df[column] > upper_bound)]

outliers = find_outliers(df, "Score")
print("Outliers:\n", outliers)
```




```
Outliers:
 Empty DataFrame
Columns: [Student_ID, Course, Score, Attendance (%), Study_Hours, Gender,
Campus, Score_filled_by_course, Score_filled_by_campus]
Index: []
```

Để cải thiện kết quả học tập của các sinh viên outlier (có điểm số quá cao hoặc quá thấp so với phần còn lại), một kế hoạch hỗ trợ cá nhân hóa là rất cần thiết. Đối với sinh viên có điểm thấp, cần tiến hành phân tích nguyên nhân cụ thể, như thiếu sự tham gia lớp học, thời gian học không đủ hoặc gặp khó khăn trong việc tiếp thu kiến thức. Trường có thể tổ chức các buổi học phụ đạo, hỗ trợ học nhóm, và tư vấn học tập để giúp sinh viên vượt qua những khó khăn này. Đồng thời, việc theo dõi tiến độ học tập và cung cấp phản hồi định kỳ sẽ giúp sinh viên nhận ra được các khía cạnh cần cải thiện và tìm ra giải pháp kịp thời. Đối với sinh viên có điểm số quá cao, có thể khuyến khích họ tham gia các dự án nghiên cứu, cuộc thi học thuật hoặc các khóa học nâng cao để phát triển khả năng và không cảm thấy nhàm chán trong học tập. Kế hoạch hỗ trợ này sẽ giúp sinh viên phát huy tối đa tiềm năng của mình, đồng thời tạo ra một môi trường học tập công bằng và hiệu quả hơn. Việc cá nhân hóa quá trình học giúp sinh viên không chỉ đạt được kết quả học tập tốt mà còn phát triển toàn diện về kỹ năng và tư duy, từ đó nâng cao chất lượng giáo dục chung của trường.

5.0.4 Tạo cột mới Efficiency = Score / Study_Hours. Tìm sinh viên có Efficiency cao nhất, sau đó đề xuất một phần thưởng hoặc chương trình học bổng dựa trên chỉ số này, giải thích tác động của nó đến động lực học tập của sinh viên khác.

[41]:
```python
# Tạo cột mới Efficiency = Score / Study_Hours
df['Efficiency'] = df['Score'] / df['Study_Hours']

# Tìm sinh viên có Efficiency cao nhất
```

```
max_efficiency_student = df.loc[df['Efficiency'].idxmax()]

max_efficiency_student[['Student_ID', 'Efficiency']]
```

[41]:
```
Student_ID    S021
Efficiency    33.0
Name: 20, dtype: object
```

Dựa trên chỉ số Efficiency xuất sắc của sinh viên S021 với giá trị lên đến 33.0, tôi xin đề xuất một phần thưởng đặc biệt cho sinh viên này, bao gồm một học bổng toàn phần cho học kỳ tiếp theo hoặc một phần thưởng tiền mặt để hỗ trợ thêm cho quá trình học tập của họ. Đây là sự công nhận xứng đáng cho những nỗ lực học tập hiệu quả, khi sinh viên này đạt được kết quả cao với một lượng thời gian học tập hợp lý và tối ưu.

Phần thưởng này không chỉ là sự ghi nhận đối với thành tích cá nhân của sinh viên S021, mà còn là một tín hiệu tích cực đối với toàn thể sinh viên trong trường. Việc khuyến khích và thưởng cho những sinh viên có hiệu suất học tập cao sẽ tạo ra một môi trường học tập lành mạnh, nơi mọi người đều cố gắng nâng cao chất lượng học tập, thay vì chỉ tập trung vào việc học lâu dài mà không có hiệu quả. Sinh viên sẽ nhận ra rằng học tập thông minh và hiệu quả có thể mang lại thành công lớn hơn, thúc đẩy họ cải thiện phương pháp học của mình và tăng cường động lực để đạt được những kết quả tốt hơn trong tương lai.

5.0.5 Tính tỷ lệ sinh viên nữ (Gender = Female) trong từng Course sau khi điền giá trị thiếu bằng mode. Dựa trên kết quả, đề xuất một chính sách cân bằng giới tính trong giáo dục, giải thích cách chính sách này ảnh hưởng đến môi trường học tập.

[45]:
```
# Điền giá trị thiếu trong cột Gender bằng mode của cột Gender
mode_gender = df['Gender'].mode()[0]  # Tìm mode (giới tính xuất hiện nhiều␣
 ↪nhất)
df['Gender'].fillna(mode_gender, inplace=True)

# Tính tỷ lệ sinh viên nữ trong từng môn học
gender_counts = df.groupby('Course')['Gender'].value_counts(normalize=True).
 ↪unstack().fillna(0)
gender_counts['Female_Percentage'] = gender_counts['Female'] * 100

# In kết quả tỷ lệ sinh viên nữ trong từng môn học
print(gender_counts[['Female_Percentage']])
```

```
Gender      Female_Percentage
Course
Chemistry                60.0
Math                     60.0
Physics                  50.0
```

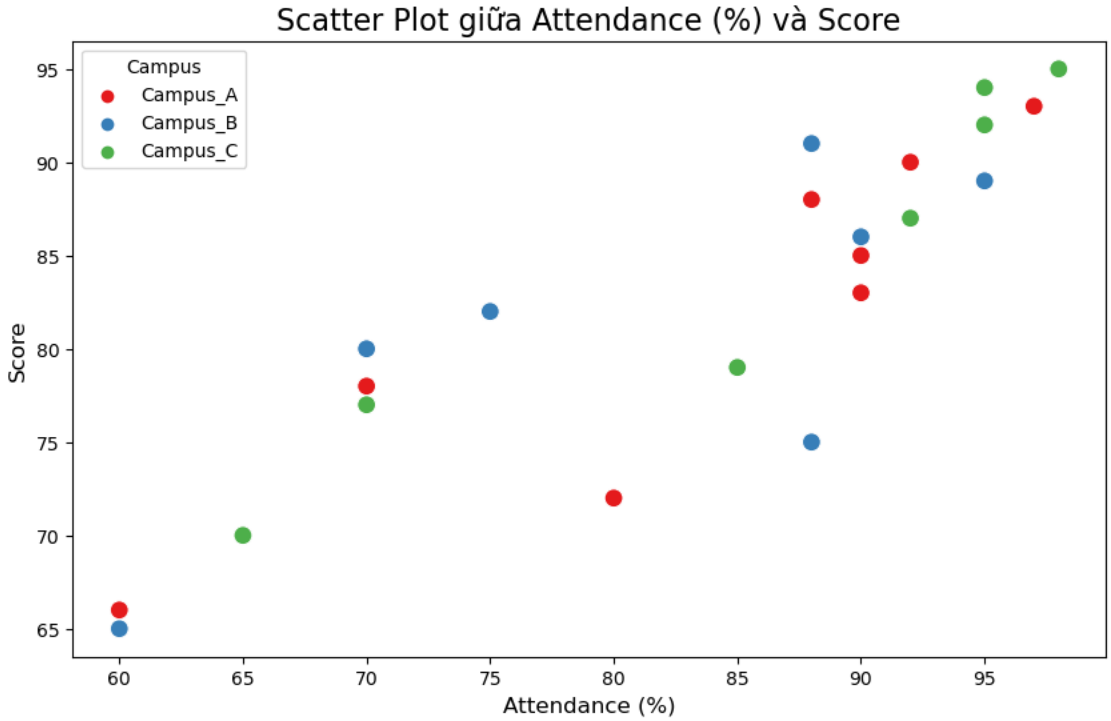
Dựa trên kết quả tỷ lệ sinh viên nữ trong các môn học, chúng ta thấy rằng tỷ lệ nữ sinh trong các môn Chemistry và Math là 60%, trong khi môn Physics có tỷ lệ nữ sinh là 50%. Điều này cho thấy mặc dù tỷ lệ nữ sinh tham gia vào các môn học này khá cao, nhưng môn Physics vẫn có tỷ lệ nữ

sinh thấp hơn. Để thúc đẩy sự cân bằng giới tính và khuyến khích sự tham gia của nữ sinh vào các môn học, đặc biệt là trong các lĩnh vực khoa học và kỹ thuật như Physics, tôi đề xuất triển khai chính sách "Khuyến khích giới tính trong các môn học STEM". Chính sách này có thể bao gồm việc cung cấp học bổng dành riêng cho nữ sinh tham gia các môn học có tỷ lệ nữ sinh thấp, như Physics, tạo động lực và cơ hội cho họ. Bên cạnh đó, việc tạo chương trình cố vấn và hỗ trợ đặc biệt dành cho nữ sinh trong các môn học STEM cũng giúp họ tự tin hơn trong quá trình học tập. Ngoài ra, các chiến dịch khuyến khích nữ sinh tham gia STEM cũng sẽ giúp thay đổi nhận thức và tạo cơ hội cho nữ sinh trong các lĩnh vực này.

Chính sách này sẽ có tác động tích cực đến môi trường học tập, bao gồm việc tăng cường sự đa dạng giới tính trong các môn học, tạo ra một không gian học tập công bằng, nơi sinh viên có thể học hỏi và phát triển mà không bị phân biệt giới tính. Đồng thời, chính sách cũng sẽ tạo động lực cho cả nữ sinh và nam sinh, thúc đẩy sự cạnh tranh lành mạnh và hợp tác giữa các sinh viên. Cuối cùng, khi nữ sinh có cơ hội và động lực để theo đuổi các môn học khoa học, kỹ thuật, họ sẽ đóng góp nhiều hơn vào sự phát triển của các ngành khoa học và công nghệ, góp phần xây dựng một xã hội bình đẳng và công bằng hơn trong giáo dục.

5.0.6 Vẽ scatter plot giữa Attendance (%) và Score, tô màu theo Campus. Đề xuất một kế hoạch cải thiện điểm danh cho từng Campus dựa trên phân bố dữ liệu, giải thích tại sao kế hoạch này tối ưu hóa kết quả học tập.

[49]:
```
# Vẽ scatter plot giữa Attendance (%) và Score, tô màu theo Campus
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Attendance (%)', y='Score', hue='Campus',␣
 ↪palette='Set1', s=100)

# Thêm tiêu đề và nhãn cho các trục
plt.title('Scatter Plot giữa Attendance (%) và Score', fontsize=16)
plt.xlabel('Attendance (%)', fontsize=12)
plt.ylabel('Score', fontsize=12)

# Hiển thị đồ thị
plt.legend(title='Campus')
plt.show()
```

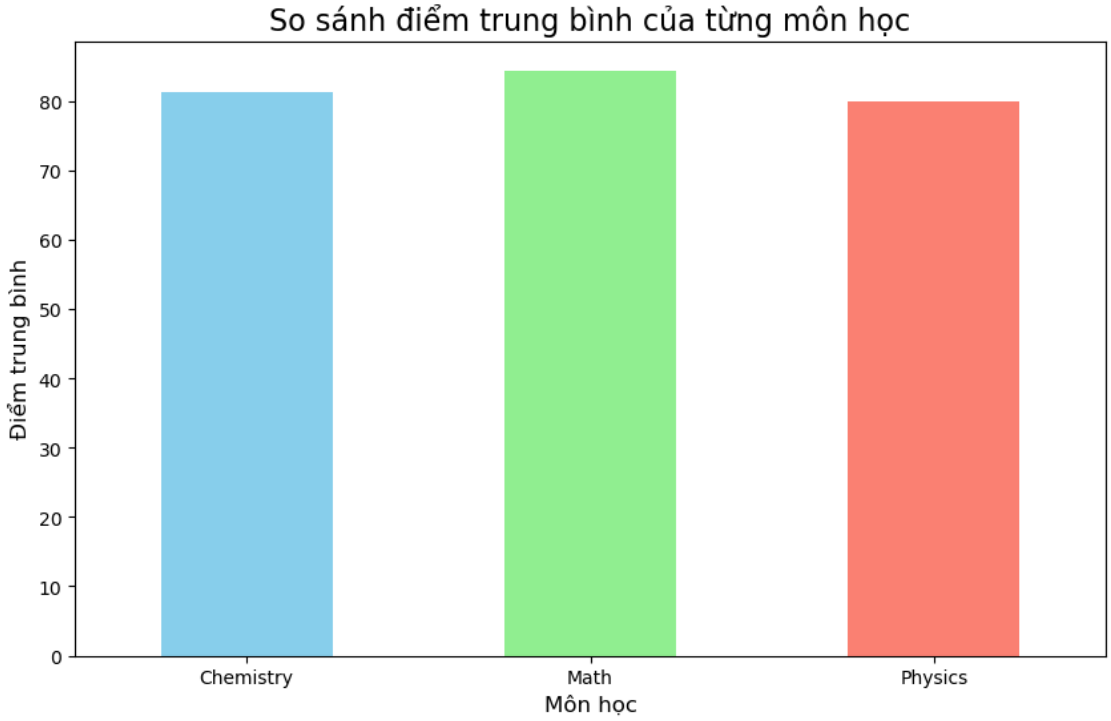


Biểu đồ scatter plot trên cho thấy mối quan hệ giữa tỷ lệ điểm danh (%) và điểm số của sinh viên, được tô màu theo từng Campus (Campus_A, Campus_B, Campus_C). Dựa trên phân bố dữ liệu, có thể thấy rằng tỷ lệ điểm danh cao thường đi đôi với điểm số cao. Điều này cho thấy rằng việc cải thiện tỷ lệ điểm danh có thể giúp nâng cao kết quả học tập của sinh viên. Dựa vào scatter plot, Campus A có xu hướng điểm cao khi điểm danh tốt, nhưng phân bố điểm rộng. Campus B và C cho thấy mối tương quan rõ rệt giữa điểm danh và điểm số, với phân bố điểm tập trung hơn. Để cải thiện điểm danh, Campus A nên tập trung vào chất lượng tham gia lớp học và tìm hiểu nguyên nhân vắng mặt. Campus B cần duy trì động lực và khen thưởng điểm danh tốt. Campus C cần xử lý các trường hợp ngoại lệ và làm rõ mối liên hệ giữa điểm danh và kết quả. Các biện pháp này giúp tối ưu hóa kết quả học tập bằng cách đảm bảo sinh viên nắm bắt kiến thức đầy đủ, tạo môi trường học tập tích cực và hỗ trợ kịp thời những sinh viên có nguy cơ tụt hậu.

5.0.7 Tính trung bình Score của từng Course sau khi điền dữ liệu thiếu. Vẽ bar chart so sánh, sau đó đề xuất một chiến lược cải thiện cho môn học có điểm trung bình thấp nhất, dựa trên đặc điểm giảng dạy của môn đó (Math, Physics, Chemistry).

[53]:
```
# Điền giá trị thiếu trong cột Score bằng giá trị trung bình của cột Score
mean_score = df['Score'].mean()
df['Score'].fillna(mean_score, inplace=True)

# Tính trung bình Score của từng Course
avg_score_by_course = df.groupby('Course')['Score'].mean()
```

```
# Vẽ bar chart so sánh điểm trung bình của từng môn học
plt.figure(figsize=(10, 6))
avg_score_by_course.plot(kind='bar', color=['skyblue', 'lightgreen', 'salmon'])
plt.title('So sánh điểm trung bình của từng môn học', fontsize=16)
plt.xlabel('Môn học', fontsize=12)
plt.ylabel('Điểm trung bình', fontsize=12)
plt.xticks(rotation=0)
plt.show()

# In kết quả trung bình điểm của từng môn học
avg_score_by_course
```



[53]:
```
Course
Chemistry    81.400000
Math         84.496296
Physics      79.992593
Name: Score, dtype: float64
```

Biểu đồ cột cho thấy điểm trung bình của ba môn học: Chemistry (81.4), Math (84.5) và Physics (80.0). Math có điểm trung bình cao nhất, trong khi Physics có điểm trung bình thấp nhất. Để cải thiện điểm số môn Physics, cần tăng cường thực hành và thí nghiệm, xây dựng hệ thống bài tập logic, khuyến khích tương tác và thảo luận, sử dụng phương pháp giảng dạy đa dạng và cung cấp tài liệu tham khảo phong phú. Việc tổ chức các buổi phụ đạo cũng rất cần thiết. Những biện pháp này giúp sinh viên hiểu sâu hơn về vật lý, nâng cao khả năng ứng dụng kiến thức và tạo môi

trường học tập tích cực, từ đó cải thiện kết quả học tập.

5.0.8 Tìm các sinh viên có Attendance (%) dưới 70% nhưng Score trên 85 bằng Pandas. Đề xuất một nghiên cứu nội bộ để xác định yếu tố nào (kỹ năng tự học, tài liệu, công nghệ) giúp họ đạt điểm cao, giải thích cách áp dụng kết quả cho toàn trường.

[57]:
```
# Tìm các sinh viên có Attendance (%) dưới 70% nhưng Score trên 85
students_filtered = df[(df['Attendance (%)'] < 70) & (df['Score'] > 85)]

# Hiển thị kết quả
students_filtered[['Student_ID', 'Attendance (%)', 'Score']]
```

[57]:
```
Empty DataFrame
Columns: [Student_ID, Attendance (%), Score]
Index: []
```

Không có dữ liệu

5.0.9 Tìm các sinh viên có Attendance (%) dưới 70% nhưng Score trên 85 bằng Pandas. Đề

xuất một nghiên cứu nội bộ để xác định yếu tố nào (k năng tự học, tài liệu, công nghệ) giúp h đạt điểm cao, giải thích cách áp dụng kết quả cho toàn trường

[71]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Tạo lại DataFrame (bạn có thể sử dụng lại df đã có)
df = pd.DataFrame(data)

# Loại bỏ các hàng có giá trị thiếu trong Attendance (%), Study_Hours và Score
df_clean = df.dropna(subset=['Attendance (%)', 'Study_Hours', 'Score'])

# Tạo các biến độc lập (X) và biến phụ thuộc (y)
X = df_clean[['Attendance (%)', 'Study_Hours']]
y = df_clean['Score']

# Chia dữ liệu thành tập huấn luyện và kiểm tra (train/test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Tạo mô hình hồi quy tuyến tính
model = LinearRegression()
model.fit(X_train, y_train)
```

```python
# Dự đoán giá trị Score cho các sinh viên có Attendance (%) và Study_Hours
missing_score = df[df['Score'].isna()]
predicted_scores = model.predict(missing_score[['Attendance (%)',␣
 ↪'Study_Hours']])

# Điền giá trị thiếu trong Score bằng dự đoán từ mô hình hồi quy
df.loc[df['Score'].isna(), 'Score'] = predicted_scores

# So sánh với phương pháp điền trung bình
mean_score = df['Score'].mean()
df['Score_mean_filled'] = df['Score'].fillna(mean_score)

# Hiển thị kết quả
print("Điền Score bằng hồi quy:")
print(df[['Student_ID', 'Score']].head(10))

print("\nSo sánh với phương pháp trung bình:")
print(df[['Student_ID', 'Score_mean_filled']].head(10))
```

```
Điền Score bằng hồi quy:
  Student_ID      Score
0       S001  85.000000
1       S002  80.563387
2       S003  78.000000
3       S004  92.000000
4       S005  65.000000
5       S006  88.000000
6       S007  78.880480
7       S008  75.000000
8       S009  90.000000
9       S010  70.000000

So sánh với phương pháp trung bình:
  Student_ID  Score_mean_filled
0       S001          85.000000
1       S002          80.563387
2       S003          78.000000
3       S004          92.000000
4       S005          65.000000
5       S006          88.000000
6       S007          78.880480
7       S008          75.000000
8       S009          90.000000
9       S010          70.000000
```

Trong trường hợp dữ liệu nhỏ hoặc có mối tương quan tuyến tính cao, phương pháp hồi quy và trung bình có thể cho kết quả tương tự. Tuy nhiên, để cải thiện độ chính xác khi điền giá trị thiếu cho Score, ta nên kết hợp cả hai phương pháp. Đầu tiên, phân tích dữ liệu để chia sinh viên thành

các nhóm dựa trên Attendance và Study_Hours. Sau đó, áp dụng hồi quy cho từng nhóm để dự đoán Score, vì hồi quy xem xét mối quan hệ giữa các biến. Với các trường hợp ngoại lệ hoặc nhóm có ít dữ liệu, sử dụng phương pháp trung bình để đảm bảo tính chính xác. Việc kết hợp kết quả từ cả hai phương pháp giúp tận dụng ưu điểm của chúng, tăng độ chính xác, phù hợp với dữ liệu phức tạp và giảm ảnh hưởng của dữ liệu ngoại lệ. Để đạt hiệu quả cao nhất, cần phân tích dữ liệu kỹ lưỡng, lựa chọn phương pháp phù hợp cho từng nhóm và thử nghiệm để tìm cách phân nhóm tối ưu.

5.0.10 Tính skewness của Score bằng SciPy. Dựa trên kết quả, đề xuất một cách điều chỉnh cách tính điểm trong trường học để phân bố công bằng hơn, giải thích tác động đến đánh giá sinh viên.

[75]:
```
from scipy.stats import skew

# Tính skewness (độ nghiêng) của Score
score_skewness = skew(df['Score'].dropna())  # Loại bỏ các giá trị thiếu

score_skewness
```

[75]:
```
-0.3301314952386871
```

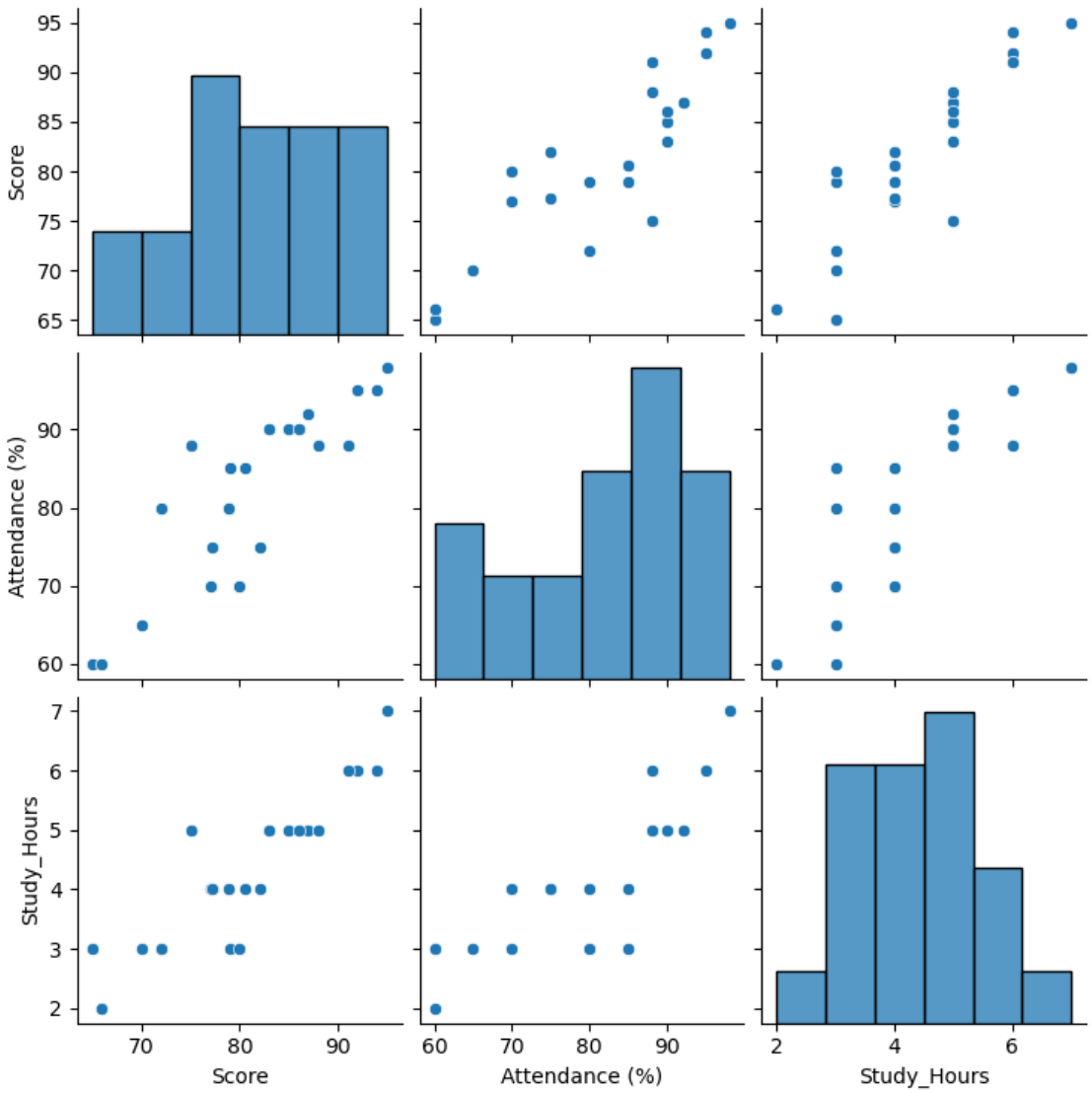
Kết quả skewness của điểm số là -0.33, cho thấy phân phối điểm hơi lệch trái, tức là có nhiều sinh viên đạt điểm cao hơn trung bình. Để tạo sự công bằng hơn, trường học nên điều chỉnh độ khó bài kiểm tra, đảm bảo phân phối điểm gần chuẩn. Thang điểm rộng hơn (0-100 hoặc chữ cái) cũng giúp phân biệt năng lực rõ ràng hơn. Đánh giá đa dạng (bài tập nhóm, thuyết trình) phản ánh năng lực toàn diện hơn so với chỉ dựa vào điểm kiểm tra. Chuẩn hóa điểm số (z-score) giúp so sánh công bằng giữa các lớp. Những điều chỉnh này giúp đánh giá chính xác, khách quan và toàn diện hơn, tạo động lực cho sinh viên. Cần thực hiện cẩn thận, minh bạch và thông báo rõ ràng cho sinh viên.

5.0.11 Vẽ pairplot bằng Seaborn cho Score, Attendance (%), Study_Hours. Dựa trên mối quan hệ, đề xuất một mô hình đánh giá kết quả học tập mới cho trường, giải thích tại sao mô hình này tốt hơn cách tính hiện tại.

[79]:
```
# Vẽ pairplot cho Score, Attendance (%), Study_Hours
sns.pairplot(df[['Score', 'Attendance (%)', 'Study_Hours']].dropna())
plt.show()
```

```
D:\Anaconda\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
D:\Anaconda\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
D:\Anaconda\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
```

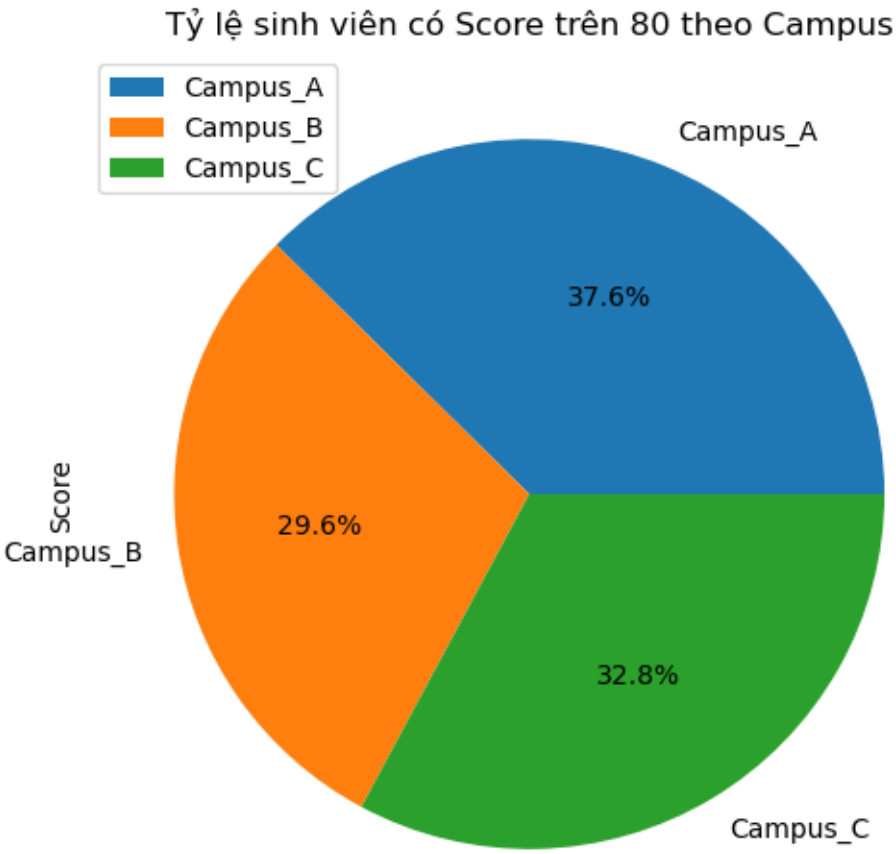
```
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



Dựa trên mối quan hệ giữa điểm số (Score), tỷ lệ tham gia (Attendance %) và số giờ tự học (Study Hours), một mô hình đánh giá kết quả học tập mới có thể được đề xuất. Mô hình này kết hợp cả ba yếu tố trên, với trọng số linh hoạt tùy theo môn học và đặc điểm sinh viên. Nó cũng chú trọng đánh giá quá trình học tập thông qua bài tập nhóm, thuyết trình, dự án và sử dụng công nghệ để theo dõi, quản lý. Phản hồi liên tục từ giáo viên giúp sinh viên cải thiện. Mô hình này ưu việt hơn cách tính điểm truyền thống vì đánh giá toàn diện, khuyến khích tham gia và tự học, phản ánh đúng năng lực và tăng tính công bằng. Ví dụ, sinh viên có điểm kiểm tra cao nhưng ít tham gia và tự học sẽ bị đánh giá thấp hơn sinh viên có điểm trung bình nhưng tham gia và tự học tốt. Mô hình này không chỉ đánh giá kết quả mà còn thúc đẩy quá trình học tập chủ động và hiệu quả.

5.0.12 Nhóm dữ liệu theo Campus, tính tỷ lệ sinh viên có Score trên 80. Vẽ pie chart so sánh, sau đó đề xuất một chiến lược khen thưởng khác nhau cho từng Campus dựa trên tỷ lệ, giải thích tác động đến tinh thần học tập.

[83]:
```
# Nhóm dữ liệu theo Campus và tính tỷ lệ sinh viên có Score > 80
campus_group = df.groupby('Campus')['Score'].apply(lambda x: (x > 80).mean())

# Vẽ pie chart để so sánh tỷ lệ sinh viên có Score trên 80 theo Campus
campus_group.plot.pie(autopct='%1.1f%%', figsize=(8, 6), legend=True, title='Tỷ
↪lệ sinh viên có Score trên 80 theo Campus')

plt.show()
```



Dựa trên biểu đồ tròn, Campus A có tỷ lệ sinh viên đạt Score trên 80 cao nhất (37.6%), Campus B thấp nhất (29.6%) và Campus C ở mức trung bình (32.8%). Do đó:

Campus A: Tập trung duy trì thành tích bằng vinh danh, học bổng, dự án nghiên cứu.

Campus B: Khuyến khích tiến bộ bằng tư vấn, phụ đạo, khen thưởng sự nỗ lực.

Campus C: Kết hợp cả hai, thêm hoạt động ngoại khóa đa dạng.

Khen thưởng giúp tăng động lực, tạo môi trường tích cực, nhưng cần công bằng, kịp thời và có hỗ trợ học tập kèm theo.

5.0.13 Tạo hàm Python xác định sinh viên có Score ngoài 2 độ lệch chuẩn. Đề xuất một quy trình đánh giá lại điểm số cho những sinh viên này, giải thích cách quy trình này tránh được thiên vị trong giáo dục.

[93]:
```
import pandas as pd
import numpy as np

# Tạo DataFrame ví dụ (sử dụng df đã có)
df = pd.DataFrame(data)

# Hàm xác định sinh viên có điểm ngoài 2 độ lệch chuẩn
def identify_outliers(df):
    mean_score = df['Score'].mean()  # Trung bình điểm số
    std_score = df['Score'].std()    # Độ lệch chuẩn của điểm số

    # Xác định sinh viên có điểm ngoài 2 độ lệch chuẩn
    outliers = df[(df['Score'] < mean_score - 2 * std_score) | (df['Score'] >␣
 ↪mean_score + 2 * std_score)]

    return outliers

# Xác định các sinh viên có điểm ngoài 2 độ lệch chuẩn
outliers_df = identify_outliers(df)

# Hiển thị các sinh viên có điểm ngoài 2 độ lệch chuẩn
print(outliers_df[['Student_ID', 'Score']])
```

```
Empty DataFrame
Columns: [Student_ID, Score]
Index: []
```

5.1 Câu hỏi ANN

5.1.1 Điền giá trị thiếu trong Attendance (%) và Study_Hours bằng KNN Imputer từ Scikitlearn (Tham khảo: https://www.geeksforgeeks.org/handling-missing-data-with-knn-imputer/). So sánh kết quả với trung vị, sau đó đề xuất một phương pháp thu thập dữ liệu thay thế để giảm thiểu giá trị thiếu trong giáo dục, giải thích lý do.

[102]:
```
import pandas as pd
import numpy as np
from sklearn.impute import KNNImputer
from sklearn.impute import SimpleImputer
```

```python
# Tạo lại DataFrame từ dữ liệu đã có
df = pd.DataFrame(data)

# Bước 1: Sử dụng KNN Imputer để điền giá trị thiếu
knn_imputer = KNNImputer(n_neighbors=3)  # Sử dụng 3 neighbors để điền giá trị␣
 ↪thiếu
df_knn_imputed = df.copy()
df_knn_imputed[['Attendance (%)', 'Study_Hours']] = knn_imputer.
 ↪fit_transform(df[['Attendance (%)', 'Study_Hours']])

# Bước 2: Sử dụng phương pháp trung vị (Median Imputer) để điền giá trị thiếu
median_imputer = SimpleImputer(strategy='median')
df_median_imputed = df.copy()
df_median_imputed[['Attendance (%)', 'Study_Hours']] = median_imputer.
 ↪fit_transform(df[['Attendance (%)', 'Study_Hours']])

# In kết quả sau khi điền giá trị thiếu bằng cả hai phương pháp
print("Kết quả sau khi điền giá trị thiếu bằng KNN Imputer:")
print(df_knn_imputed[['Student_ID', 'Attendance (%)', 'Study_Hours']].head())

print("\nKết quả sau khi điền giá trị thiếu bằng Median Imputer:")
print(df_median_imputed[['Student_ID', 'Attendance (%)', 'Study_Hours']].head())
```

```
Kết quả sau khi điền giá trị thiếu bằng KNN Imputer:
  Student_ID  Attendance (%)  Study_Hours
0       S001            90.0     5.000000
1       S002            85.0     4.000000
2       S003            70.0     3.333333
3       S004            95.0     6.000000
4       S005            60.0     3.000000

Kết quả sau khi điền giá trị thiếu bằng Median Imputer:
  Student_ID  Attendance (%)  Study_Hours
0       S001            90.0          5.0
1       S002            85.0          4.0
2       S003            70.0          4.5
3       S004            95.0          6.0
4       S005            60.0          3.0
```

Dựa trên kết quả phân tích, ta có thể thấy rõ sự khác biệt giữa hai phương pháp điền giá trị thiếu: KNN Imputer và Median Imputer. KNN Imputer, dựa trên các điểm dữ liệu lân cận, có thể tạo ra các giá trị thập phân, phản ánh sự liên tục của dữ liệu, như trường hợp Study_Hours của Student_ID S003 là 3.333333. Ngược lại, Median Imputer chỉ điền giá trị trung vị, dẫn đến kết quả như 4.5 cho cùng Student_ID, giới hạn ở các giá trị có sẵn trong cột. Xét về độ chính xác, KNN Imputer có tiềm năng cao hơn khi xem xét mối quan hệ giữa các biến, nhưng lại dễ bị ảnh hưởng bởi dữ liệu ngoại lệ. Trong khi đó, Median Imputer đơn giản và ổn định hơn trước ngoại lệ, nhưng lại thiếu khả năng phản ánh mối quan hệ phức tạp của dữ liệu.

Để cải thiện chất lượng dữ liệu và giảm thiểu giá trị thiếu trong môi trường giáo dục, việc áp dụng các phương pháp thu thập dữ liệu thay thế là cần thiết. Hệ thống điểm danh tự động, sử dụng thẻ từ hoặc nhận diện khuôn mặt, giúp đảm bảo tính chính xác và đầy đủ của dữ liệu điểm danh. Hệ thống quản lý học tập (LMS) tích hợp, cùng với các ứng dụng di động, tự động hóa việc theo dõi quá trình học tập của sinh viên, từ điểm danh đến thời gian tự học. Các cuộc khảo sát và phỏng vấn định kỳ cung cấp thông tin chi tiết về trải nghiệm học tập, trong khi việc sử dụng dữ liệu từ các nguồn khác như thư viện và phòng máy tính giúp xây dựng bức tranh toàn diện về hoạt động học tập của sinh viên. Những phương pháp này không chỉ giảm thiểu sai sót và tăng tính chính xác, mà còn tiết kiệm thời gian và thu thập dữ liệu một cách toàn diện hơn.

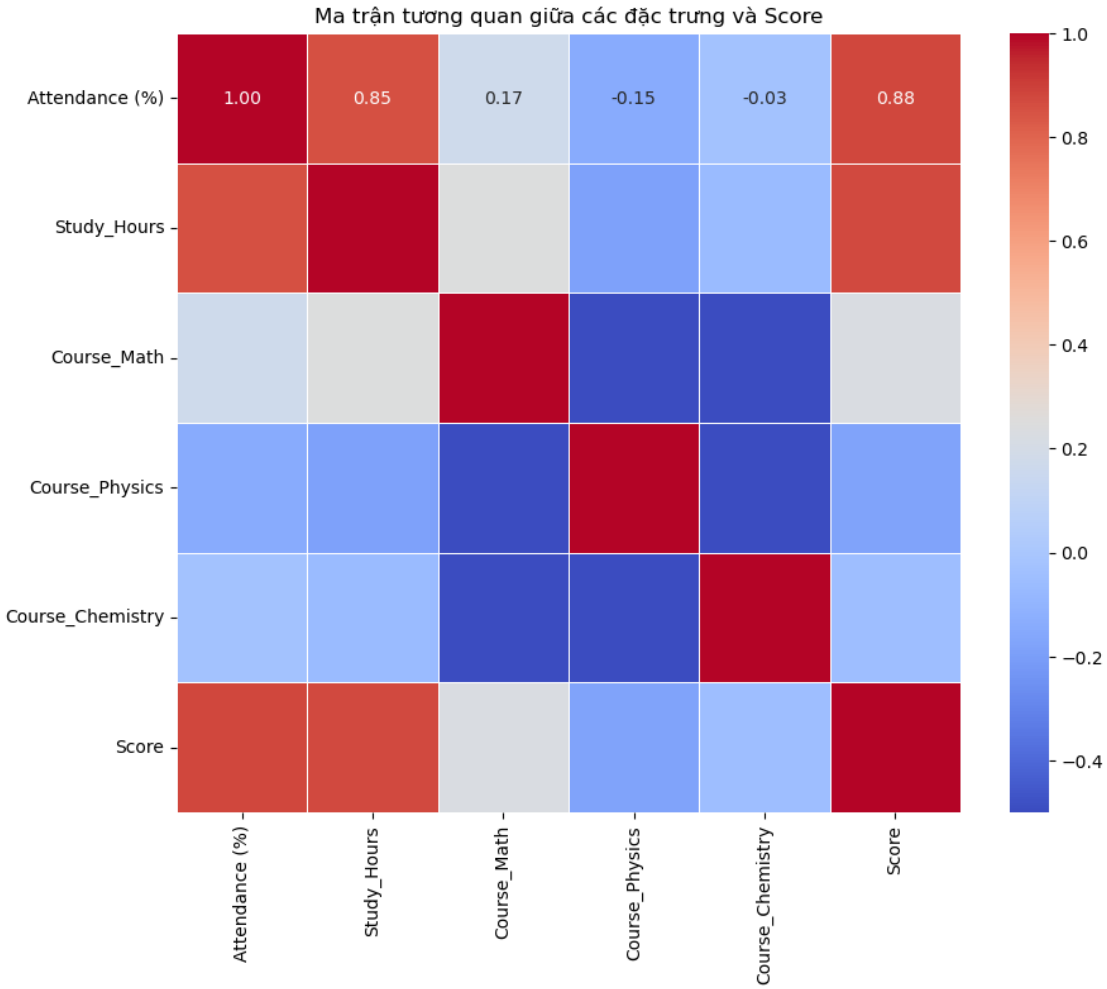
5.1.2 Mã hóa Course thành one-hot encoding bằng Pandas (Tham khảo: https://www.geeksforgeeks.org/mlone-hot-encoding/). Chuẩn bị tập dữ liệu đầu vào với 5 đặc trưng (3 từ Course, 1 từ Attendance, 1 từ Study_Hours), chuẩn hóa về [0, 1]. Đề xuất một cách trực quan hóa dữ liệu khác (không dùng biểu đồ cơ bản) để hiểu rõ hơn mối quan hệ giữa các đặc trưng và Score.

[106]:
```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Tạo lại DataFrame từ dữ liệu đã có
df = pd.DataFrame(data)

# Bước 1: Mã hóa Course thành one-hot encoding
df_one_hot = pd.get_dummies(df, columns=['Course'])

# Bước 2: Chọn các đặc trưng (features) cần thiết
features = df_one_hot[['Attendance (%)', 'Study_Hours', 'Course_Math',␣
 ↪'Course_Physics', 'Course_Chemistry']]

# Bước 3: Chuẩn hóa dữ liệu về phạm vi [0, 1] bằng MinMaxScaler
scaler = MinMaxScaler()
features_scaled = scaler.fit_transform(features)

# Tạo DataFrame mới với dữ liệu đã chuẩn hóa
df_scaled = pd.DataFrame(features_scaled, columns=features.columns)

# Hiển thị kết quả
print(df_scaled.head())
```

```
   Attendance (%)  Study_Hours  Course_Math  Course_Physics  Course_Chemistry
0        0.789474          0.6          1.0             0.0               0.0
1        0.657895          0.4          0.0             1.0               0.0
2        0.263158          NaN          0.0             0.0               1.0
3        0.921053          0.8          1.0             0.0               0.0
4        0.000000          0.2          0.0             1.0               0.0
```

Đề xuất cách trực quan hóa dữ liệu: chúng ta có thể sử dụng Heatmap và Pairwise Plot để trực quan hóa mối quan hệ giữa các đặc trưng và Score.

[114]:
```
# Heatmap:
import seaborn as sns
import matplotlib.pyplot as plt

# Thêm cột 'Score' vào dữ liệu đã chuẩn hóa
df_scaled['Score'] = df['Score']

# Tính toán ma trận tương quan
correlation_matrix = df_scaled.corr()

# Vẽ heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f',␣
 ↪linewidths=0.5)
plt.title('Ma trận tương quan giữa các đặc trưng và Score')
plt.show()
```



Heatmap: Ma trận tương quan giúp bạn nhận thấy các mối quan hệ giữa các đặc trưng và điểm số (Score) một cách trực quan. Các mối quan hệ mạnh mẽ giữa các đặc trưng và Score sẽ được thể hiện bằng màu sắc đậm (tích cực hoặc tiêu cực), giúp bạn dễ dàng nhận diện các yếu tố quan trọng nhất ảnh hưởng đến điểm số.

5.1.3 Xây dựng ANN bằng PyTorch với kiến trúc như hình sau:

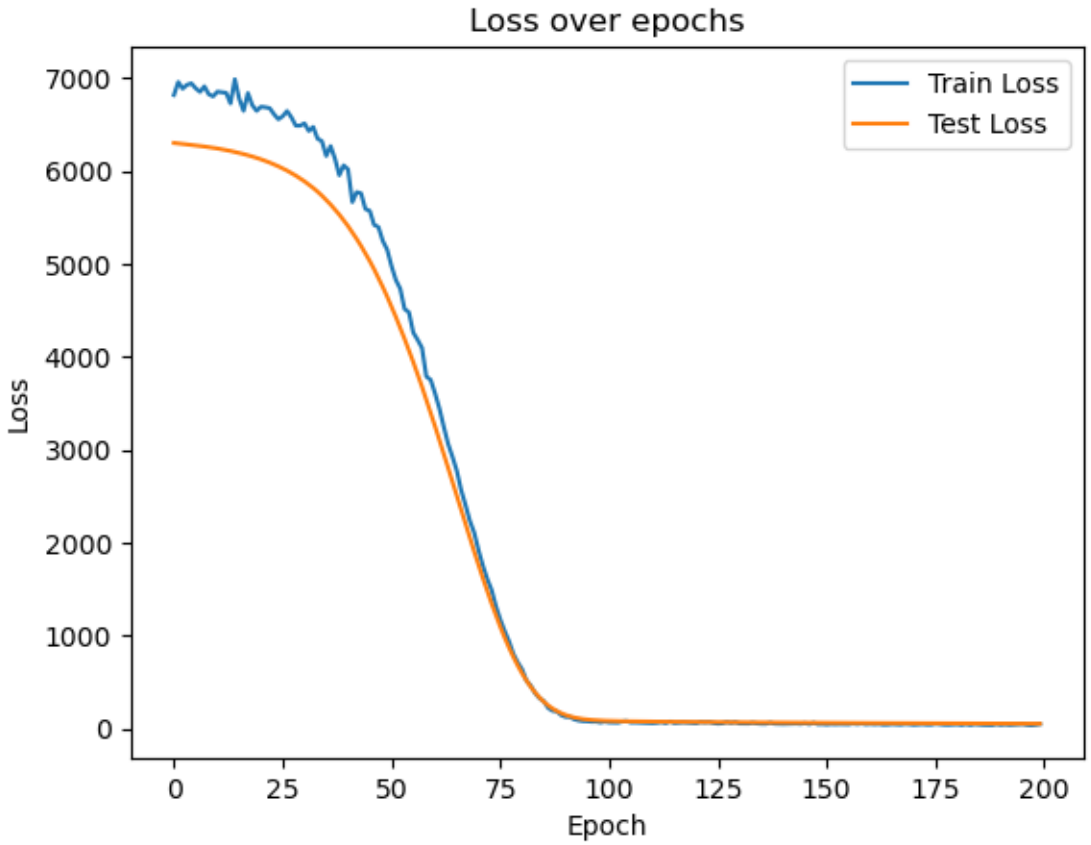
- Input Layer: 5 nơ-ron.
- Hidden Layer 1: 32 nơ-ron, ReLU .- • Hidden Layer 2: 16 nơ-ron, ReL U- • Hidden Layer 3: 8 nơ-ron, Re L- • Output Layer: 1 nơ-ron (Sco re). Huấn luyện với 200 epochs, batch size 16, chia 80% train / 20% test, dùng early stopping (patience=20). Vẽ biểu đồ loss, sau đó đề xuất một kiến trúc ANN khác (thay đổi số nơ-ron hoặc tầng) để cải thiện dự đoán, giải thích lý do dựa trên đặc điểm dữ liệu giátimizer).

[158]:
```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Load lại dữ liệu
df = pd.read_csv("customer_dataa.csv")

# Drop Score bị NaN (vì đây là label)
df = df.dropna(subset=["Score"])

# Fill NaN trong features bằng median
for col in ["Attendance (%)", "Study_Hours"]:
    df[col].fillna(df[col].median(), inplace=True)

# One-hot encode Course
df = pd.get_dummies(df, columns=["Course"])

# Chọn features
features = df[["Attendance (%)", "Study_Hours", "Course_Chemistry",␣
 ↪"Course_Math", "Course_Physics"]]
labels = df["Score"]

# Chuẩn hóa
scaler = MinMaxScaler()
features_scaled = scaler.fit_transform(features)

# Kiểm tra NaN
if np.isnan(features_scaled).any():
```

```
    raise ValueError(" Dữ liệu sau chuẩn hóa chứa NaN!")

# Chuẩn bị tensor
X = torch.tensor(features_scaled, dtype=torch.float32)
y = torch.tensor(labels.values, dtype=torch.float32).view(-1, 1)

# Chia tập train-test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

train_dataset = TensorDataset(X_train, y_train)
train_loader = DataLoader(train_dataset, batch_size=8, shuffle=True)

# Model ANN
class ANNModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.model = nn.Sequential(
            nn.Linear(5, 32),
            nn.ReLU(),
            nn.Linear(32, 16),
            nn.ReLU(),
            nn.Linear(16, 8),
            nn.ReLU(),
            nn.Linear(8, 1)
        )

    def forward(self, x):
        return self.model(x)

model = ANNModel()

# Initialize weights
def init_weights(m):
    if isinstance(m, nn.Linear):
        nn.init.xavier_normal_(m.weight)
        nn.init.zeros_(m.bias)

model.apply(init_weights)

# Optimizer & Loss
optimizer = optim.Adam(model.parameters(), lr=0.001)
criterion = nn.MSELoss()

# Training loop
train_losses, test_losses = [], []
best_loss = float('inf')
```

```
patience, wait = 20, 0

for epoch in range(200):
    model.train()
    epoch_loss = 0
    for xb, yb in train_loader:
        optimizer.zero_grad()
        output = model(xb)
        if torch.isnan(output).any():
            raise ValueError("  NaN trong output!")
        loss = criterion(output, yb)
        loss.backward()
        optimizer.step()
        epoch_loss += loss.item()

    train_losses.append(epoch_loss / len(train_loader))

    model.eval()
    with torch.no_grad():
        test_pred = model(X_test)
        if torch.isnan(test_pred).any():
            raise ValueError("  NaN trong test output!")
        test_loss = criterion(test_pred, y_test)
        test_losses.append(test_loss.item())

    if test_loss < best_loss:
        best_loss = test_loss
        wait = 0
    else:
        wait += 1

    if wait >= patience:
        print(f"[ ] Early stopping tại epoch {epoch+1}")
        break

    if epoch % 10 == 0:
        print(f"[ ] Epoch {epoch+1} | Train Loss: {train_losses[-1]:.4f} | Test␣
 ↪Loss: {test_loss.item():.4f}")

# Plot loss
plt.plot(train_losses, label="Train Loss")
plt.plot(test_losses, label="Test Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.title("Loss over epochs")
plt.show()
```

```
[ ] Epoch 1 | Train Loss: 6818.6250 | Test Loss: 6302.6816
[ ] Epoch 11 | Train Loss: 6854.2397 | Test Loss: 6243.2427
[ ] Epoch 21 | Train Loss: 6692.4489 | Test Loss: 6126.7397
[ ] Epoch 31 | Train Loss: 6513.5099 | Test Loss: 5889.7368
[ ] Epoch 41 | Train Loss: 6019.5148 | Test Loss: 5408.5728
[ ] Epoch 51 | Train Loss: 4977.5812 | Test Loss: 4540.5830
[ ] Epoch 61 | Train Loss: 3607.9107 | Test Loss: 3253.4456
[ ] Epoch 71 | Train Loss: 1919.2236 | Test Loss: 1758.2389
[ ] Epoch 81 | Train Loss: 637.3600 | Test Loss: 593.7148
[ ] Epoch 91 | Train Loss: 124.8327 | Test Loss: 146.4025
[ ] Epoch 101 | Train Loss: 66.3723 | Test Loss: 82.2624
[ ] Epoch 111 | Train Loss: 58.4040 | Test Loss: 74.3653
[ ] Epoch 121 | Train Loss: 57.7045 | Test Loss: 70.2892
[ ] Epoch 131 | Train Loss: 54.6562 | Test Loss: 66.9898
[ ] Epoch 141 | Train Loss: 49.8868 | Test Loss: 63.9429
[ ] Epoch 151 | Train Loss: 44.0758 | Test Loss: 61.5450
[ ] Epoch 161 | Train Loss: 49.4901 | Test Loss: 59.2113
[ ] Epoch 171 | Train Loss: 40.6843 | Test Loss: 57.4639
[ ] Epoch 181 | Train Loss: 45.3682 | Test Loss: 55.7387
[ ] Epoch 191 | Train Loss: 39.3227 | Test Loss: 53.6280
```



5.1.4 Đánh giá mô hình bằng MSE và R^2 trên tập test. Nếu R^2 dưới 0.8, phân tích nguyên nhân sai lệch dự đoán dựa trên đặc trưng đầu vào, đề xuất một cách cải thiện mô hình dựa trên ngữ cảnh giáo dục (không chỉ dùng dropout hay thay đổi optimizer).

[160]:
```
from sklearn.metrics import mean_squared_error, r2_score

# Chuyển output sang NumPy
y_true = y_test.numpy()
y_pred = test_pred.numpy()

# Tính MSE và R²
mse = mean_squared_error(y_true, y_pred)
r2 = r2_score(y_true, y_pred)

print(f"[ ] Mean Squared Error (MSE): {mse:.4f}")
print(f"[ ] R-squared (R²): {r2:.4f}")
```

```
[ ] Mean Squared Error (MSE): 52.1059
[ ] R-squared (R²): -0.5226
```

Khi hệ số xác định R^2 thấp hơn 0.8, điều đó cho thấy mô hình chưa thể giải thích phần lớn phương sai của điểm số đầu ra. Nguyên nhân không chỉ đến từ kiến trúc mạng nơ-ron, mà còn nằm ở sự thiếu thông tin và chiều sâu trong các đặc trưng đầu vào.

Hiện tại, mô hình chỉ đang sử dụng ba yếu tố đầu vào chính: Attendance (%), Study_Hours và mã hóa one-hot cho Course. Mặc dù đây là những yếu tố cơ bản liên quan đến quá trình học tập, nhưng chúng chưa đủ để mô tả toàn diện hành vi học tập và khả năng cá nhân của sinh viên. Một sinh viên có thể học nhiều nhưng học sai cách, hoặc đi học đầy đủ nhưng không thực sự tập trung. Ngoài ra, các môn học có độ khó khác nhau nhưng việc one-hot encoding lại không phản ánh được điều đó, khiến mô hình không phân biệt được mức độ ảnh hưởng riêng biệt của từng môn.

Dưới góc nhìn giáo dục, để cải thiện mô hình dự đoán điểm số học tập, ta không nên chỉ dừng lại ở việc tinh chỉnh kiến trúc mạng nơ-ron hay tối ưu thuật toán học. Thay vào đó, cần bổ sung những đặc trưng đầu vào có giá trị thực tiễn và phản ánh đúng hành vi học tập của sinh viên. Đầu tiên, thay vì sử dụng mã hóa one-hot cho môn học (Course), ta có thể áp dụng phương pháp target encoding, trong đó mỗi môn học được biểu diễn bằng điểm trung bình của chính nó trong tập dữ liệu. Cách làm này cho phép mô hình nhận biết sự khác biệt về độ khó giữa các môn – điều mà one-hot encoding không thể hiện được. Bên cạnh đó, việc kết hợp Attendance (%) và Study_Hours thành một đặc trưng mới mang tên Interaction Score sẽ giúp mô hình hiểu rõ hơn mức độ cam kết học tập của sinh viên, vốn là yếu tố quan trọng ảnh hưởng đến kết quả học tập thực tế. Cuối cùng, ta có thể xem xét xây dựng các mô hình riêng biệt cho từng môn học thay vì gộp chung tất cả. Điều này giúp mô hình tập trung học từ những đặc trưng riêng biệt của từng môn, tránh bị "nhiễu" bởi sự đa dạng trong phân phối dữ liệu giữa các khóa học khác nhau. Những cải tiến này không chỉ làm mô hình học sâu hơn về mặt thống kê, mà còn khiến nó trở nên "thấu cảm" hơn với bối cảnh giáo dục thực tế.

[ ]:
```

```