**ÔN TẬP**

**Biểu đồ Boxplot**

import **matplotlib**.**pyplot** as **plt**

*# Dữ liệu điểm số*

data = [55, 58, 60, 62, 65, 67, 68, 70, 72, 75,

        78, 80, 82, 85, 90, 92, 95, 98, 100, 150]

*# Vẽ biểu đồ boxplot*

**plt**.**boxplot**(data)

*# Thêm tiêu đề và nhãn*

**plt**.**title**('Biểu đồ Boxplot của Điểm Số')

**plt**.**ylabel**('Điểm số')

**plt**.**xticks**([1], ['Điểm số'])

*# Hiển thị biểu đồ*

**plt**.**show**()

A diagram of a box plot

Description automatically generated

**Giải thích biểu đồ:**

QR (Interquartile Range) là một chỉ số thống kê dùng để đo độ phân tán của dữ liệu trong một tập hợp. Nó đại diện cho khoảng giữa của dữ liệu, nơi mà 50% giá trị nằm trong đó.

**1. Phân bố điểm số**

* Khoảng Q1 và Q3
  + Q1 (25%) : 68
  + Q3 (75%): 90
  + Khoảng giữa (IQR) = 90 – 68 = 22

**2. Giá trị trung vị (median)**

Giá trị trung vị trong biểu đồ trên là 83. Điều này cho thấy rằng một nửa số sinh viên có điểm dưới 82 và nửa còn lại có điểm trên 82

**3. Giá trị ngoại lai**

* Giá trị nhỏ nhất là 55 (trong khoảng Q1): Điều này cho thấy điểm thấp nhất của lớp là 55 điểm, điều này cho thấy giáo viên cần phải hỗ trợ một số bạn trong lớp để cải thiện điểm số của học sinh.
* Giá trị lớn nhất trong khoảng hợp lệ là 100 (dưới 123): Điều này cho thấy trong lớp vẫn có học sinh rất xuất sắc và đạt được số điểm rất cao, cho thấy cách dạy của giáo viên cũng có mang lại những hiệu quả rất tốt, và bạn nào được số điểm này thì giáo viên có thể bồi dưỡng để học nâng cao hoặc tham gia các kì thi học sinh giỏi.
* Giá trị ngoại lai là 150: Đây có thể là một trường hợp đặc biệt hoặc sai sót trong quá trình nhập dữ liệu vào bảng điểm, cần được giáo viên xem xét để hiểu rõ nguyên nhân và khắc phục nếu có sai sót.
* Giá trị Q1 là 68 cho thấy rằng có 25% học sinh trong lớp có điểm số thấp hơn 68. Điều này cho thấy rằng một phần tư số sinh viên có thành tích thấp hơn mức này
* Giá trị Q3 là 90 cho thấy rằng 75% học sinh trong lớp có thành tích suất sắc, có thể là những học sinh cần được bồi dưỡng để tham gia kì thi học sinh giỏi
* Whisker từ 55 đến 68 cho thấy rằng có sinh viên đạt điểm ở mức này, whisker từ 90 trở lên cho thấy có những sinh viên nằm ở mức trên 90.

**4. Khả năng phân tích**

Sự phân tán: Với IQR = 22 điều này cho thấy có sự đa dạng nhất định trong thành tích của học sinh. Khoảng từ 68 đến 90 cho thấy hầu hết các sinh viên có điểm số tương đối tốt, cho thấy lớp này có chất lượng học tập khá cao.

**5. Kết luận:**

Diễn giải tổng quan:

* Có 50% số lượng sinh viên đạt điểm trong khoảng từ 68 đến 90(Ô hình chữ nhật mà ta thấy trên biểu đồ), điều này cho thấy lớp học có thành tích tốt.
* Giá trị ngoại lai 150 có thể là một trường hợp đặc biệt hoặc sai sót, cần tìm hiểu rõ để biết nguyên nhân.
* Giá trị trung vị là 82 cho thấy rằng một nửa số sinh viên đạt điểm trên mức này, và có thể khuyến nghị rằng phương pháp giảng dạy cần điều chỉnh để giúp đỡ những sinh viên có điểm thấp hơn.

**6. Khuyến nghị**

* **Điều tra thêm**: Nên xem xét kỹ lưỡng điểm số 150 để đảm bảo không có sai sót.
* **Chương trình hỗ trợ**: Cần có chương trình bổ trợ cho những sinh viên có điểm thấp hơn 68 để cải thiện thành tích.

**Biểu đồ cột**

A graph with different colored bars

Description automatically generated

**Biện luận biểu đồ**

**1. Mô tả biểu đồ**

Biểu đồ cột này thể hiện số lượng học sinh trong lớp theo các xếp loại từ F là thấp nhất đến A là cao nhất. Mỗi cột đại diện cho số lượng sinh viên đạt được theo từng xếp loại. Trục dọc bên trái là trục thể hiện số lượng sinh viên cho từng xếp loại, trục ngang là cái xếp loại điểm

**2. Phân tích từng xếp loại:**

* Xếp loại B+: Cột B+ là cột cao nhất, thể hiện rằng 20 sinh viên đạt xếp loại B+, cho thấy rằng đây là xếp loại phổ biến nhất trong lớp. Điều này phản ánh rằng phần lớn sinh viên trong lớp đạt loại giỏi (khoảng 28%)
* Xếp loại B: Số lượng sinh viên đạt cũng khá cao với 15 sinh viên, cho thấy đây là một nhóm lớn sinh viên có thành tích tốt, xếp sau tỉ lệ sinh viên đoạt loại giỏi, nhóm khá này chiếm khoảng 21%. Điều này phản ánh số lượng sinh viên học khá cũng rất nhiều, cố vấn học tập có thể hỗ trợ nhóm sinh viên này để có thể tiếp tục nâng điểm lên thành loại giỏi trong tương lai.
* Xếp loại C+: Có 9 học sinh đạt xếp loại C+, điều này cho thấy rằng nhóm này vẫn còn nhiều cơ hội để cải thiện. Nhóm này chiếm khoảng 12%
* Xếp loại A: Chỉ có 7 học sinh đạt xếp loại A, cho thấy rằng việc đạt điểm cao là điều không dễ dàng và chỉ có một số ít học sinh xuất sắc. Nhóm này chỉ chiếm khoảng 10%.
* Xếp loại D và F: Số lượng học sinh ở các xếp loại D+ (3), D (2), và F (3) tương đối thấp, cho thấy rằng có một số học sinh gặp khó khăn nhưng không nhiều. Nhóm này chiếm 11%

**3. Nhận diện xu hướng:**

Nhận thấy rằng số lượng sinh viên có thành tích học tập tốt cũng rất đông, tuy nhiên vẫn còn một số sinh viên còn điểm khá thấp, đòi hỏi cố vấn học tập phải hỗ trợ, khuyến khích các bạn sinh viên này phải nổ lực hơn trong việc tự học để không bị rớt môn. Kết quả tổng kết cũng cho thấy để đạt được loại A của môn này khá khó, hầu hết các bạn chỉ có thể đạt B hoặc B+. Tuy nhiên điều này cũng cho thấy khả năng học của các bạn sinh viên thuộc lớp này khá tốt.

**Biểu đồ tròn**

A green circle with a red triangle

Description automatically generated

**1. Mô tả biểu đồ**

Biểu đồ tròn này thể hiện tỷ lệ sinh viên đậu và rớt dựa trên tổng kết điểm. Trong đó, tỷ lệ sinh viên đậu chiếm 95.8% được thể hiện bằng màu xanh lục và tỷ lệ rớt chỉ chiếm 4.2% được thể hiện bằng màu đỏ.

**2. Phân tích tỉ lệ đậu**

* Tỷ lệ đậu cao: "Với 95.8% sinh viên đậu, chúng ta có thể thấy rằng phần lớn sinh viên trong lớp đã đạt yêu cầu học tập. Đây là một kết quả rất tích cực, cho thấy rằng phương pháp giảng dạy hoặc chương trình học đã mang lại hiệu quả cao."
* Ý nghĩa của tỷ lệ đậu: "Tỷ lệ đậu cao không chỉ phản ánh sự nỗ lực của sinh viên mà còn cho thấy sự hỗ trợ và hướng dẫn từ giáo viên có thể đã giúp sinh viên vượt qua kỳ thi."

**3. Phân tích tỉ lệ rớt**

* Tỷ lệ rớt thấp: "Tỷ lệ rớt chỉ 4.2% là một con số tương đối thấp, điều này cho thấy rằng rất ít sinh viên gặp khó khăn trong việc đạt yêu cầu học tập."
* Tìm hiểu nguyên nhân: "Mặc dù tỷ lệ rớt là nhỏ, nhưng nhà trường vẫn cần xem xét lý do tại sao một số sinh viên không đạt yêu cầu. Có thể có các yếu tố như vấn đề cá nhân, sự thiếu chú ý trong học tập, hoặc khó khăn trong việc nắm bắt kiến thức."

**4. Nhận diện xu hướng**

"Xu hướng tổng thể cho thấy rằng lớp học có một môi trường học tập tích cực, với sự hỗ trợ và động viên từ cả giáo viên và bạn bè."

**5. Kết luận và khuyến nghị**

* Tóm tắt: "Tóm lại, biểu đồ tròn cho thấy rằng phần lớn sinh viên đã thành công trong việc đạt yêu cầu học tập, với tỷ lệ đậu cao là 95.8%."
* Khuyến nghị: "Mặc dù tỷ lệ rớt rất thấp, nhà trường nên tiếp tục theo dõi và hỗ trợ những sinh viên này để đảm bảo rằng họ có thể cải thiện và không bị bỏ lại phía sau. Các chương trình hỗ trợ bổ sung, như tư vấn học tập hoặc các lớp ôn tập, có thể giúp những sinh viên này đạt được kết quả tốt hơn trong tương lai."

**Biểu đồ line + point**

**1. Mô tả biểu đồ**

Biểu đồ này hiển thị sự biến thiên điểm số của một sinh viên theo từng lần kiểm tra. Các điểm dữ liệu được thể hiện bằng các điểm (markers) và nối với nhau bằng đường thẳng.

**2. Phân tích các giá trị điểm**

Giá trị điểm lần lượt cho các lần kiểm tra là:

KTL1: 8.5

KTL2: 8.0

TBQT: 8.3

Thi: 7.3

Tổng kết: 7.8

**3. Phân tích từng lần kiểm tra**

* KTL1 (8.5): "Sinh viên có điểm cao nhất ở lần kiểm tra đầu tiên (KTL1) với 8.5. Điều này cho thấy rằng sinh viên có sự chuẩn bị tốt và khả năng nắm bắt kiến thức tốt trong giai đoạn đầu."
* KTL2 (8.0): "Điểm số giảm nhẹ ở KTL2 với 8.0. Mặc dù vẫn cao, nhưng sự giảm nhẹ này có thể chỉ ra rằng sinh viên gặp một chút khó khăn hơn trong lần kiểm tra này, hoặc nội dung kiểm tra có thể khó hơn."
* TBQT (8.3): "Điểm trung bình quá trình học tập (TBQT) ở mức 8.3 cho thấy sinh viên vẫn duy trì được mức độ học tập tốt. Sự phục hồi trong điểm số từ KTL2 cho thấy sinh viên đã điều chỉnh và cải thiện khả năng của mình."
* Thi (7.3): "Điểm thi giảm đáng kể xuống còn 7.3. Đây là điểm thấp nhất trong toàn bộ quá trình kiểm tra, cho thấy sinh viên có thể đã gặp khó khăn trong kỳ thi, hoặc có thể do áp lực thi cử."
* Tổng kết (7.8): "Điểm tổng kết là 7.8, cho thấy rằng mặc dù sinh viên đã có một số điểm số cao trong các kỳ kiểm tra trước, nhưng kết quả thi đã ảnh hưởng tiêu cực đến tổng kết. Tuy nhiên, tổng kết vẫn ở mức khá."

**4. Nhận diện xu hướng**

"Xu hướng tổng thể cho thấy rằng sinh viên có thể đã có một khởi đầu tốt nhưng gặp một số khó khăn trong kỳ thi. Mặc dù vẫn duy trì được kết quả tốt, nhưng có thể cần phải xem xét lại chiến lược học tập trước kỳ thi."

**5. Kết luận và khuyến nghị**

* Tóm tắt: "Tóm lại, biểu đồ này cho thấy sự biến thiên điểm số của sinh viên từ cao đến thấp, với điểm số cao nhất ở KTL1 và thấp nhất ở kỳ thi."
* Khuyến nghị: "Để cải thiện điểm thi, sinh viên có thể cần tăng cường ôn tập trước kỳ thi, tham gia các lớp bổ trợ hoặc tìm kiếm sự hỗ trợ từ giáo viên để hiểu rõ hơn các nội dung khó khăn."

**Kiểm tra dữ liệu có phân phối chuẩn hay không**

from **scipy**.**stats** import norm

*# Danh sách các cột có kiểu số để kiểm tra*

numeric\_columns = ['KTL1', 'KTL2', 'TBQT', 'Thi', 'Tongket']

*# Lặp qua từng cột và kiểm tra tính phân phối chuẩn*

for column in numeric\_columns:

**print**(f"\nKiểm tra cột: {column}")

*# Bước 1: Vẽ histogram*

**plt**.**figure**(figsize=(8, 6))

*# Vẽ histogram của dữ liệu*

    count, bins, ignored = **plt**.**hist**(df[column], bins=10, density=True, edgecolor='k', alpha=0.7, color='skyblue')

*# Tính toán trung bình và độ lệch chuẩn của cột*

    mean = **np**.**mean**(df[column])

    std = **np**.**std**(df[column])

*# Vẽ đường cong phân phối chuẩn*

    xmin, xmax = **plt**.**xlim**()  *# Lấy giới hạn của trục x hiện tại*

    x = **np**.**linspace**(xmin, xmax, 100)  *# Tạo 100 điểm giữa xmin và xmax*

    p = norm.**pdf**(x, mean, std)  *# Tính mật độ xác suất cho từng điểm x*

**plt**.**plot**(x, p, 'r', linewidth=2)  *# Vẽ đường màu đỏ cho đường phân phối chuẩn*

*# Thêm tiêu đề và nhãn cho biểu đồ*

**plt**.**title**(f'Histogram and Normal Curve of {column}')

**plt**.**xlabel**('Value')

**plt**.**ylabel**('Density')

*# Hiển thị biểu đồ*

**plt**.**show**()

*# Bước 2: Vẽ Q-Q plot*

**plt**.**figure**(figsize=(8, 6))

**stats**.**probplot**(df[column], dist="norm", plot=**plt**)

**plt**.**title**(f'Q-Q Plot of {column}')

**plt**.**show**()

*# Bước 3: Kiểm định Shapiro-Wilk Test*

    stat, p\_value = **stats**.**shapiro**(df[column])

**print**(f'Shapiro-Wilk Test for {column}: Statistics={stat:.3f}, p-value={p\_value:.3f}')

    if p\_value > 0.05:

**print**(f"Dữ liệu cột {column} có phân phối chuẩn")

    else:

**print**(f"Dữ liệu cột {column} không có phân phối chuẩn")

*# Bước 4: Kiểm định Kolmogorov-Smirnov Test, ddof=1 khi muốn ước lượng trên tổng thể, ddof=0 khi không cần ước lượng trên tồng thể*

    stat\_ks, p\_value\_ks = **stats**.**kstest**(df[column], 'norm', args=(**np**.**mean**(df[column]), **np**.**std**(df[column], ddof=1)))

**print**(f'Kolmogorov-Smirnov Test for {column}: Statistics={stat\_ks:.3f}, p-value={p\_value\_ks:.3f}')

    if p\_value\_ks > 0.05:

**print**(f"Dữ liệu cột {column} có phân phối chuẩn (theo KS Test)")

    else:

**print**(f"Dữ liệu cột {column} không có phân phối chuẩn (theo KS Test)")

**1. Nhập thư viện**

from **scipy**.**stats** import norm

Nhập hàm thư viện scipy.stats để tính toán phân phối chuẩn và thực hiện các kiểm định thống kê.

**2. Danh sách cột số**

numeric\_columns = ['KTL1', 'KTL2', 'TBQT', 'Thi', 'Tongket']

**3. Vòng lặp duyệt qua từng cột**

for column in numeric\_columns:

**print**(f"\nKiểm tra cột: {column}")

Dòng này chỉ in ra để biết là sắp kiểm tra cột nào có phân phối chuẩn hay không.

**4. Vẽ biểu đồ histogram**

Chiều ngang là 8, chiều dài là 6 và tính theo đơn vị inch

**plt**.**figure**(figsize=(8, 6))

*# Vẽ histogram của dữ liệu*

    count, bins, ignored = **plt**.**hist**(df[column], bins=10, density=True, edgecolor='k', alpha=0.7, color='skyblue')

Các tham số:

bins = 10 : Số lượng cột trong histogram

density = True: Biểu diễn mật độ thay vì tần xuất

**5. Tính Toán Trung Bình và Độ Lệch Chuẩn**

    mean = **np**.**mean**(df[column])

    std = **np**.**std**(df[column])

Hai giá trị này được tính để vẽ đường phân phối chuẩn

**6. Vẽ Đường Phân Phối Chuẩn**

xmin, xmax = **plt**.**xlim**()  *# Lấy giới hạn của trục x hiện tại*

    x = **np**.**linspace**(xmin, xmax, 100)  *# Tạo 100 điểm giữa xmin và xmax*

    p = norm.**pdf**(x, mean, std)  *# Tính mật độ xác suất cho từng điểm x*

**plt**.**plot**(x, p, 'r', linewidth=2)  *# Vẽ đường màu đỏ cho đường phân phối chuẩn*

**6.1. xmin, xmax = plt.xlim()**

* Mục đích: Lấy giới hạn hiện tại của trục x trong biểu đồ.
* Cách hoạt động:
  + Hàm plt.xlim() trả về hai giá trị: giới hạn thấp nhất (xmin) và giới hạn cao nhất (xmax) của trục x. Điều này rất hữu ích để đảm bảo rằng đường phân phối chuẩn mà bạn sẽ vẽ sẽ nằm trong cùng một phạm vi với dữ liệu histogram.

**6.2. x = np.linspace(xmin, xmax, 100)**

* Mục đích: Tạo ra một dãy các giá trị x để tính toán mật độ xác suất.
* Cách hoạt động:
  + np.linspace(xmin, xmax, 100) sẽ tạo ra 100 điểm số đều giữa xmin và xmax.
  + Điều này có nghĩa là bạn sẽ có 100 giá trị x mà bạn sẽ sử dụng để tính toán mật độ xác suất cho đường phân phối chuẩn. Sự phân bố đều giúp đường curve trở nên mượt mà và chính xác hơn.

**6.3. p = norm.pdf(x, mean, std)**

* Mục đích: Tính toán mật độ xác suất cho từng giá trị x.
* Cách hoạt động:
  + norm.pdf(x, mean, std) tính toán giá trị mật độ xác suất (Probability Density Function - PDF) cho từng điểm trong mảng x dựa trên trung bình (mean) và độ lệch chuẩn (std) đã được tính trước đó.
  + Kết quả là một mảng p, chứa các giá trị mật độ xác suất tương ứng với các giá trị x. Những giá trị này sẽ xác định độ cao của đường phân phối chuẩn mà bạn sẽ vẽ.

**6.4. plt.plot(x, p, 'r', linewidth=2)**

* Mục đích: Vẽ đường phân phối chuẩn lên biểu đồ.
* Cách hoạt động:
  + plt.plot(x, p, 'r', linewidth=2) sẽ vẽ đường curve bằng các điểm x và các giá trị mật độ xác suất p.
  + Tham số 'r' chỉ định màu đỏ cho đường curve.
  + linewidth=2 xác định độ dày của đường vẽ. Điều này giúp cho đường phân phối chuẩn trở nên dễ nhìn hơn trên biểu đồ.

**7. Tiêu đề và nhãn**

Biểu đồ tần xuất và đường cong bình thường(đường phân phối chuẩn)

Chữ f trong dòng đầu tiên là F-string cho phép chèn cái column vào mà khỏi cần ghi ra.

**plt**.**title**(f'Histogram and Normal Curve of {column}')

**plt**.**xlabel**('Value')

**plt**.**ylabel**('Density') #là mật độ

**8. Vẽ QQ-Plot**

*# Bước 2: Vẽ Q-Q plot*

**plt**.**figure**(figsize=(8, 6))

**stats**.**probplot**(df[column], dist="norm", plot=**plt**)

**plt**.**title**(f'Q-Q Plot of {column}')

**plt**.**show**()

**1. plt.figure(figsize=(8, 6))**

* **Mục đích**: Tạo một biểu đồ mới với kích thước cụ thể.
* **Cách hoạt động**:
  + figsize=(8, 6) chỉ định kích thước của biểu đồ. Cụ thể:
    - **8 inches**: chiều ngang
    - **6 inches**: chiều dọc
  + Kích thước này giúp đảm bảo biểu đồ có đủ không gian để hiển thị rõ ràng các thành phần mà không bị chồng chéo hoặc quá nhỏ.

**2. stats.probplot(df[column], dist="norm", plot=plt)**

* **Mục đích**: Tạo QQ-plot để kiểm tra tính phân phối chuẩn của dữ liệu trong cột.
* **Cách hoạt động**:
  + stats.probplot là một hàm trong scipy.stats dùng để vẽ QQ-plot.
  + Tham số df[column]: Chỉ định dữ liệu mà bạn muốn kiểm tra. df[column] lấy dữ liệu từ cột cụ thể trong DataFrame df.
  + Tham số dist="norm": Chỉ định rằng bạn đang so sánh với phân phối chuẩn (normal distribution).
  + Tham số plot=plt: Chỉ định rằng QQ-plot sẽ được vẽ lên biểu đồ hiện tại bằng matplotlib (plt).

**3. plt.title(f'Q-Q Plot of {column}')**

* **Mục đích**: Thêm tiêu đề cho biểu đồ.
* **Cách hoạt động**:
  + f'Q-Q Plot of {column}': Đây là một f-string, giúp chèn tên cột vào tiêu đề. Nếu column có giá trị là 'KTL1', tiêu đề sẽ trở thành 'Q-Q Plot of KTL1'.

**4. plt.show()**

* **Mục đích**: Hiển thị biểu đồ.
* **Cách hoạt động**:
  + Hàm plt.show() sẽ hiển thị tất cả các biểu đồ đã được tạo ra trong phiên làm việc hiện tại. Đây là bước cuối cùng để người dùng có thể nhìn thấy QQ-plot mà bạn vừa tạo.

**9. Kiểm Định Shapiro-Wilk**

|  |  |
| --- | --- |
| Theo Guilford | Theo Roundtree |
| **A purple and white rectangular table with text  Description automatically generated with medium confidence** |  |

*# Bước 3: Kiểm định Shapiro-Wilk Test*

    stat, p\_value = **stats**.**shapiro**(df[column]) Statistics(thống kê)

**print**(f'Shapiro-Wilk Test for {column}: Statistics={stat:.3f}, p-value={p\_value:.3f}')

#stat là hệ số tương quan

    if p\_value > 0.05: #Đây là giá trị thống kê từ kiểm định Shapiro-Wilk

**print**(f"Dữ liệu cột {column} có phân phối chuẩn")

    else:

**print**(f"Dữ liệu cột {column} không có phân phối chuẩn")

**1. stat, p\_value = stats.shapiro(df[column])**

* **Mục đích**: Thực hiện kiểm định Shapiro-Wilk để kiểm tra xem dữ liệu có phân phối chuẩn hay không.
* **Cách hoạt động**:
  + stats.shapiro() là một hàm trong scipy.stats dùng để thực hiện kiểm định Shapiro-Wilk.
  + df[column]: Chỉ định cột dữ liệu từ DataFrame mà bạn muốn kiểm tra.
  + Kết quả của hàm bao gồm hai giá trị:
    - **stat**: Giá trị thống kê của kiểm định (test statistic).
    - **p\_value**: Giá trị p-value, cho biết xác suất để bác bỏ giả thuyết không (null hypothesis).

**2. print(f'Shapiro-Wilk Test for {column}: Statistics={stat:.3f}, p-value={p\_value:.3f}')**

* **Mục đích**: In ra kết quả của kiểm định Shapiro-Wilk.
* **Cách hoạt động**:
  + f'Shapiro-Wilk Test for {column}: ...': Đây là một f-string, cho phép bạn chèn giá trị của biến column vào chuỗi.
  + {stat:.3f} và {p\_value:.3f}: Định dạng kết quả thống kê và p-value với 3 chữ số thập phân.
  + Dòng này sẽ hiển thị thông tin về kiểm định, bao gồm giá trị thống kê và p-value.

**3. if p\_value > 0.05:**

* **Mục đích**: Kiểm tra giả thuyết thống kê dựa trên giá trị p-value.
* **Cách hoạt động**:
  + Ở đây, bạn đang kiểm tra xem p-value có lớn hơn 0.05 hay không.
  + Ngưỡng 0.05 là một tiêu chuẩn phổ biến để xác định ý nghĩa thống kê.

**4. print(f"Dữ liệu cột {column} có phân phối chuẩn")**

* **Mục đích**: Thông báo rằng dữ liệu có thể được xem là phân phối chuẩn.
* **Cách hoạt động**:
  + Nếu p-value lớn hơn 0.05, điều này có nghĩa là không đủ bằng chứng để bác bỏ giả thuyết không (nghĩa là dữ liệu có phân phối chuẩn).

**5. else: và print(f"Dữ liệu cột {column} không có phân phối chuẩn")**

* **Mục đích**: Thông báo rằng dữ liệu không có phân phối chuẩn.
* **Cách hoạt động**:
  + Nếu p-value nhỏ hơn hoặc bằng 0.05, điều này có nghĩa là có đủ bằng chứng để bác bỏ giả thuyết không, tức là dữ liệu không tuân theo phân phối chuẩn.

**10. Kiểm Định Kolmogorov-Smirnov**

*# Bước 4: Kiểm định Kolmogorov-Smirnov Test, ddof=1 khi muốn ước lượng trên tổng thể, ddof=0 khi không cần ước lượng trên tồng thể*

    stat\_ks, p\_value\_ks = **stats**.**kstest**(df[column], 'norm', args=(**np**.**mean**(df[column]), **np**.**std**(df[column], ddof=1)))

**print**(f'Kolmogorov-Smirnov Test for {column}: Statistics={stat\_ks:.3f}, p-value={p\_value\_ks:.3f}')

    if p\_value\_ks > 0.05:

**print**(f"Dữ liệu cột {column} có phân phối chuẩn (theo KS Test)")

    else:

**print**(f"Dữ liệu cột {column} không có phân phối chuẩn (theo KS Test)")

Khi nào cần sử dụng Shapiro-Wilk Test và Kolmogorov-Smirnov Test

**1. Shapiro-Wilk Test (S-W):**

* **Mục đích**: Đặc biệt được thiết kế để kiểm tra tính phân phối chuẩn.
* **Tình huống sử dụng**:
  + Thích hợp cho mẫu nhỏ (thường dưới 50, nhưng có thể dùng cho mẫu đến 2000).
  + Dữ liệu cần kiểm tra phân phối chuẩn một cách trực tiếp.
* **Điểm mạnh**:
  + Hiệu quả và nhạy cảm hơn với sự khác biệt nhỏ từ phân phối chuẩn.

**2. Kolmogorov-Smirnov Test (K-S):**

* **Mục đích**: Kiểm tra xem một mẫu dữ liệu có phân phối giống một phân phối cụ thể (như phân phối chuẩn) hay không.
* **Tình huống sử dụng**:
  + Có thể được sử dụng cho mẫu lớn hơn (trên 20).
  + Thích hợp khi bạn muốn so sánh một mẫu với một phân phối cụ thể hoặc so sánh hai mẫu với nhau.
* **Điểm mạnh**:
  + Có thể được áp dụng cho bất kỳ phân phối nào, không chỉ phân phối chuẩn.

**Tóm lại:**

* **Shapiro-Wilk Test**: Sử dụng khi bạn muốn kiểm tra tính phân phối chuẩn, đặc biệt là với mẫu nhỏ.
* **Kolmogorov-Smirnov Test**: Sử dụng khi bạn muốn so sánh một mẫu với một phân phối cụ thể (có thể là phân phối chuẩn hoặc khác) hoặc so sánh hai mẫu với nhau, thích hợp cho mẫu lớn hơn.

= > dữ liệu có 72 sinh viên thì nên sử dụng Shapiro-Wilk Test

**Chi bình phương**

def **diemChu**(diem):

  if diem < 4.0:

    return "F"

  if diem < 5.0:

    return "D"

  if diem < 5.5:

    return "D+"

  if diem < 6.5:

    return "C"

  if diem < 7.0:

    return "C+"

  if diem < 8.0:

    return "B"

  if diem < 9.0:

    return "B+"

  else:

    return "A"

df['TongketDiemChu'] = df['Tongket'].**apply**(**diemChu**)

**print**(df)

IdSV Gioitinh Malop KTL1 KTL2 TBQT Thi Tongket TongketDiemChu

0 1 Nam DA20TTA 8.5 8.0 8.3 7.3 7.8 B

1 2 Nam DA20TTA 6.5 6.8 6.7 6.8 6.8 C+

2 3 Nam DA20TTA 7.0 5.8 6.4 5.9 6.2 C

3 4 Nữ DA20TTA 10.0 10.0 10.0 8.0 9.0 A

4 5 Nam DA20TTA 10.0 8.0 9.0 8.0 8.5 B+

.. ... ... ... ... ... ... ... ... ...

67 68 Nam DA20TTB 7.5 8.0 7.8 7.1 7.5 B

68 69 Nam DA20TTB 9.5 6.8 8.2 7.1 7.7 B

69 70 Nam DA20TTB 5.5 7.0 6.3 6.0 6.2 C

70 71 Nam DA20TTB 8.5 6.8 7.7 2.5 5.1 D+

71 72 Nữ DA20TTB 9.5 7.5 8.5 9.1 8.8 B+

[72 rows x 9 columns]

**3. Bảng tổng hợp (bangtonghop)**

bangtonghop = **pd**.**crosstab**(index=df["TongketDiemChu"], columns=df["Malop"],margins=True)

bangtonghop

| **Malop** | **DA20TTA** | **DA20TTB** | **All** |
| --- | --- | --- | --- |
| **TongketDiemChu** |  |  |  |
| A | 2 | 5 | 7 |
| B | 6 | 9 | 15 |
| B+ | 8 | 12 | 20 |
| C | 6 | 7 | 13 |
| C+ | 4 | 5 | 9 |
| D | 2 | 0 | 2 |
| D+ | 1 | 2 | 3 |
| F | 0 | 3 | 3 |
| All | 29 | 43 | 72 |

 **pd.crosstab(...)**:

* Đây là hàm từ thư viện Pandas dùng để tạo bảng tổng hợp giữa hai (hoặc nhiều) biến. Bảng này sẽ cho bạn thấy số lượng (hoặc tần số) của các kết hợp giữa các giá trị của các biến đó.

 **index=df["TongketDiemChu"]**:

* Tham số index xác định biến nào sẽ được sử dụng làm hàng trong bảng tổng hợp. Trong trường hợp này, nó lấy giá trị từ cột TongketDiemChu trong DataFrame df. Mỗi giá trị duy nhất trong cột này sẽ tạo thành một hàng trong bảng.

 **columns=df["Malop"]**:

* Tham số columns xác định biến nào sẽ được sử dụng làm cột trong bảng tổng hợp. Ở đây, nó lấy giá trị từ cột Malop, tương ứng với từng lớp học. Mỗi giá trị duy nhất trong cột này sẽ tạo thành một cột trong bảng.

 **margins=True**:

* Tham số này cho phép thêm tổng số hàng và cột vào bảng tổng hợp. Cụ thể, nó sẽ thêm một hàng tổng cho mỗi cột và một cột tổng cho mỗi hàng, cho phép bạn thấy tổng số quan sát cho từng chữ điểm và tổng số cho từng lớp.

Sử dụng pd.crosstab, bảng tổng hợp được tạo ra để đếm số lượng sinh viên trong mỗi loại chữ điểm theo từng lớp (Malop). Bảng này cũng tính tổng cho mỗi hàng và mỗi cột.

Kết quả của bảng này sẽ là:

* Mỗi hàng tương ứng với một loại chữ điểm (A, B, B+, C, C+, D, D+, F).
* Mỗi cột tương ứng với một lớp học (DA20TTA, DA20TTB) và cột "All" cho tổng số.

**4. Bảng quan sát (bangquansat)**

bangquansat=bangtonghop.iloc[0:8,0:2]

bangquansat

| **Malop** | **DA20TTA** | **DA20TTB** |
| --- | --- | --- |
| **TongketDiemChu** |  |  |
| A | 2 | 5 |
| B | 6 | 9 |
| B+ | 8 | 12 |
| C | 6 | 7 |
| C+ | 4 | 5 |
| D | 2 | 0 |
| D+ | 1 | 2 |
| F | 0 | 3 |

Bảng bangquansat chỉ lấy 8 hàng đầu tiên từ bangtonghop, tức là số lượng sinh viên theo chữ điểm cho hai lớp DA20TTA và DA20TTB.

Lấy 8 dòng và lấy 2 cột (0 và 1)

**5. Kiểm định Chi-squared (chi2\_contingency)**

from **scipy**.**stats** import **chi2\_contingency**

**chi2\_contingency**(bangquansat)

 **Tham số đầu vào**:

* bangquansat: Đây là một DataFrame chứa bảng tần số (crosstab) mà bạn đã tạo ra trước đó. Nó chứa số lượng sinh viên trong các ô phân loại theo chữ điểm và lớp học.

 **Kết quả trả về**:

* Hàm chi2\_contingency sẽ trả về một tuple bao gồm:
  + **Giá trị thống kê Chi-squared**: Cho biết mức độ khác biệt giữa tần số quan sát và tần số mong đợi.
  + **Giá trị p**: Xác suất để nhận được giá trị thống kê này (hoặc lớn hơn) nếu giả thuyết không đúng.
  + **Số bậc tự do (degrees of freedom)**: Thể hiện số lượng thông tin độc lập trong phân tích.
  + **Tần số mong đợi**: Mảng chứa các giá trị tần số mong đợi cho từng ô trong bảng.

Chi2ContingencyResult(statistic=5.70038333759264, pvalue=0.575136608827536, dof=7, expected\_freq=array([[ 2.81944444, 4.18055556],

[ 6.04166667, 8.95833333],

[ 8.05555556, 11.94444444],

[ 5.23611111, 7.76388889],

[ 3.625 , 5.375 ],

[ 0.80555556, 1.19444444],

[ 1.20833333, 1.79166667],

[ 1.20833333, 1.79166667]]))

Dùng hàm chi2\_contingency từ thư viện scipy.stats để kiểm tra xem có sự khác biệt đáng kể về phân phối chữ điểm giữa hai lớp hay không.

* **Kết quả**:
  + statistic: Giá trị thống kê Chi-squared.
  + pvalue: Giá trị p để kiểm tra giả thuyết.
  + dof: Số bậc tự do.
  + expected\_freq: Tần số mong đợi cho từng ô trong bảng.

**6. So sánh với giá trị tới hạn Chi-squared**

chi\_alpha=**stats**.chi2.**ppf**(0.95, 7)

**print**(chi\_alpha)

14.067140449340169

 **0.95**: Đây là xác suất mà bạn muốn tính. Trong trường hợp này, bạn đang tìm giá trị ngưỡng tại mức ý nghĩa 95%, tức là bạn muốn tìm giá trị mà 95% diện tích dưới đường cong phân phối Chi-squared nằm bên trái.

 **7**: Đây là số bậc tự do (degrees of freedom). Trong trường hợp của bạn, số bậc tự do là 7, thường được tính theo công thức (số hàng−1)×(số cột−1)trong kiểm định Chi-squared.

**Cách tính chi bình phương(bằng công thức)**

| **Malop** | **DA20TTA** | **DA20TTB** | **All** |
| --- | --- | --- | --- |
| **TongketDiemChu** |  |  |  |
| A | 2 | 5 | 7 |
| B | 6 | 9 | 15 |
| B+ | 8 | 12 | 20 |
| C | 6 | 7 | 13 |
| C+ | 4 | 5 | 9 |
| D | 2 | 0 | 2 |
| D+ | 1 | 2 | 3 |
| F | 0 | 3 | 3 |
| All | 29 | 43 | 72 |

**Bước 1: Tính tần số mong đợi (Expected Frequency)**

Tần số mong đợi cho mỗi ô được tính bằng công thức:

A black text on a white background

Description automatically generated

Tính tần số mong đợi cho ô dòng A cột DA20TTA

EA,DA22TTA = (7x29)/72 = 2,8194444444

Tính tần số mong đợi cho ô dòng A cột DA22TTB

EA,DA22TTB = (7x43)/72 = 4,18055…..

Tương tự cho các ô còn lại

**Bước 2: Tính giá trị thống kê Chi-squared(stat)**

([[ 2.81944444, 4.18055556],

[ 6.04166667, 8.95833333],

[ 8.05555556, 11.94444444],

[ 5.23611111, 7.76388889],

[ 3.625 , 5.375 ],

[ 0.80555556, 1.19444444],

[ 1.20833333, 1.79166667],

[ 1.20833333, 1.79166667]]))

A mathematical equation with numbers and symbols

Description automatically generated

Tính cho ô A ở lớp DA20TTA

X2A = (2-2,81944444)2/2,81944444 = 0,2381…

Tính cho tất cả các ô và cộng lại thì ta được giá trị thống kê Chi-Square bằng statistic=5.70038333759264

**Bước 3: Tính giá trị p**

Được tính bằng máy tính

H0: Điểm chữ tổng kết của sinh viên thì không liên quan đến lớp mà sinh viên theo học

Biện luận:

Cách 1: Sau khi thực hiện kiểm định chi bình phương, ta có statistic=5.70038333759264 < chi alpha=14.067140449340169 ==> Chấp nhận H0. Tức là, Điểm chữ tổng kết của sinh viên thì không liên quan gì đến lớp mà sv đang theo học trong môn này.

Cách 2: Sau khi thực hiện kiểm định chi bình phương, ta có pvalue=0.575136608827536 > alpha=0.05 ==> Chấp nhận H0. Tức là, Điểm chữ tổng kết của sinh viên thì không liên quan gì đến lớp mà sv đang theo học trong môn này.

Tuy nhiên, số ô tần số mong đợi < 5 là 9/16 ô (>20%) nên kết luận này chỉ đúng trên mẫu khảo sát, chưa thể khẳng định trên tổng thể.

Như vậy, phát biểu "Điểm chữ tổng kết của sinh viên có liên quan đến việc sinh viên học lớp nào trong môn học này" là sai.

**Các loại biểu đồ**

**1. Biểu đồ cột (Bar Chart)**

* **Mô tả**: Dùng để so sánh các giá trị của các danh mục khác nhau.
* **Thư viện**: Matplotlib, Seaborn.
* **Tham số**:
  + x: Danh sách hoặc mảng các nhãn cho các cột.
  + height: Giá trị của các cột.
  + color: Màu sắc của cột.

import **matplotlib**.**pyplot** as **plt**

**2. Biểu đồ tròn (Pie Chart)**

* **Mô tả**: Dùng để thể hiện tỷ lệ phần trăm của các phần trong tổng thể.
* **Thư viện**: Matplotlib.
* **Tham số**:
  + x: Giá trị của các phần.
  + labels: Nhãn cho các phần.
  + autopct: Định dạng hiển thị tỷ lệ phần trăm.

**3. Biểu đồ đường (Line Chart)**

* **Mô tả**: Thể hiện sự thay đổi của một biến theo thời gian hoặc một biến liên tục khác.
* **Thư viện**: Matplotlib, Seaborn.
* **Tham số**:
  + x: Giá trị trục x (thời gian hoặc biến độc lập).
  + y: Giá trị trục y (biến phụ thuộc).
  + color: Màu sắc của đường.

Code kiem\_dinh\_gia\_thuyet + phan\_tich\_tuong\_quan

Kiểm định chi bình phương

df = pd.read\_excel('DL\_dienthiketthucmon.xlsx', sheet\_name='Sheet1')

df

| **IdSV** | **Gioitinh** | **Malop** | **KTL1** | **KTL2** | **TBQT** | **Thi** | **Tongket** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | Nam | DA20TTA | 8.5 | 8.0 | 8.3 | 7.3 | 7.8 |
| 1 | 2 | Nam | DA20TTA | 6.5 | 6.8 | 6.7 | 6.8 | 6.8 |
| 2 | 3 | Nam | DA20TTA | 7.0 | 5.8 | 6.4 | 5.9 | 6.2 |
| 3 | 4 | Nữ | DA20TTA | 10.0 | 10.0 | 10.0 | 8.0 | 9.0 |
| 4 | 5 | Nam | DA20TTA | 10.0 | 8.0 | 9.0 | 8.0 | 8.5 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 67 | 68 | Nam | DA20TTB | 7.5 | 8.0 | 7.8 | 7.1 | 7.5 |
| 68 | 69 | Nam | DA20TTB | 9.5 | 6.8 | 8.2 | 7.1 | 7.7 |
| 69 | 70 | Nam | DA20TTB | 5.5 | 7.0 | 6.3 | 6.0 | 6.2 |
| 70 | 71 | Nam | DA20TTB | 8.5 | 6.8 | 7.7 | 2.5 | 5.1 |
| 71 | 72 | Nữ | DA20TTB | 9.5 | 7.5 | 8.5 | 9.1 | 8.8 |

72 rows × 8 columns

**Câu 1:**

Có phát biểu cho rằng: "Điểm chữ tổng kết của sinh viên có liên quan đến việc sinh viên học lớp nào trong môn học này". Hãy thực hiện kiểm định Chi-bình phương để tra phát biểu trên là đúng hay sai

def **diemChu**(diem):

  if diem < 4.0:

    return "F"

  if diem < 5.0:

    return "D"

  if diem < 5.5:

    return "D+"

  if diem < 6.5:

    return "C"

  if diem < 7.0:

    return "C+"

  if diem < 8.0:

    return "B"

  if diem < 9.0:

    return "B+"

  else:

    return "A"

df['TongketDiemChu'] = df['Tongket'].**apply**(**diemChu**)

**print**(df)

IdSV Gioitinh Malop KTL1 KTL2 TBQT Thi Tongket TongketDiemChu

0 1 Nam DA20TTA 8.5 8.0 8.3 7.3 7.8 B

1 2 Nam DA20TTA 6.5 6.8 6.7 6.8 6.8 C+

2 3 Nam DA20TTA 7.0 5.8 6.4 5.9 6.2 C

3 4 Nữ DA20TTA 10.0 10.0 10.0 8.0 9.0 A

4 5 Nam DA20TTA 10.0 8.0 9.0 8.0 8.5 B+

.. ... ... ... ... ... ... ... ... ...

67 68 Nam DA20TTB 7.5 8.0 7.8 7.1 7.5 B

68 69 Nam DA20TTB 9.5 6.8 8.2 7.1 7.7 B

69 70 Nam DA20TTB 5.5 7.0 6.3 6.0 6.2 C

70 71 Nam DA20TTB 8.5 6.8 7.7 2.5 5.1 D+

71 72 Nữ DA20TTB 9.5 7.5 8.5 9.1 8.8 B+

[72 rows x 9 columns]

bangtonghop = **pd**.**crosstab**(index=df["TongketDiemChu"], columns=df["Malop"],margins=True)

bangtonghop

| **Malop** | **DA20TTA** | **DA20TTB** | **All** |
| --- | --- | --- | --- |
| **TongketDiemChu** |  |  |  |
| A | 2 | 5 | 7 |
| B | 6 | 9 | 15 |
| B+ | 8 | 12 | 20 |
| C | 6 | 7 | 13 |
| C+ | 4 | 5 | 9 |
| D | 2 | 0 | 2 |
| D+ | 1 | 2 | 3 |
| F | 0 | 3 | 3 |
| All | 29 | 43 | 72 |

bangquansat=bangtonghop.iloc[0:8,0:2]

bangquansat

| **Malop** | **DA20TTA** | **DA20TTB** |
| --- | --- | --- |
| **TongketDiemChu** |  |  |
| **A** | **2** | **5** |
| **B** | **6** | **9** |
| **B+** | **8** | **12** |
| **C** | **6** | **7** |
| **C+** | **4** | **5** |
| **D** | **2** | **0** |
| **D+** | **1** | **2** |
| **F** | **0** | **3** |

from **scipy**.**stats** import **chi2\_contingency**

**chi2\_contingency**(bangquansat)

Chi2ContingencyResult(statistic=5.70038333759264, pvalue=0.575136608827536, dof=7, expected\_freq=array([[ 2.81944444, 4.18055556],

[ 6.04166667, 8.95833333],

[ 8.05555556, 11.94444444],

[ 5.23611111, 7.76388889],

[ 3.625 , 5.375 ],

[ 0.80555556, 1.19444444],

[ 1.20833333, 1.79166667],

[ 1.20833333, 1.79166667]]))

chi\_alpha=**stats**.chi2.**ppf**(0.95, 7)

**print**(chi\_alpha)

14.067140449340169

H0: Điểm chữ tổng kết của sinh viên thì không liên quan đến lớp mà sinh viên theo học

Biện luận:

Cách 1: Sau khi thực hiện kiểm định chi bình phương, ta có statistic=5.70038333759264 < chi alpha=14.067140449340169 ==> Chấp nhận H0. Tức là, Điểm chữ tổng kết của sinh viên thì không liên quan gì đến lớp mà sv đang theo học trong môn này.

Cách 2: Sau khi thực hiện kiểm định chi bình phương, ta có pvalue=0.575136608827536 > alpha=0.05 ==> Chấp nhận H0. Tức là, Điểm chữ tổng kết của sinh viên thì không liên quan gì đến lớp mà sv đang theo học trong môn này.

Tuy nhiên, số ô tần số mong đợi < 5 là 9/16 ô (>20%) nên kết luận này chỉ đúng trên mẫu khảo sát, chưa thể khẳng định trên tổng thể.

Như vậy, phát biểu "Điểm chữ tổng kết của sinh viên có liên quan đến việc sinh viên học lớp nào trong môn học này" là sai.

**HỆ SỐ TƯƠNG QUAN**

from **scipy**.**stats** import norm

*# Danh sách các cột có kiểu số để kiểm tra*

numeric\_columns = ['KTL1', 'KTL2', 'TBQT', 'Thi', 'Tongket']

*# Lặp qua từng cột và kiểm tra tính phân phối chuẩn*

for column in numeric\_columns:

**print**(f"\nKiểm tra cột: {column}")

*# Bước 1: Vẽ histogram*

**plt**.**figure**(figsize=(8, 6))

*# Vẽ histogram của dữ liệu*

    count, bins, ignored = **plt**.**hist**(df[column], bins=10, density=True, edgecolor='k', alpha=0.7, color='skyblue')

*# Tính toán trung bình và độ lệch chuẩn của cột*

    mean = **np**.**mean**(df[column])

    std = **np**.**std**(df[column])

*# Vẽ đường cong phân phối chuẩn*

    xmin, xmax = **plt**.**xlim**()  *# Lấy giới hạn của trục x hiện tại*

    x = **np**.**linspace**(xmin, xmax, 100)  *# Tạo 100 điểm giữa xmin và xmax*

    p = norm.**pdf**(x, mean, std)  *# Tính mật độ xác suất cho từng điểm x*

**plt**.**plot**(x, p, 'r', linewidth=2)  *# Vẽ đường màu đỏ cho đường phân phối chuẩn*

*# Thêm tiêu đề và nhãn cho biểu đồ*

**plt**.**title**(f'Histogram and Normal Curve of {column}')

**plt**.**xlabel**('Value')

**plt**.**ylabel**('Density')

*# Hiển thị biểu đồ*

**plt**.**show**()

*# Bước 2: Vẽ Q-Q plot*

**plt**.**figure**(figsize=(8, 6))

**stats**.**probplot**(df[column], dist="norm", plot=**plt**)

**plt**.**title**(f'Q-Q Plot of {column}')

**plt**.**show**()

*# Bước 3: Kiểm định Shapiro-Wilk Test*

    stat, p\_value = **stats**.**shapiro**(df[column])

**print**(f'Shapiro-Wilk Test for {column}: Statistics={stat:.3f}, p-value={p\_value:.3f}')

    if p\_value > 0.05:

**print**(f"Dữ liệu cột {column} có phân phối chuẩn")

    else:

**print**(f"Dữ liệu cột {column} không có phân phối chuẩn")

*# Bước 4: Kiểm định Kolmogorov-Smirnov Test, ddof=1 khi muốn ước lượng trên tổng thể, ddof=0 khi không cần ước lượng trên tồng thể*

    stat\_ks, p\_value\_ks = **stats**.**kstest**(df[column], 'norm', args=(**np**.**mean**(df[column]), **np**.**std**(df[column], ddof=1)))

**print**(f'Kolmogorov-Smirnov Test for {column}: Statistics={stat\_ks:.3f}, p-value={p\_value\_ks:.3f}')

    if p\_value\_ks > 0.05:

**print**(f"Dữ liệu cột {column} có phân phối chuẩn (theo KS Test)")

    else:

**print**(f"Dữ liệu cột {column} không có phân phối chuẩn (theo KS Test)")

Kiểm tra cột: KTL1

A graph with a red line

Description automatically generated

A graph with blue dots and red line

Description automatically generated

Shapiro-Wilk Test for KTL1: Statistics=0.864, p-value=0.000

Dữ liệu cột KTL1 không có phân phối chuẩn

Kolmogorov-Smirnov Test for KTL1: Statistics=0.152, p-value=0.065

Dữ liệu cột KTL1 có phân phối chuẩn (theo KS Test)

Kiểm tra cột: KTL2

A graph with a red line

Description automatically generated

A graph of a graph

Description automatically generated with medium confidence

Shapiro-Wilk Test for KTL2: Statistics=0.955, p-value=0.011

Dữ liệu cột KTL2 không có phân phối chuẩn

Kolmogorov-Smirnov Test for KTL2: Statistics=0.147, p-value=0.079

Dữ liệu cột KTL2 có phân phối chuẩn (theo KS Test)

Kiểm tra cột: TBQT

A graph with a red line

Description automatically generated

A graph with blue dots and red lines

Description automatically generated

Shapiro-Wilk Test for TBQT: Statistics=0.931, p-value=0.001

Dữ liệu cột TBQT không có phân phối chuẩn

Kolmogorov-Smirnov Test for TBQT: Statistics=0.145, p-value=0.089

Dữ liệu cột TBQT có phân phối chuẩn (theo KS Test)

Kiểm tra cột: Thi

A graph with a red line

Description automatically generated

A graph with a red line and blue dots

Description automatically generated

Shapiro-Wilk Test for Thi: Statistics=0.938, p-value=0.002

Dữ liệu cột Thi không có phân phối chuẩn

Kolmogorov-Smirnov Test for Thi: Statistics=0.124, p-value=0.199

Dữ liệu cột Thi có phân phối chuẩn (theo KS Test)

Kiểm tra cột: Tongket

A graph with a red line

Description automatically generated

A graph with blue dots and red line

Description automatically generated

Shapiro-Wilk Test for Tongket: Statistics=0.951, p-value=0.007

Dữ liệu cột Tongket không có phân phối chuẩn

Kolmogorov-Smirnov Test for Tongket: Statistics=0.098, p-value=0.458

Dữ liệu cột Tongket có phân phối chuẩn (theo KS Test)

Giả sử kiểm định biến có phân phối chuẩn

 --> dùng Pearson

*# Kiểm định tính tương quan giữa KTL1 và KTL2*

correlation\_coefficient, p\_value = **stats**.**pearsonr**(df['KTL1'], df['KTL2'])

*# Hiển thị kết quả*

**print**(f'Hệ số tương quan Pearson giữa KTL1 và KTL2: {correlation\_coefficient:.2f}')

**print**(f'Giá trị p: {p\_value:.3f}')

*# Kiểm tra tính có ý nghĩa*

alpha = 0.05

if p\_value < alpha:

**print**("Có tương quan giữa KTL1 và KTL2 (bác bỏ H0)")

else:

**print**("Không có tương quan giữa KTL1 và KTL2 (không bác bỏ H0)")

Hệ số tương quan Pearson giữa KTL1 và KTL2: 0.26

Giá trị p: 0.026

Có tương quan giữa KTL1 và KTL2 (bác bỏ H0)

Giả sử kiểm định trên phân phối không chuẩn

from **scipy**.**stats** import **spearmanr**

*# Tính toán hệ số tương quan Spearman*

corr, p\_value = **spearmanr**(df['KTL1'], df['KTL2'])

**print**(f'Tương quan Spearman: {corr:.2f}, p-value: {p\_value:.3f}')

*# Kết luận*

if p\_value < 0.05:

**print**("Có mối tương quan đáng kể giữa KTL1 và KTL2 theo kiểm định Spearman.")

else:

**print**("Không có mối tương quan đáng kể giữa KTL1 và KTL2 theo kiểm định Spearman.")

Tương quan Spearman: 0.23, p-value: 0.057

Không có mối tương quan đáng kể giữa KTL1 và KTL2 theo kiểm định Spearman.

Vẽ bảng heatmap bảng tương quan

from **matplotlib**.**colors** import **ListedColormap**

*# Tính toán ma trận tương quan*

correlation\_matrix=df[['KTL1','KTL2','TBQT','Thi', 'Tongket' ]].**corr**()

**print**(correlation\_matrix)

*# Vẽ heatmap*

**plt**.**figure**(figsize=(10, 8))

*# Tạo bảng màu tùy chỉnh với màu trắng cho 0 và màu đỏ cho âm và dương*

cmap = **plt**.**cm**.Reds  *# Đỏ cho âm, trắng cho 0, đỏ cho dương*

**sns**.**heatmap**(correlation\_matrix, annot=True, fmt=".2f", cmap=cmap, square=True, cbar\_kws={"shrink": 1.0})

**plt**.**title**('Heatmap of Correlation Matrix')

**plt**.**show**()

KTL1 KTL2 TBQT Thi Tongket

KTL1 1.000000 0.261630 0.903563 0.558474 0.772921

KTL2 0.261630 1.000000 0.649520 0.342653 0.517387

TBQT 0.903563 0.649520 1.000000 0.595303 0.840773

Thi 0.558474 0.342653 0.595303 1.000000 0.935302

Tongket 0.772921 0.517387 0.840773 0.935302 1.000000

A screenshot of a grid

Description automatically generated

**HỒI QUY TUYẾN TÍNH BỘI**

import numpy as **np**

import **pandas** as **pd**

import **matplotlib**.**pyplot** as **plt**

from mpl\_toolkits.mplot3d import Axes3D

from **sklearn**.**linear\_model** import **LinearRegression**

*# Tạo dữ liệu*

data = {

    'Dien\_tich': [70, 85, 90, 60, 120, 100, 75, 110, 95, 105],

    'So\_phong\_ngu': [2, 3, 3, 2, 4, 3, 2, 4, 3, 3],

    'Khoang\_cach': [5, 3, 4, 8, 2, 3, 6, 4, 5, 2],

    'Gia\_nha': [150, 200, 220, 130, 300, 250, 180, 280, 240, 270]

}

df = **pd**.**DataFrame**(data)

*# Xác định biến độc lập và biến phụ thuộc*

X = df[['Dien\_tich', 'So\_phong\_ngu', 'Khoang\_cach']]

y = df['Gia\_nha']

*# Xây dựng mô hình hồi quy tuyến tính bội*

model = **LinearRegression**()

model.**fit**(X, y)

*# Tạo lưới điểm để vẽ bề mặt hồi quy*

x\_surf, y\_surf = **np**.meshgrid(**np**.linspace(X['Dien\_tich'].**min**(), X['Dien\_tich'].**max**(), 100),

**np**.linspace(X['So\_phong\_ngu'].**min**(), X['So\_phong\_ngu'].**max**(), 100))

z\_surf = model.**predict**(**np**.c\_[x\_surf.ravel(), y\_surf.ravel(), **np**.full\_like(x\_surf.ravel(), X['Khoang\_cach'].**mean**())])

z\_surf = z\_surf.reshape(x\_surf.shape)

*# Vẽ biểu đồ*

fig = **plt**.**figure**(figsize=(12, 8))

ax = fig.**add\_subplot**(111, projection='3d')

*# Vẽ các điểm dữ liệu*

ax.scatter(df['Dien\_tich'], df['So\_phong\_ngu'], df['Gia\_nha'], c='r', marker='o', alpha=0.6)

*# Vẽ bề mặt hồi quy*

ax.plot\_surface(x\_surf, y\_surf, z\_surf, color='b', alpha=0.3)

*# Gắn nhãn cho các trục*

ax.set\_xlabel('Diện tích (m²)')

ax.set\_ylabel('Số phòng ngủ')

ax.set\_zlabel('Giá nhà (triệu VND)')

**plt**.**title**('Hồi quy tuyến tính bội - Giá nhà')

**plt**.**show**()

A graph of a graph with numbers and lines

Description automatically generated with medium confidence

import numpy as **np**

import **pandas** as **pd**

import **matplotlib**.**pyplot** as **plt**

from **sklearn**.**linear\_model** import **LinearRegression**

*# Tạo dữ liệu*

data = {

    'Dien\_tich': [70, 85, 90, 60, 120, 100, 75, 110, 95, 105],

    'So\_phong\_ngu': [2, 3, 3, 2, 4, 3, 2, 4, 3, 3],

    'Khoang\_cach': [5, 3, 4, 8, 2, 3, 6, 4, 5, 2],

    'Gia\_nha': [150, 200, 220, 130, 300, 250, 180, 280, 240, 270]

}

df = **pd**.**DataFrame**(data)

*# Xác định biến độc lập và biến phụ thuộc*

X = df[['Dien\_tich', 'So\_phong\_ngu', 'Khoang\_cach']]

y = df['Gia\_nha']

*# Xây dựng mô hình hồi quy tuyến tính bội*

model = **LinearRegression**()

model.**fit**(X, y)

*# Tạo các biểu đồ phân tán với đường hồi quy cho từng biến độc lập*

fig, axes = **plt**.**subplots**(1, 3, figsize=(18, 6))

for i, feature in **enumerate**(X.columns):

    ax = axes[i]

    ax.scatter(df[feature], y, c='r', marker='o', alpha=0.6)

*# Tạo một đường hồi quy cho từng biến độc lập*

    X\_feature = df[[feature]]

    model\_feature = **LinearRegression**()

    model\_feature.**fit**(X\_feature, y)

    y\_pred = model\_feature.**predict**(X\_feature)

    ax.plot(df[feature], y\_pred, color='b')

    ax.set\_xlabel(feature)

    ax.set\_ylabel('Gia\_nha')

    ax.set\_title(f'Hồi quy tuyến tính - {feature}')

**plt**.**tight\_layout**()

**plt**.**show**()

A graph with a blue line

Description automatically generated

import numpy as **np**

import **pandas** as **pd**

import **matplotlib**.**pyplot** as **plt**

from **sklearn**.**linear\_model** import **LinearRegression**

*# Tạo dữ liệu*

data = {

    'Dien\_tich': [70, 85, 90, 60, 120, 100, 75, 110, 95, 105],

    'So\_phong\_ngu': [2, 3, 3, 2, 4, 3, 2, 4, 3, 3],

    'Khoang\_cach': [5, 3, 4, 8, 2, 3, 6, 4, 5, 2],

    'Gia\_nha': [150, 200, 220, 130, 300, 250, 180, 280, 240, 270]

}

df = **pd**.**DataFrame**(data)

*# Xác định biến độc lập và biến phụ thuộc*

X = df[['Dien\_tich', 'So\_phong\_ngu', 'Khoang\_cach']]

y = df['Gia\_nha']

*# Xây dựng mô hình hồi quy tuyến tính bội*

model = **LinearRegression**()

model.**fit**(X, y)

*# Tạo các biểu đồ phân tán với đường hồi quy cho từng biến độc lập*

**plt**.**figure**(figsize=(10, 6))

colors = ['r', 'g', 'b']

labels = ['Dien\_tich', 'So\_phong\_ngu', 'Khoang\_cach']

for i, feature in **enumerate**(X.columns):

*# Tạo một đường hồi quy cho từng biến độc lập*

    X\_feature = df[[feature]]

    model\_feature = **LinearRegression**()

    model\_feature.**fit**(X\_feature, y)

    y\_pred = model\_feature.**predict**(X\_feature)

*# Vẽ các điểm dữ liệu*

**plt**.**scatter**(df[feature], y, alpha=0.6, c=colors[i], label=f'Dữ liệu - {feature}')

*# Vẽ đường hồi quy*

**plt**.**plot**(df[feature], y\_pred, color=colors[i], linewidth=2, label=f'Hồi quy - {feature}')

**plt**.**xlabel**('Giá trị của biến độc lập')

**plt**.**ylabel**('Gia\_nha')

**plt**.**title**('Hồi quy tuyến tính bội')

**plt**.**legend**()

**plt**.**show**()

A graph with red and green dots

Description automatically generated

**CHỈNH HỢP, TỔ HỢP**

Chỉnh hợp

import **itertools**

A = "Xanh", "Vàng", "Đỏ"

result = **list**(**itertools**.**permutations**(A, 2))

**print**("Chỉnh hợp chập 2 của mảng A là: ")

count = 0

for e in result:

**print**(e)

    count+=1

**print**("Kết quả chỉnh hợp chập 2 của mảng A = ",count)

Chỉnh hợp chập 2 của mảng A là:

('Xanh', 'Vàng')

('Xanh', 'Đỏ')

('Vàng', 'Xanh')

('Vàng', 'Đỏ')

('Đỏ', 'Xanh')

('Đỏ', 'Vàng')

Kết quả chỉnh hợp chập 2 của mảng A = 6

Tổ hợp

import **itertools**

A = "Xanh", "Vàng", "Đỏ"

result = **list**(**itertools**.**combinations**(A, 2))

**print**("Chỉnh tổ chập 2 của mảng A là: ")

count = 0

for e in result:

**print**(e)

    count+=1

**print**("Kết quả tổ hợp chập 2 của mảng A = ",count)

Chỉnh tổ chập 2 của mảng A là:

('Xanh', 'Vàng')

('Xanh', 'Đỏ')

('Vàng', 'Đỏ')

Kết quả tổ hợp chập 2 của mảng A = 3

Chỉnh hợp lặp

import **itertools**

A = "1", "2", "3"

k=2

result = **itertools**.**product**(A,repeat=k)

for a **in** result:

**print**(a)

('1', '1')

('1', '2')

('1', '3')

('2', '1')

('2', '2')

('2', '3')

('3', '1')

('3', '2')

('3', '3')

Tổ hợp lặp

import **itertools**

A = "A", "B", "C"

result = **itertools**.**combinations\_with\_replacement**(A,2)

for a **in** result:

**print**(a)

('A', 'A')

('A', 'B')

('A', 'C')

('B', 'B')

('B', 'C')

('C', 'C')

Bài tập

1. Với năm chữ số 1;2;3;4;5 có thể thành lập được bao nhiêu số tự nhiên có 3 chữ số khác nhau

import **itertools**

A = 1, 2, 3, 4, 5

result = **itertools**.**permutations**(A, 3)

count = 0

for a **in** result:

**print**(a)

    count +=1

**print**("Có ",count, " số được tạo thành")

(1, 2, 3)

(1, 2, 4)

(1, 2, 5)

(1, 3, 2)

(1, 3, 4)

(1, 3, 5)

(1, 4, 2)

(1, 4, 3)

(1, 4, 5)

(1, 5, 2)

(1, 5, 3)

(1, 5, 4)

(2, 1, 3)

(2, 1, 4)

(2, 1, 5)

(2, 3, 1)

(2, 3, 4)

(2, 3, 5)

(2, 4, 1)

(2, 4, 3)

(2, 4, 5)

(2, 5, 1)

(2, 5, 3)

(2, 5, 4)

(3, 1, 2)

...

(5, 4, 1)

(5, 4, 2)

(5, 4, 3)

Có 60 số được tạo thành

2. Số cách xếp 5 nữ và 5 nam thành 1 hàng dọc, sao cho nữ và nam ngồi xem kẽ nhau

import **itertools**

import **math**

*# Danh sách nữ và nam*

A = ["nu1", "nu2", "nu3", "nu4", "nu5"]

B = ["nam1", "nam2", "nam3", "nam4", "nam5"]

ChinhHop\_A = **list**(**itertools**.**permutations**(A))

ChinhHop\_B = **list**(**itertools**.**permutations**(B))

valid\_arrangements = []

for female in ChinhHop\_A:

    for male in ChinhHop\_B:

        arrangement = []

        for f, m in **zip**(female, male):

            arrangement.**append**(f)

            arrangement.**append**(m)

        valid\_arrangements.**append**(arrangement)

for male in ChinhHop\_B:

    for female in ChinhHop\_A:

        arrangement = []

        for m, f in **zip**(male, female):

            arrangement.**append**(m)

            arrangement.**append**(f)

        valid\_arrangements.**append**(arrangement)

**print**(f"Số cách xếp hợp lệ: {**len**(valid\_arrangements)}")

**print**("Một số cách sắp xếp:")

for i in **range**(**min**(30, **len**(valid\_arrangements))):  *# In ra tối đa 5 cách sắp xếp*

**print**(valid\_arrangements[i])

Số cách xếp hợp lệ: 28800

Một số cách sắp xếp:

['nu1', 'nam1', 'nu2', 'nam2', 'nu3', 'nam3', 'nu4', 'nam4', 'nu5', 'nam5']

['nu1', 'nam1', 'nu2', 'nam2', 'nu3', 'nam3', 'nu4', 'nam5', 'nu5', 'nam4']

['nu1', 'nam1', 'nu2', 'nam2', 'nu3', 'nam4', 'nu4', 'nam3', 'nu5', 'nam5']

['nu1', 'nam1', 'nu2', 'nam2', 'nu3', 'nam4', 'nu4', 'nam5', 'nu5', 'nam3']

['nu1', 'nam1', 'nu2', 'nam2', 'nu3', 'nam5', 'nu4', 'nam3', 'nu5', 'nam4']

['nu1', 'nam1', 'nu2', 'nam2', 'nu3', 'nam5', 'nu4', 'nam4', 'nu5', 'nam3']

['nu1', 'nam1', 'nu2', 'nam3', 'nu3', 'nam2', 'nu4', 'nam4', 'nu5', 'nam5']

['nu1', 'nam1', 'nu2', 'nam3', 'nu3', 'nam2', 'nu4', 'nam5', 'nu5', 'nam4']

['nu1', 'nam1', 'nu2', 'nam3', 'nu3', 'nam4', 'nu4', 'nam2', 'nu5', 'nam5']

['nu1', 'nam1', 'nu2', 'nam3', 'nu3', 'nam4', 'nu4', 'nam5', 'nu5', 'nam2']

['nu1', 'nam1', 'nu2', 'nam3', 'nu3', 'nam5', 'nu4', 'nam2', 'nu5', 'nam4']

['nu1', 'nam1', 'nu2', 'nam3', 'nu3', 'nam5', 'nu4', 'nam4', 'nu5', 'nam2']

['nu1', 'nam1', 'nu2', 'nam4', 'nu3', 'nam2', 'nu4', 'nam3', 'nu5', 'nam5']

['nu1', 'nam1', 'nu2', 'nam4', 'nu3', 'nam2', 'nu4', 'nam5', 'nu5', 'nam3']

['nu1', 'nam1', 'nu2', 'nam4', 'nu3', 'nam3', 'nu4', 'nam2', 'nu5', 'nam5']

['nu1', 'nam1', 'nu2', 'nam4', 'nu3', 'nam3', 'nu4', 'nam5', 'nu5', 'nam2']

['nu1', 'nam1', 'nu2', 'nam4', 'nu3', 'nam5', 'nu4', 'nam2', 'nu5', 'nam3']

['nu1', 'nam1', 'nu2', 'nam4', 'nu3', 'nam5', 'nu4', 'nam3', 'nu5', 'nam2']

3. Một hợp có 3 bi xanh, 4 bi đỏ, 5 bi vàng. Chọn ngẫu nhiên 3 bi sao cho có đủ 3 màu. Số cách chọn là:

import **itertools**

Xanh = ["xanh1" , "xanh2" , "xanh3"]

Do = ["do1", "do2", "do3", "do4" ]

Vang = ["vang1", "vang2", "vang3", "vang4", "vang5"]

ChinhHop\_Xanh = **list**(**itertools**.**permutations**(Xanh,1))

ChinhHop\_Do = **list**(**itertools**.**permutations**(Do,1))

ChinhHop\_Vang = **list**(**itertools**.**permutations**(Vang,1))

KetQua = []

for a in ChinhHop\_Xanh:

    for b in ChinhHop\_Do:

        for c in ChinhHop\_Vang:

            M = []

            for x,y,z in **zip**(a,b,c):

                M.**append**(x)

                M.**append**(y)

                M.**append**(z)

            KetQua.**append**(M)

**print**("Số cách hợp lệ là: ",**len**(KetQua))

for x in KetQua:

**print**(x)

Số cách hợp lệ là: 60

['xanh1', 'do1', 'vang1']

['xanh1', 'do1', 'vang2']

['xanh1', 'do1', 'vang3']

['xanh1', 'do1', 'vang4']

['xanh1', 'do1', 'vang5']

['xanh1', 'do2', 'vang1']

['xanh1', 'do2', 'vang2']

['xanh1', 'do2', 'vang3']

['xanh1', 'do2', 'vang4']