BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC KINH TẾ TP HỒ CHÍ MINH

TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ

ĐỒ ÁN MÔN HỌC

ĐỀ TÀI:

PHÂN TÍCH VÀ DỰ BÁO MỨC LƯƠNG
CỦA CÁC CÔNG VIỆC THUỘC LĨNH VỰC KHOA HỌC DỮ LIỆU

Học phần: LẬP TRÌNH PHÂN TÍCH DỮ LIỆU

Chuyên ngành: Khoa học dữ liệu – Khóa 47

Nhóm Sinh Viên:

Họ và tên	MSSV
Lê Trần Khánh Phú	31211024087
Phạm Minh Phước	31211027663
Đỗ Quang Thiên Phú	31211024191
Trương Thanh Phong	31211027662
Nguyễn Tân Niên	31211027661

Giảng Viên: TS. Nguyễn An Tế

TP. Hồ Chí Minh, ngày 26 tháng 11 năm 2023

LỜI CẢM ƠN

Để hoàn thành đề tài này, trước tiên, nhóm chúng em xin gửi đến khoa Công nghệ thông tin, trường Công nghệ và thiết kế - Đại học UEH lời cảm ơn chân thành và sâu sắc nhất vì đã đưa môn Lập trình phân tích dữ liệu vào chương trình giảng dạy, giúp chúng em được học, khai thác và thực hành nhiều kiến thức bổ ích từ học phần này.

Đặc biệt, nhóm chúng em xin gửi lời cảm ơn sâu sắc nhất đến giảng viên giảng dạy TS. Nguyễn An tế đã nhiệt tình giảng dạy, truyền tải, hướng dẫn chúng em nhiều kiến thức bổ ích trong suốt thời gian học. Căm ơn sự hỗ trợ, giúp đỡ tận tính của Thầy để nhóm chúng em có thể thực hiện đề tài một cách tốt nhất.

Vì kiến thức của nhóm còn nhiều hạn chế, trong quá trình học và hoàn thiện đề tài chắc chắn chúng em không tránh khỏi những sai sót. Nhóm chúng em rất mong nhận được nhận xét, ý kiến và đánh giá từ Thầy để đề tài của nhóm được tốt hơn.

Nhóm chúng em xin chân thành cảm ơn Thầy. Kính chúc Thầy có nhiều sức khỏe, thành công và hạnh phúc.

MỤC LỤC

DANH MỤC HÌNH ẢNH

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

1. Xác định đề tài

Data jobs là một trong những nhóm ngành về công nghệ thông tin đang có sức hút hiện nay. Tập dữ liệu **Data jobs salaries** dạng bảng này bao gồm danh sách tất cả các mức lương được thu thập cho tới hiện tại, cùng với các thông tin chi tiết như: kinh nghiệm, tên công việc, loại hình làm việc (full-time, part-time,..), thời lượng làm việc từ xa, thời lượng, vị trí công ty, v.v. Các yếu tố này sẽ được nhóm tiến hành thực hiện phân tích thông qua các bước khai thác dữ liệu nghiên cứu, sử dụng thuật toán các máy học áp dụng phương pháp hồi quy các giá trị thu được từ bộ dữ liệu.

2. Mục đích nghiên cứu

Mục tiêu nghiên cứu là hiểu rõ nội dung các mức lương theo kinh nghiệm, trình độ, quốc gia. Qua đó xác định sự đa dạng và sự khác biệt của các mức lương. Ngoài ra, cũng đặt ra mục tiêu xác định sự tương đồng giữa các mức lương dựa trên các đặc điểm của thuộc tính khác và áp dụng Machine Learning để hiểu rõ hơn về các mô hình hồi quy trong dữ liệu.

3. Giới hạn phạm vi đề tài

Hạn chế nghiên cứu trong các khía cạnh sau: Phân tích lương của các công việc Dữ liệu bị hạn chế thời gian từ năm 2020 đến năm 2023. Sự tương đồng giữa các mức lương sẽ được xác định dựa trên các thuộc tính nhất định. Áp dụng Machine Learning, nhóm sẽ sử dụng các phương pháp hồi quy để đạt được cái nhìn toàn diện về cấu trúc và xu hướng trong dữ liệu, nhưng với sự tập trung đặc biệt vào các nhóm dữ liệu có sự tương đồng đáng chú ý.

4. Phương pháp nghiên cứu

Sau khi sử dụng các phương pháp phân tích dữ liệu, để tìm ra lương được trả cho một công việc thuộc lĩnh vực Dữ liệu thì nhóm sẽ cần đi tìm mối quan hệ của hàm $y = f(x)$ và mục tiêu của nhóm là tìm hàm $f(x)$ tốt để có thể dự đoán được lương của công việc Data Science. Từ đó, nhóm sẽ xây dựng mô hình máy học hồi quy có giám sát bởi vì biến mục tiêu là một biến liên tục.

5. Ngôn ngữ sử dụng

Ngôn ngữ lập trình chính cho đề tài này là Python, và sử dụng nền tảng Google Colab để triển khai mã nguồn và tạo báo cáo. Điều này giúp giảng viên và đồng nghiệp dễ dàng theo dõi quá trình nghiên cứu và kiểm tra các kết quả.

CHƯƠNG 2: TỔNG QUAN BỘ DỮ LIỆU NGHIÊN CỨU

1. Tổng quan về bộ dữ liệu thu thập

1.1. Giới thiệu bộ dữ liệu

Bộ dữ liệu "Data jobs salaries" được sử dụng trong báo cáo của nhóm được lấy từ trang Kaggle - một trang web phổ biến với hàng trăm tập dữ liệu phân tích. Bộ dữ liệu mô tả các mức lương công việc về dữ liệu được được cập nhật liên tục ở các vị trí, các quốc gia khác nhau. Ứng với mỗi hàng trong tập dữ liệu Data jobs salaries là mức lương cho 1 vị trí, công ty, mức kinh nghiệm, khối lượng công việc từ xa, ... Việc cập nhật liên tục các mức lương trong suốt khoảng thời gian đã hình thành tập dữ liệu Data jobs salaries với tổng số mức lương trong bộ dữ liệu là 8805 dòng.

1.2. Các thuộc tính của bộ dữ liệu

Tên thuộc tính	Mô tả thuộc tính	Loại thuộc tính
work_year	Năm mà lương được trả	Numerical
experience_level	Trình độ kinh nghiệm trong năm. Với các giá trị: • EN (Entry-level) - Junior • MI (Mid-level) - Intermediate • SE (Senior-level) - Expert • EX (Executive-level) - Director	Categorical
employment_type	Loại hình tuyển dụng. • PT (Part-time) • FT (Full-time) • CT (Contract) • FL (Freelance)	Categorical
job_title	Danh xưng công việc	Categorical
salary	Tổng số tiền lương đã trả	Numerical

salary_currency	Đơn vị tiền tệ của tiền lương được trả (mã tiền tệ ISO 4217)	Categorical
salary_in_usd	Mức lương tính bằng đơn vị USD	Numerical
employee_residence	Quốc gia thường trú của nhân viên (mã quốc gia ISO 3166)	Categorical
remote_ratio	Khối lượng công việc làm việc từ xa trên tổng lượng công việc • 0 - Không làm việc từ xa (dưới 20%) • 50 - Làm việc từ xa một phần • 100 - Hoàn toàn làm việc từ xa (trên 80%)	Numerical
company_location	Quốc gia đặt trụ sở chính của công ty hoặc chi nhánh trong hợp đồng nhân viên (mã quốc gia ISO 3166)	Categorical
company_size	Quy mô công ty, theo số người làm việc cho công ty trong năm • S (Small) - Dưới 50 nhân viên • M (Median) - Từ 50 đến 250 nhân viên • L (Large) - Trên 250 nhân viên	Categorical

1.3. Mục tiêu khai thác bộ dữ liệu hướng đến

Ứng dụng vào bộ dữ liệu Data jobs salaries, sau khi tiền xử lý và chuẩn hóa dữ liệu về dạng số, nhóm xác định biến mục tiêu và biến độc lập để xây dựng mô hình hồi quy để dự báo biến mục tiêu. Ta xác định được các biến độc lập đã được tiền xử lý, sau đó dựa vào mô hình hồi quy để dự báo mức lương dựa vào các biến độc lập như work_year, experience_level, employment_type, job_title, remote_ratio, company_location, company_size, để có thể đưa ra được kết quả dự báo chính xác.

2. Tiến trình xử lý dữ liệu thu thập

Ta sử dụng bộ dữ liệu Data jobs salaries để hình thành những phân tích liên quan đến các mức lương. Để làm được điều này, ta chia dự án thành các nhiệm vụ nhỏ có thể tổ chức và quản lý được thông qua sáu giai đoạn của quy trình phân tích dữ liệu:

- **Giai đoạn 1 - Business Understanding:**

Ở đây mục tiêu chúng ta hướng đến là làm sao có thể thực hiện được phân tích dữ liệu về mức lương khi có được các thông tin liên quan. Khi thể hiện được mức lương thông qua các yếu tố khác, ta sẽ tiếp cận được mục tiêu quan trọng là giúp các doanh nghiệp có thể đưa ra các mức lương phù hợp với từng vị trí, kinh nghiệm.

- **Giai đoạn 2 - Data Understanding:**

Kiểm tra tình trạng dữ liệu để xác định dữ liệu có đang phù hợp với mục tiêu khai thác hay không. Bước làm này ta sẽ xác định những thuộc tính cần thiết cho quá trình phân tích cũng như kiểm tra mức độ "sạch" của bộ dữ liệu với các tiêu chuẩn đề ra: không có chứa dữ liệu missing data, lọc nhiễu và lượt bớt dữ liệu.

- **Giai đoạn 3 - Data preparation:**

Thực hiện các bước tiền xử lý để chuẩn hóa dữ liệu sẵn sàng cho các giai đoạn tiếp theo. Giai đoạn này thường chiếm đến 90% thời gian của cả quy trình. Ta sẽ thực hiện xử lý bộ dữ liệu theo những phát hiện tìm được ở giai đoạn 2 bằng lập trình phân tích dữ liệu - tiền xử lý dữ liệu.

- **Giai đoạn 4 - Modeling:**

Sử dụng các mô hình thống kê, máy học để xác định các mẫu/ quy luật của dữ liệu. Với bộ dữ liệu Data jobs salaries, ta cần thực hiện lập trình phân tích bộ dữ liệu gốc làm sao để hình thành những mô hình hướng mục tiêu đề ra cũng như thể hiện nó qua biểu đồ biểu diễn trực quan dữ liệu giúp người đọc quan sát dễ dàng và thuận tiện đánh giá, phân tích.

- **Giai đoạn 5 - Evaluation:**

Kiểm tra tính hiệu quả của mô hình có đáp ứng với mục tiêu doanh nghiệp hay không, đủ tin cậy hay không.

- **Giai đoạn 6 - Deployment:**

Đưa mô hình giải pháp vào ứng dụng trong các hoạt động của doanh nghiệp. Bước này làm ta có thể đưa ra những giải pháp dựa theo đánh giá và mô hình ở giai đoạn 4 và 5.

CHƯƠNG 3: TIỀN XỬ LÝ VÀ PHÂN TÍCH BỘ DỮ LIỆU NGHIÊN CỨU

1. Kiểm tra tình trạng bộ dữ liệu gốc

1.1. Tình trạng bộ dữ liệu thu thập

Bộ dữ liệu Data jobs salary được thu thập cần phải kiểm tra các yếu tố về tình trạng đầu vào để làm sạch dữ liệu (Data cleaning/ cleansing), loại bỏ nhiễu (remove noise) cũng như hiệu chỉnh những phần dữ liệu không nhất quán (correct data inconsistencies)

- **Chi tiết bộ dữ liệu**

```
df.info()
print("số lượng phần tử bộ dữ liệu:", df.size)
print("Số dòng và cột bộ dữ liệu:", df.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8805 entries, 0 to 8804
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   work_year           8805 non-null   int64 
 1   experience_level    8805 non-null   object
 2   employment_type     8805 non-null   object
 3   job_title           8805 non-null   object
 4   salary              8805 non-null   int64 
 5   salary_currency     8805 non-null   object
 6   salary_in_usd       8805 non-null   int64 
 7   employee_residence  8805 non-null   object
 8   remote_ratio        8805 non-null   int64 
 9   company_location    8805 non-null   object
 10  company_size        8805 non-null   object
dtypes: int64(4), object(7)
memory usage: 756.8+ KB
số lượng phần tử bộ dữ liệu: 96855
Số dòng và cột bộ dữ liệu: (8805, 11)
```

- **Kiểm tra số lượng dữ liệu đầu vào**

Ta thấy bộ dữ liệu gồm 8805 dòng và 11 cột với 2 kiểu dữ liệu chính là: phân loại (7 thuộc tính) và số (4 thuộc tính).

- **Kiểm tra dữ liệu thiếu**

```
# Kiểm tra Null value
df.isnull().sum()
```

```
work_year           0
experience_level    0
employment_type     0
job_title           0
salary              0
salary_currency     0
salary_in_usd       0
employee_residence  0
remote_ratio        0
company_location    0
company_size        0
dtype: int64
```

Từ kết quả ta thấy được trong thuộc tính bộ dữ liệu hiện không có giá trị bị thiếu, bộ dữ liệu đã thu thập và cập nhật đầy đủ các giá trị.

- **Kiểm tra dữ liệu nhiễu**

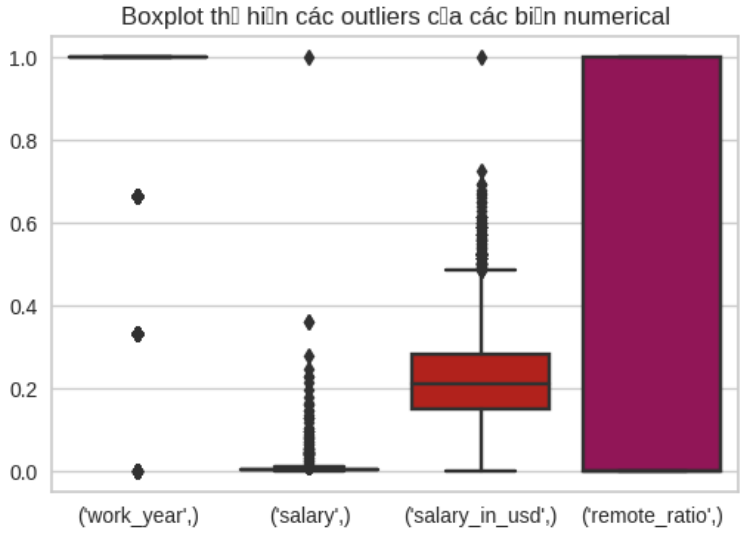
Ta kiểm tra dữ liệu nhiễu bằng biểu đồ hộp boxplot. Trong thống kê mô tả, boxplot là một loại biểu đồ thường được sử dụng trong phân tích dữ liệu. Bên cạnh đó, boxplot còn biểu diễn trực quan sự phân phối dữ liệu số và độ lệch thông qua hiển thị tứ phân vị, bao gồm: điểm tối thiểu, phần tư thứ nhất Q1 (thấp hơn), trung vị, phần tư thứ ba Q3 (trên) và điểm tối đa. Biểu đồ hộp boxplot cung cấp một bản tóm tắt trực quan về dữ liệu cho phép các nhà nghiên cứu nhanh chóng xác định các giá trị trung vị, độ phân tán của tập dữ liệu và các dấu hiệu của độ lệch. Cho nên, ta sẽ sử dụng boxplot để kiểm tra các điểm ngoại lệ (nhiễu Outlier) nằm bên ngoài râu của boxplot.

Trước hết ta sẽ chuẩn hóa các dữ liệu numerical về Minmaxscaler để có thể dễ dàng xem các điểm dữ liệu.

```
numerical = df[['work_year', 'salary','salary_in_usd',
'remote_ratio']]
scaler = MinMaxScaler()
scaler.fit(numerical)
numerical = scaler.transform(numerical)
numerical = pd.DataFrame(numerical, columns = [['work_year',
'salary','salary_in_usd', 'remote_ratio']])
numerical.head()
```

Ta truyền vào thuộc tính dữ liệu định tính để quan sát nhiễu.

```
numeric_columns = ['work_year', 'salary', 'salary_in_usd',
'remote_ratio']
plt.subplots(figsize = (6,4))
sns.boxplot(numerical)
plt.title("Boxplot thể hiện các outliers của các biến numerical")
plt.show()
```



Hình 1: Boxplot thể hiện các outliers của các biến numerical

Quan sát thấy từ boxplot ta có các dữ liệu nhiễu outlier nằm ngoài râu của biểu đồ hộp nên sẽ tiến hành lọc nhiễu bộ dữ liệu.

1.2. Phân tích khám phá (EDA)

Phân tích khám phá dữ liệu (EDA) là một bước quan trọng trong quá trình phân tích dữ liệu. EDA giúp có cái nhìn đầu tiên về dữ liệu, từ đó mường tượng được độ phức tạp của dữ liệu và vạch ra những bước đầu tiên cần làm.

Trong biểu diễn trực quan dữ liệu, EDA được thực hiện bằng cách sử dụng các kỹ thuật trực quan hóa dữ liệu để khám phá các đặc điểm và mối quan hệ của dữ liệu. Các kỹ thuật này giúp hiểu rõ hơn về dữ liệu, từ đó có thể đưa ra các quyết định sáng suốt hơn trong quá trình phân tích.

Có hai lệnh được sử dụng phổ biến để khám phá dữ liệu trong data visualization, đó là: ***info()*** - Cung cấp thông tin tóm tắt của dữ liệu, bao gồm số lượng hàng, số lượng cột, kiểu dữ liệu của mỗi cột, v.v. như đã trình bày ở trên và ***head()*** - để xem một số hàng đầu tiên của bộ dữ liệu

```
df = pd.read_csv('/content/salaries.csv')
df.head()
```

	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remote_ratio	company_location	company_size
0	2023	EX	FT	Data Science Director	212000	USD	212000	US	0	US	M
1	2023	EX	FT	Data Science Director	190000	USD	190000	US	0	US	M
2	2023	MI	FT	Business Intelligence Engineer	35000	GBP	43064	GB	0	GB	M
3	2023	MI	FT	Business Intelligence Engineer	35000	GBP	43064	GB	0	GB	M
4	2023	SE	FT	Machine Learning Engineer	245700	USD	245700	US	0	US	M

- **Nhận diện các biến**

Bỏ 2 thuộc tính salary và salary_currency vì đã có thuộc tính salary_in_USD chuyển tiền lương về chung một đơn vị tiền là USD.

```
df.drop(df[['salary','salary_currency']],axis=1, inplace=True)
```

- **Xem mỗi thuộc tính có bao nhiêu danh mục:**

Đổi những thuộc tính work_year, remote_ratio thành biến phân loại vì các thuộc tính này có ít giá trị, đổi giá trị của các biến còn lại để dễ trong quan sát trực quan dữ liệu.

```
# Đổi những biến work_year và biến remote_ratio thành biến phân loại
df["work_year"] = df["work_year"].astype(object)
df['remote_ratio_clean']=df['remote_ratio']
df["remote_ratio"].replace([100,50,0], ['No remote work', 'Partially
remote' ,'Fully remote'],inplace = True)

# Đổi giá trị của những biến còn lại
df['salary_in_usd'] = df['salary_in_usd'].astype(int)
df["experience_level"].replace(['EN','MI','SE', 'EX'], ['Entry',
'Mid', 'Senior', 'Executive'], inplace = True)
df["employment_type"].replace(["PT", "FT", "CT", "FL"], ["Part-time",
"Full-time", "Contract", "Freelance"], inplace=True)
df["company_size"].replace(["S", "M", "L"], ["Small", "Medium",
"Large"], inplace=True)

# Chuyển đổi ký hiệu quốc gia thường trú của công ty thành theo mã
quốc gia ISO3
converted_country_ = coco.convert(names=df['company_location'],
to="ISO3")
df['company_location'] = converted_country_
# Chuyển đổi ký hiệu quốc gia thường trú của nhân viên thành theo mã
quốc gia ISO3
converted_country = coco.convert(names=df['employee_residence'],
to="ISO3")
df['employee_residence'] = converted_country
```

1.3. Giải pháp xử lý

Bộ dữ liệu không có Missing Value nhưng có rất nhiều Outlier nên ta tiến hành loại bỏ các Outlier, các Outlier sẽ tác động đến các biến phụ thuộc khác và làm thay đổi sự ảnh hưởng của chúng lên nhau.

2. Tiền xử lý dữ liệu

2.1. Phân tích đơn biến - thống kê mô tả

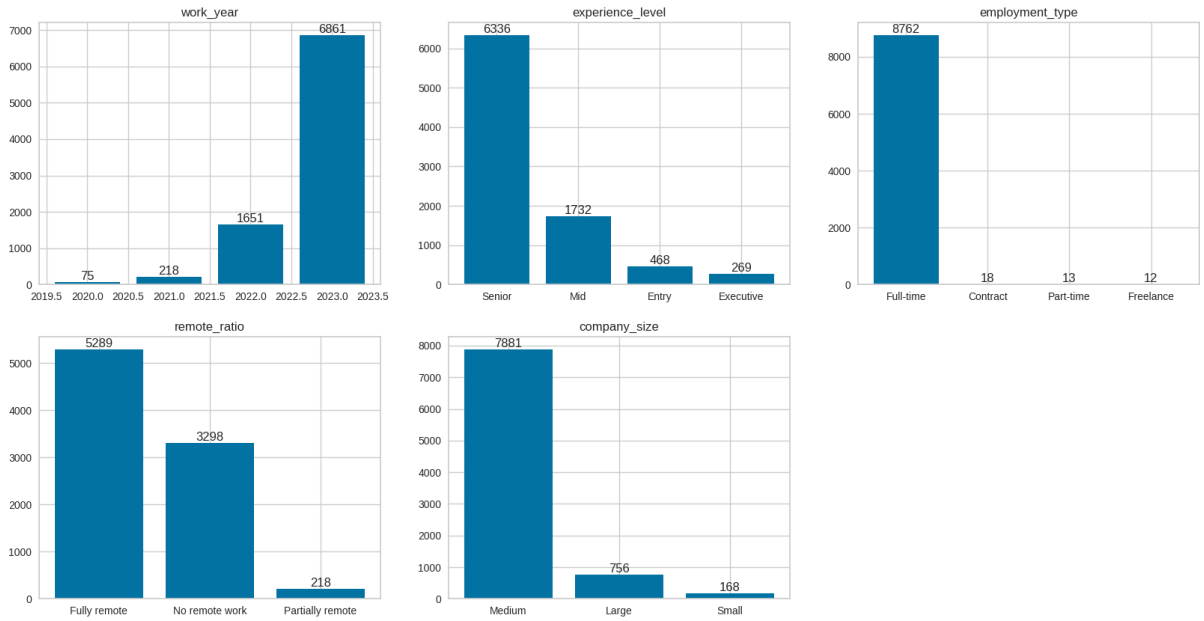
```
# Xem bảng mô tả của các cột phân loại
df.describe(include = 'O')
```

	work_year	experience_level	employment_type	job_title	employee_residence	remote_ratio	company_location	company_size
count	8805	8805	8805	8805	8805	8805	8805	8805
unique	4	4	4	124	86	3	74	3
top	2023	Senior	Full-time	Data Engineer	USA	Fully remote	USA	Medium
freq	6861	6336	8762	2062	7527	5289	7576	7881

Bộ dữ liệu có 8 biến phân loại: work_year, experience_level, employment_type, job_title, employee_residence, remote_ratio, company_location, company_size.

- **Biểu đồ cột biểu diễn những thuộc tính phân loại có ít danh mục.**

```
# Phân tích những cột phân loại có ít danh mục:
cat_col = df.select_dtypes(include='object').columns
cat_col_less = [col for col in cat_col if df[col].nunique() <= 5]


plt.figure(figsize=([20, 10]))  # Khung (width, height): inch
for i in range(len(cat_col_less)):
    counts = df[cat_col_less[i]].value_counts()
    plt.subplot(2, 3, (i + 1))  # Subplot 1 trong ma trận (2 rows, 3
cols)
    plt.title(cat_col_less[i])
    bars = plt.bar(counts.index, counts.values)


    # Thêm thông số số lượng dữ liệu trên mỗi cột
    for bar in bars:
        yval = bar.get_height()
        plt.text(bar.get_x() + bar.get_width() / 2, yval + 0.05,
round(yval, 2), ha='center', va='bottom')


plt.show()
```



Hình 2: Biểu đồ cột biểu diễn những thuộc tính phân loại có ít danh mục

Nhận xét:

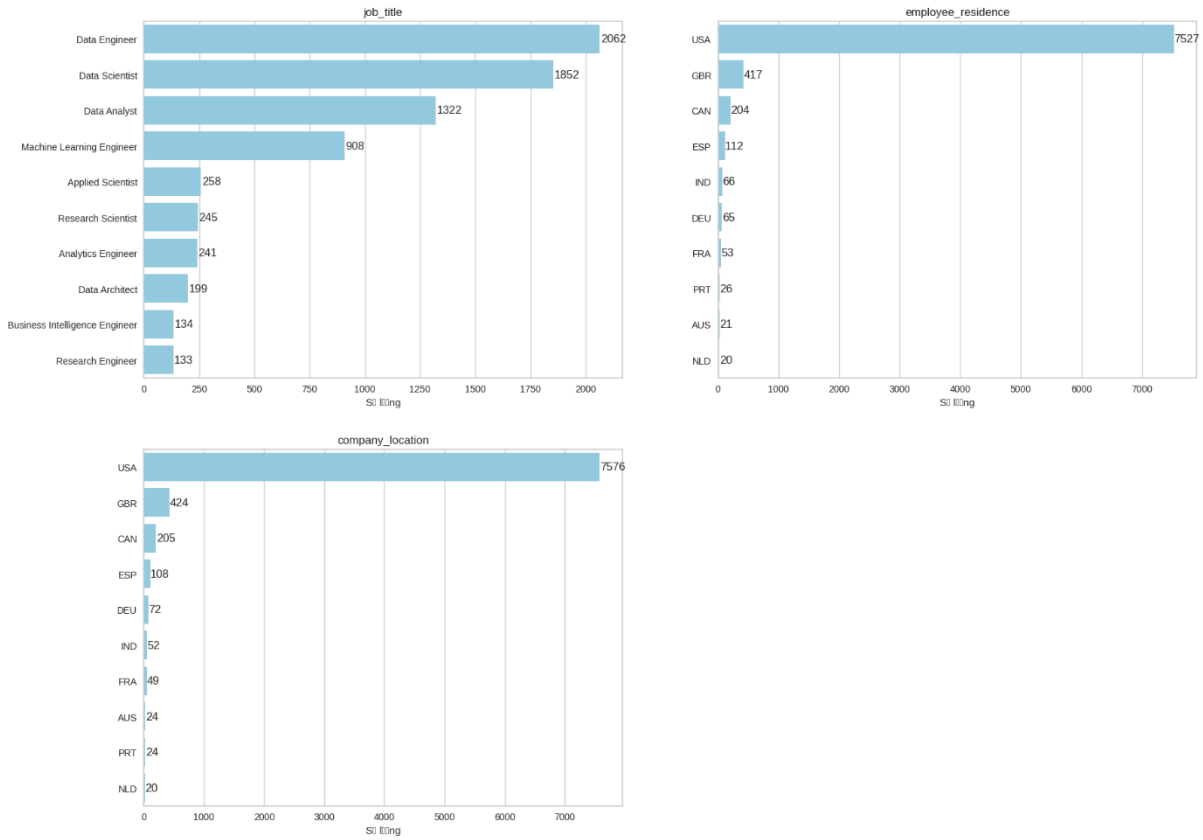
- Khi nhìn vào số lượng lương được lấy theo năm, chúng ta có thể thấy rằng năm 2020 chỉ có 75 việc làm trong tập dữ liệu này, năm 2021 tăng lên 218 việc làm, sau đó lên tới 1651 việc làm vào năm 2022 và năm 2023 là hơn 6000 việc làm. việc

làm. Số lượng việc làm có xu hướng tăng theo cấp số nhân qua các năm, tăng hơn 7000% kể từ năm 2020. Mức tăng cao nhất giữa các năm là giai đoạn 2021-2022 với mức tăng 657%. Điều này cũng có thể phụ thuộc vào việc lấy dữ liệu không đồng đều giữa các năm.

- Đối với kinh nghiệm làm việc, Senior chiếm phần trăm cao nhất chiếm gần 3/4 bộ dữ liệu. Còn Mid đứng thứ 2 với khoảng 20%. Thấp nhất là Executive, chỉ khoảng 3%. Điều này cũng hợp lý trong thực tế.
- Bỏ thuộc tính employment_type vì có đến 99,51% giá trị là Fulltime. Dữ liệu về các danh mục còn lại rất ít, tổng số lượng các danh mục còn lại chưa đến 1%.
- Với tỷ lệ làm việc từ xa thì làm việc từ xa chiếm phần trăm cao nhất và ít nhất là làm việc từ xa một phần.
- Quy mô công ty phổ biến nhất là quy mô cỡ vừa, quy mô cỡ nhỏ là ít phổ biến nhất.

- **Biểu đồ cột biểu diễn những thuộc tính phân loại có nhiều danh mục.**

```
# Biểu đồ thanh biểu diễn top 10 các thuộc tính có nhiều danh mục
cat_col_much = [col for col in cat_col if df[col].nunique() > 5]

plt.figure(figsize=(20, 15))
for i in range(len(cat_col_much)):
  plt.subplot(2, 2, (i+1)) # subplot 1 trong ma trận (2 rows, 3 cols)
  top10 = df[cat_col_much[i]].value_counts()[:10]
  ax = sns.barplot(x=top10.values, y=top10.index,color='skyblue')
  ax.set_xlabel('Số lượng')
  ax.set_title(cat_col_much[i])
  for j, v in enumerate(top10.values):
    ax.annotate(str(v), xy=(v+5,j), va='center')
plt.show()
```



Hình 3: Biểu đồ cột biểu diễn những thuộc tính phân loại có nhiều danh mục

```
print('Số lượng danh xưng công việc trong bộ dữ liệu là:',
df['job_title'].value_counts().size)
```

```
Số lượng danh xưng công việc trong bộ dữ liệu là: 124
```

Có 124 danh xưng công việc trong tập dữ liệu. Data engineer, Data scientist, Data analyst và Machine Learning Engineer là 4 chức danh công việc phổ biến nhất.

Có thể dễ dàng nhận thấy rằng những chức danh khác cũng liên quan đến 4 chức danh công việc thường xuyên xảy ra và 4 chức danh này chiếm phần lớn nên tạo cột mới phân các chức danh công việc vào 4 nhóm công việc này và thêm một cột 'other' cho các công việc còn lại.

```
# Vì chúng ta có nhiều giá trị trong cột chức danh công việc
# Hãy tạo một hàm để xác định các vai trò được lặp lại nhiều nhất dựa
trên từ khóa

def role(title):
    if any(keyword in title.lower() for keyword in ['data scientist',
'data science', 'scientist']):
        return 'Data Scientist'
    elif any(keyword in title.lower() for keyword in ['data analyst',
'analyst']):
        return 'Data Analyst'
```

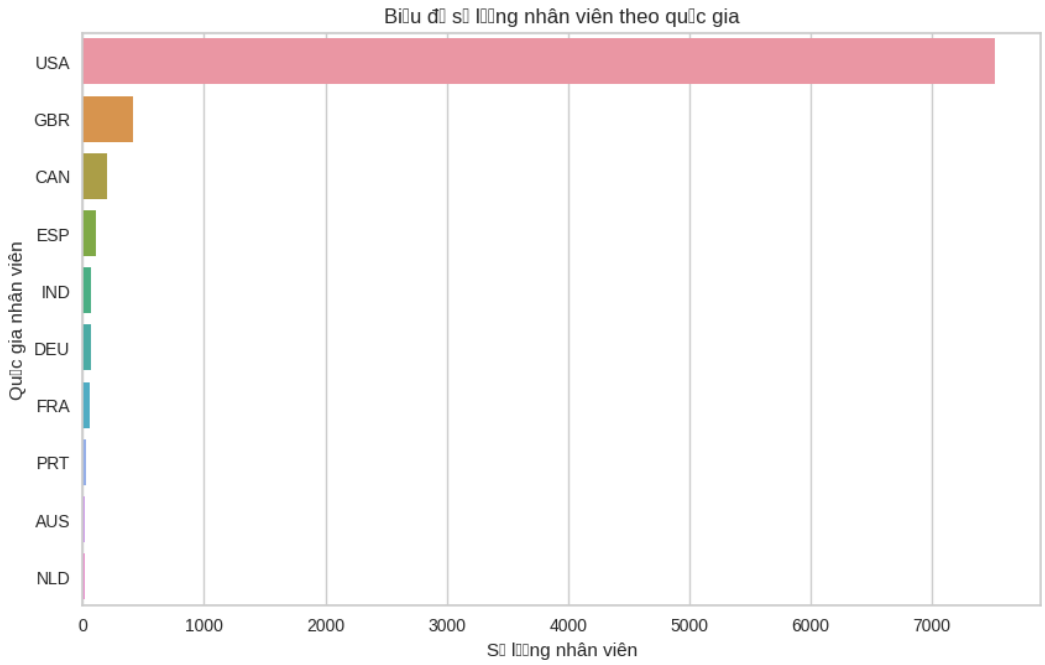
```
    elif any(keyword in title.lower() for keyword in ['data
engineer']):
        return 'Data Engineer'
    elif any(keyword in title.lower() for keyword in ['machine
learning engineer', 'machine learning', 'ai', 'ml', 'deep
learning']):
        return 'Machine Learning Engineer'
    else:
        return 'Other'
```

```
# Tạo cột mới cho các chức danh công việc
df['job_title_clean'] = df['job_title'].apply(role)
df['job_title_clean'].value_counts()
```

```
Data Scientist              2752
Data Engineer               2093
Data Analyst                1528
Other                       1247
Machine Learning Engineer   1185
Name: job_title_clean, dtype: int64
```

- **Biểu đồ thể hiện số lượng của Employee Residence:**

```
employee_res = df['employee_residence'].value_counts()
top_10_employees = employee_res.head(10)

plt.figure(figsize=(10, 6))
sns.countplot(y='employee_residence',
data=df, order=top_10_employees.index)
plt.xlabel('Số lượng nhân viên')
plt.ylabel('Quốc gia nhân viên')
plt.title('Biểu đồ số lượng nhân viên theo quốc gia')
plt.show()
```

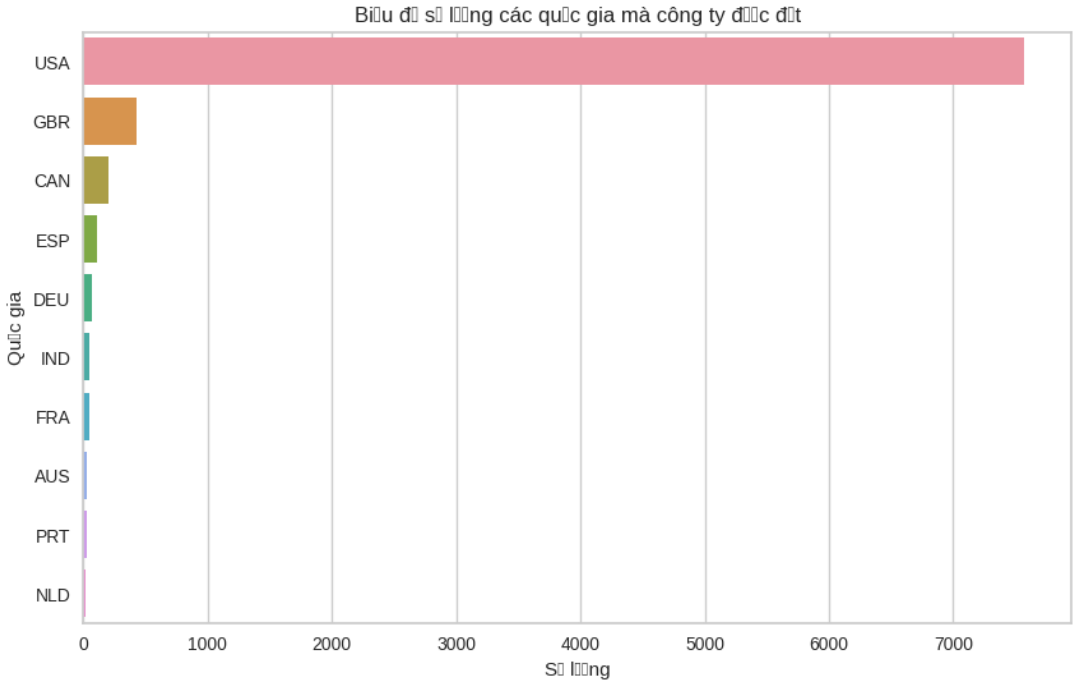


Hình 4: Biểu đồ thể hiện số lượng của Employee Residence

Nhận xét: Hầu hết nhân viên đến từ USA.

- **Biểu đồ thể hiện số lượng của Company Location:**

```
company_loc = df['company_location'].value_counts()
top_10_company = company_loc.head(10)


plt.figure(figsize=(10, 6))
sns.countplot(y='company_location', data=df,
order=top_10_company.index)
plt.xlabel('Số lượng')
plt.ylabel('Quốc gia')
plt.title('Biểu đồ số lượng các quốc gia mà công ty được đặt')
plt.show()
```



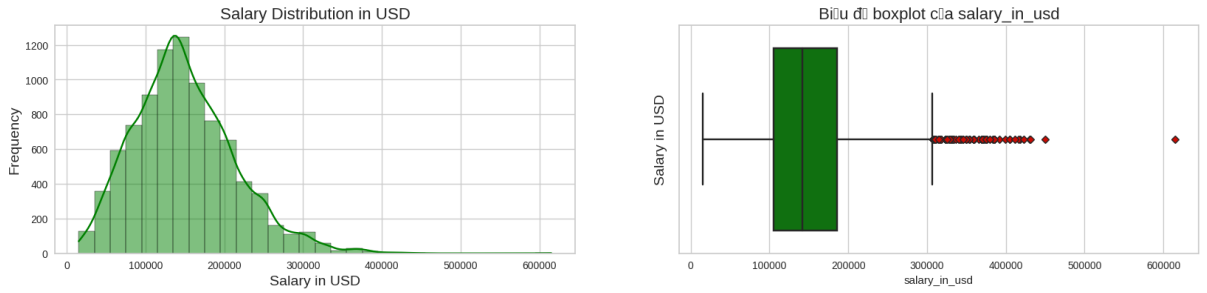
Hình 5: Biểu đồ thể hiện số lượng của Company Location

Nhận xét: Tương tự với Employee Residence, hầu hết vị trí công ty ở USA.

- **Vẽ biểu đồ phân phối với KDE và biểu đồ box plot**

```
# Tạo biểu đồ phân phối với KDE và box plot
fig, ax = plt.subplots(1, 2, figsize=(20, 4))

sns.histplot(df['salary_in_usd'], bins=30, kde=True, color='green',
edgecolor='k', ax=ax[0])

ax[0].set_title('Salary Distribution in USD', fontsize=16)
ax[0].set_xlabel('Salary in USD', fontsize=14)
ax[0].set_ylabel('Frequency', fontsize=14)

outlier = dict(marker = 'D', markerfacecolor = 'r')
```

```
sns.boxplot(x=df['salary_in_usd'], color='green',flierprops =
outlier, ax=ax[1])

ax[1].set_title('Biểu đồ boxplot của salary_in_usd', fontsize=16)
ax[1].set_ylabel('Salary in USD', fontsize=14)

plt.show()
```



Hình 6: Biểu đồ phân phối salary_in_usd với KDE và biểu đồ boxplot

Nhận xét: Từ biểu đồ bên trái, mức lương phần lớn của các công việc Data Science kể từ năm 2020 đến năm 2023 là khoảng 150 nghìn USD và nó được trải rộng từ mức lương khoảng 15 nghìn USD đến mức lương khoảng 400 nghìn USD. Từ biểu đồ bên phải, thấy được thuộc tính salary_in_usd có xuất hiện outliers nên cần phải xử lý outliers.

- **Các đại lượng về xu thế trung tâm**

```
print('Các đại lượng về xu thế trung tâm:')
print("  - Trung bình (Mean):",round(np.mean(df['salary_in_usd']),4))
print('  - Trung vị
(Median):',round(np.median(df['salary_in_usd']),4))
print('  - Yếu vị (Mode):',sp.stats.mode(df['salary_in_usd'])[0],'với
số lần là',sp.stats.mode(df['salary_in_usd'])[1])
```

```
Các đại lượng về xu thế trung tâm:
  - Trung bình (Mean): 149488.2656
  - Trung vị (Median): 142200.0
  - Yếu vị (Mode): 150000 với số lần là 188
```

- **Các đại lượng về độ phân tán**

```
## Các đại lượng về độ phân tán
print('\nCác đại lượng về độ phân tán:')

# Khoảng biến thiên (Range)
data_range = np.max(df['salary_in_usd']) -
np.min(df['salary_in_usd'])
print("  - Khoảng biến thiên (Range):", data_range)

# Tứ phân vị (Quartile)
q1 = np.percentile(df['salary_in_usd'], 25)
q2 = np.percentile(df['salary_in_usd'], 50)
```

```
q3 = np.percentile(df['salary_in_usd'], 75)
print("  - Tứ phân vị (Quartile):")
print("     Q1:", q1)
print("     Q2:", q2)
print("     Q3:", q3)

# Khoảng trải giữa (InterQuartile Range)
iqr = q3 - q1
print("  - Khoảng trải giữa (InterQuartile Range):", iqr)

# Phương sai (Variance)
variance = np.var(df['salary_in_usd'])
print("  - Phương sai (Variance):", round(variance,4))

# Độ lệch chuẩn (Standard deviation)
std_deviation = np.std(df['salary_in_usd'])
print("  - Độ lệch chuẩn (Standard Deviation):",
round(std_deviation,4))
```

```
Các đại lượng về độ phân tán:
  - Khoảng biến thiên (Range): 600201
  - Tứ phân vị (Quartile):
     Q1: 105000.0
     Q2: 142200.0
     Q3: 185900.0
  - Khoảng trải giữa (InterQuartile Range): 80900.0
  - Phương sai (Variance): 4124010353.5111
  - Độ lệch chuẩn (Standard Deviation): 64218.458
```

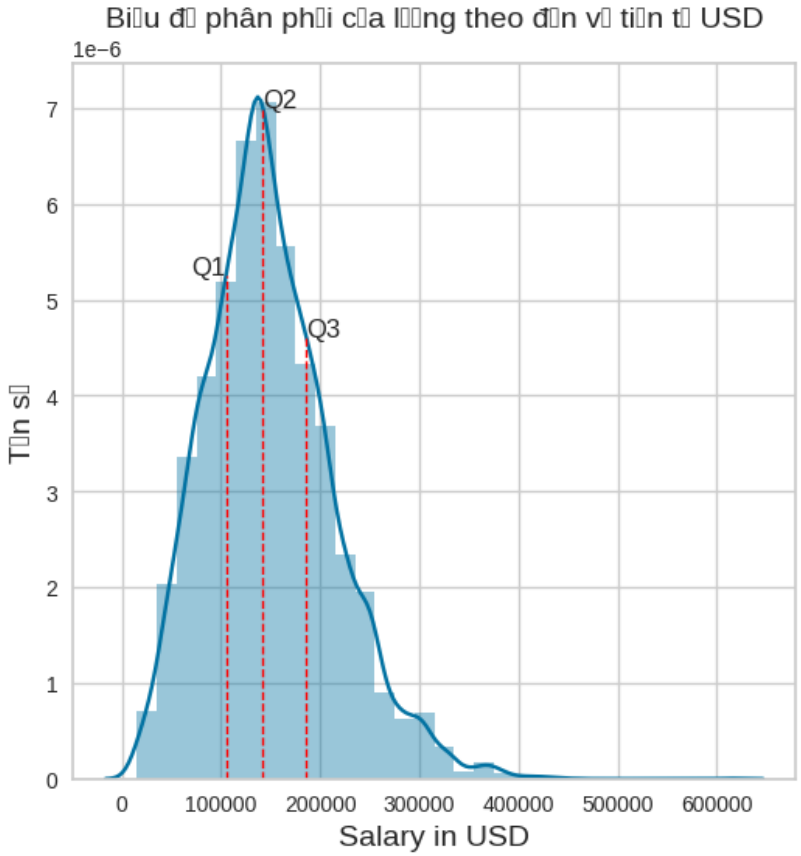
Nhận xét:

- Trong trường hợp này, khoảng biến thiên là 600,201, cho thấy có sự biến động lớn giữa giá trị cao nhất và giá trị thấp nhất.
- IQR là có giá trị là 80,900.0: là khoảng giữa Q1 và Q3.
- Phương sai là 4124010353.5111, cho thấy dữ liệu có sự biến động lớn.
- Với độ lệch chuẩn là 64218.458 nên dữ liệu có độ phân tán lớn.

- **Các đại lượng về hình dáng**

```
#Các đại lượng về hình dáng
# Vẽ biểu đồ phân phối
plt.figure(figsize = (6, 6))
sns.distplot(df['salary_in_usd'], bins=30, kde=True, hist = True)
plt.title('Biểu đồ phân phối của lương theo đơn vị tiền tệ USD',
fontsize=14)
plt.xlabel('Salary in USD', fontsize=14)
plt.ylabel('Tần số', fontsize=14)
percentiles = np.percentile(df['salary_in_usd'], [25, 50, 75])
 # Calculate heights of quartiles as the height of kde at the
quartile positions
kde = gaussian_kde(df['salary_in_usd'])
kde_q75 = kde(q3)
```

```
kde_q50 = kde(q2)
kde_q25 = kde(q1)
# Add labels for quartiles
plt.text(q3, kde_q75, 'Q3', horizontalalignment='left')
plt.text(q2, kde_q50, 'Q2', horizontalalignment='left')
plt.text(q1, kde_q25, 'Q1', horizontalalignment='right')

for percentile in [q1, q2, q3]:
    plt.axvline(percentile, color='red', linestyle='--', linewidth=1,
ymax = kde(percentile)/(7.5*1e-6) )
plt.show()
print('\nCác đại lượng về hình dáng:')
salaries = df['salary_in_usd']
#a. Độ nghiêng (skewness)
skewness = sp.stats.skew(salaries)
print("  - Độ nghiêng (skewness):", round(skewness,4))
#b. Độ nhọn (kurtosis)
kurtosis = sp.stats.kurtosis(salaries)
print("  - Độ nhọn (kurtosis):", round(kurtosis,4))
```



Hình 7: Biểu đồ phân phối của lương theo đơn vị tiền tệ USD

```
Các đại lượng về hình dáng:
  - Độ nghiêng (skewness): 0.6462
  - Độ nhọn (kurtosis): 0.86
```

Nhận xét:

- Vì chỉ số độ nghiêng skewness lớn hơn 0 nên phân phối bị lệch dương, phân bổ nhiều phía bên trái.
- Độ nhọn kurtosis < 3 nên phân phối có nhiều giá trị ở đuôi hơn so với phân phối chuẩn.

2.2. Làm sạch dữ liệu

- Dữ liệu bị thiếu

```
## Đếm số dòng chứa NA (None, NaN) của mỗi cột
print('*** Số lượng giá trị bị thiếu của các cột:')
print(len(df) - df.count())
```

```
*** Số lượng giá trị bị thiếu của các cột:
work_year            0
experience_level     0
employment_type      0
job_title            0
salary_in_usd        0
employee_residence   0
remote_ratio         0
company_location     0
company_size         0
remote_ratio_clean   0
job_title_clean      0
dtype: int64
```

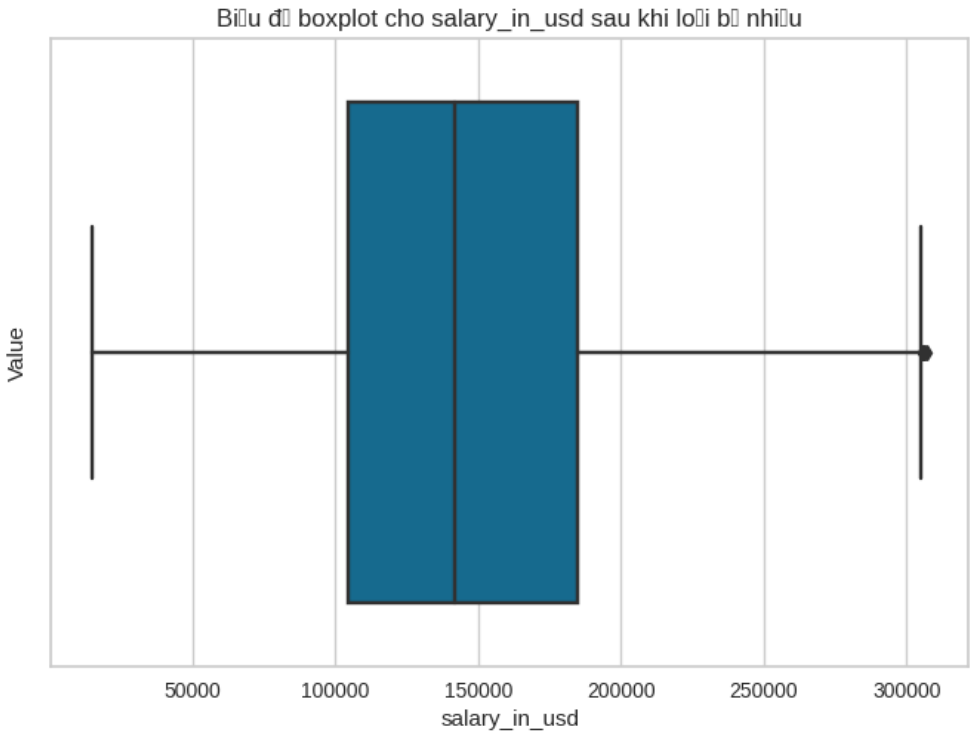
Nhận xét: Bộ dữ liệu không có dòng nào bị thiếu nên bỏ qua bước xử lý dữ liệu bị thiếu.

- Dữ liệu bị nhiễu

```
# Xử lý dữ liệu bị nhiễu
Q1 = df['salary_in_usd'].quantile(0.25)
Q3 = df['salary_in_usd'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outlier = df[(df['salary_in_usd'] < lower_bound) |
(df['salary_in_usd'] > upper_bound)]
df = df[(df['salary_in_usd'] >= lower_bound) & (df['salary_in_usd']
<= upper_bound)]

print(f"Số giá trị bị loại bỏ: {len(outlier)}")
```

```
Số giá trị bị loại bỏ: 159
```

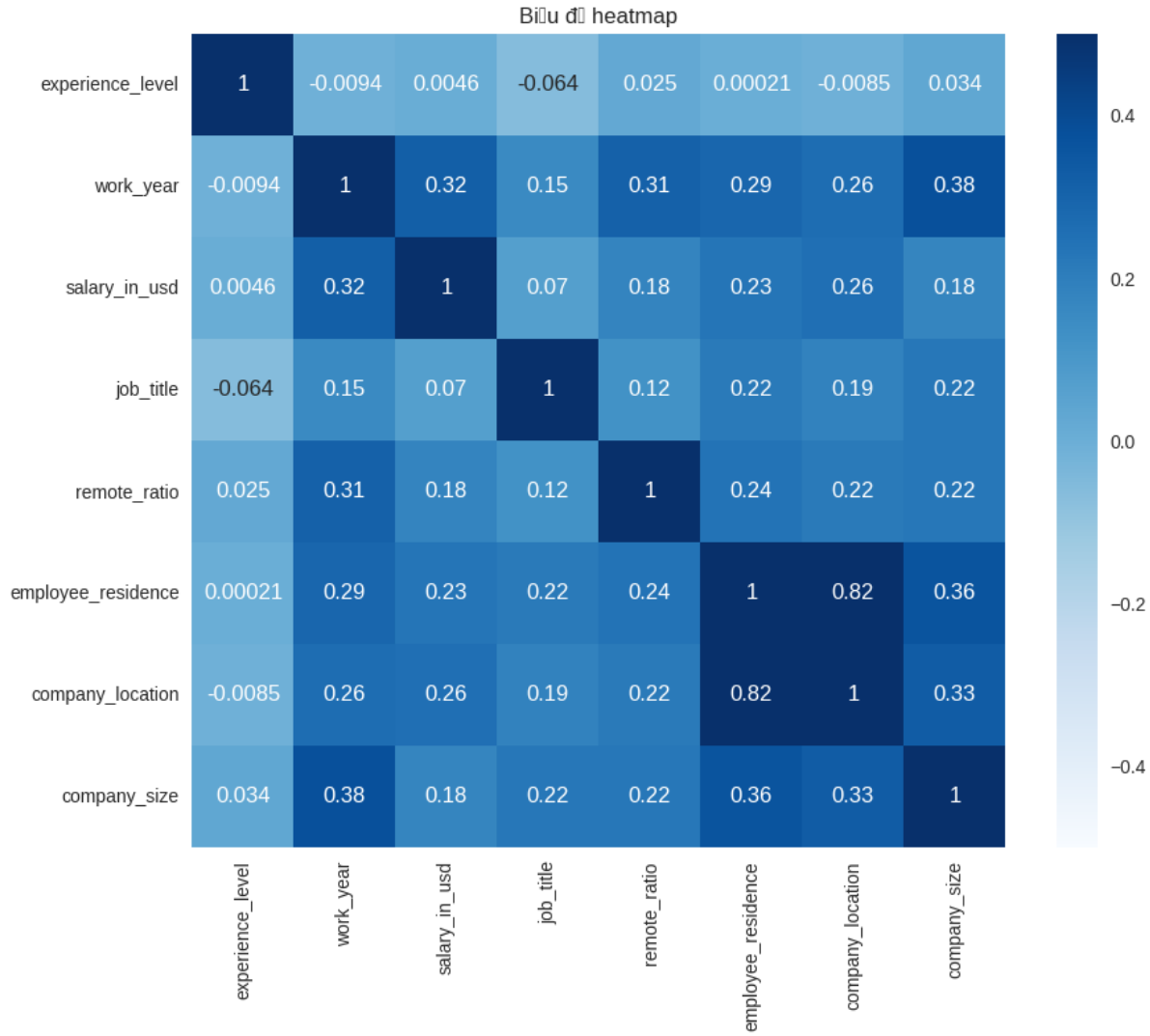
```
plt.title('Biểu đồ boxplot cho salary_in_usd sau khi loại bỏ nhiễu')
plt.xlabel('salary_in_usd')
sns.boxplot(x='salary_in_usd', data=df)
plt.ylabel('Value')
plt.show()
```



Hình 8: Biểu đồ boxplot cho salary_in_usd sau khi loại bỏ nhiễu

- **Kiểm định tương quan**

```
vars = ['experience_level', 'work_year', 'salary_in_usd',
'job_title', 'remote_ratio', 'employee_residence',
'company_location', 'company_size']

# Tạo ma trận tương quan giữa các biến
corr_matrix = df[vars].apply(lambda x: pd.factorize(x)[0]).corr()

# Tạo heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='Blues', square=True,
vmin=-0.5, vmax=0.5)

# Hiển thị biểu đồ
plt.title('Biểu đồ heatmap')
plt.show()
```



Hình 9: Biểu đồ tương quan giữa các biến

Nhận xét: Trong biểu đồ, có xuất hiện 2 thuộc tính có tương quan cao là employee_residence và company_location. Thực hiện kiểm định Chi bình phương để kiểm định lại.

```
# Bước 1: Tạo bảng chéo crosstab giữa biến employee_residence và
company_location
crosstab_table = pd.crosstab(df['employee_residence'],
df['company_location'])

# Bước 2: Chuyển đổi bảng crosstab thành array
observed_array = crosstab_table.values

# Bước 3: Kiểm định Chi-square và kết luận theo phương pháp p-value
chi2, p, _, _ = chi2_contingency(observed_array)

alpha = 0.05
confidence_level = 1 - alpha

if p < alpha:
```

```
    conclusion = "Bác bỏ giả thuyết Ho: employee_residence và
company_location là PHỤ THUỘC lẫn nhau."
else:
    conclusion = "Không thể bác bỏ giả thuyết Ho: employee_residence
và company_location là ĐỘC LẬP."

# In kết quả
print("Kết quả kiểm định Chi-square:")
print("p-value: {:.10f}".format(p))
print("Kết luận:", conclusion)
```

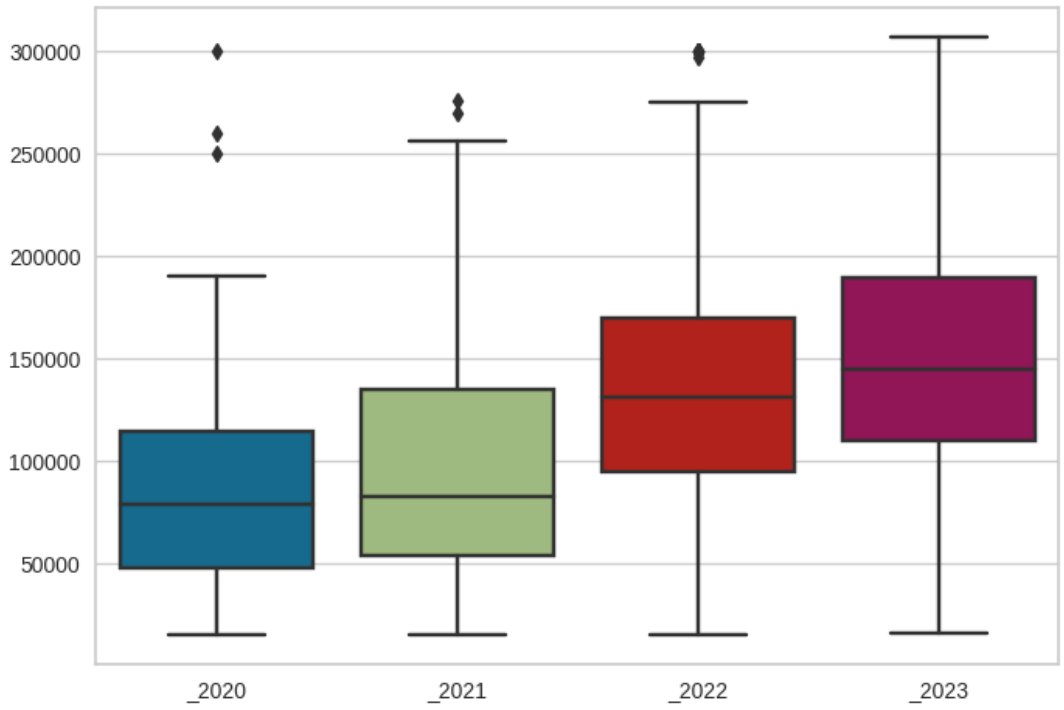
```
Kết quả kiểm định Chi-square:
p-value: 0.0000000000
Kết luận: Bác bỏ giả thuyết Ho: employee_residence và
company_location là PHỤ THUỘC lẫn nhau.
```

- **Kiểm định ANOVA**

```
# 1. Tạo dataframe chứa các mẫu dữ liệu
_2020=df[df['work_year']==2020]['salary_in_usd'].reset_index(drop=True)
_2021=df[df['work_year']==2021]['salary_in_usd'].reset_index(drop=True)
_2022=df[df['work_year']==2022]['salary_in_usd'].reset_index(drop=True)
_2023=df[df['work_year']==2023]['salary_in_usd'].reset_index(drop=True)
data=pd.DataFrame({'_2020':_2020,'_2021':_2021,'_2022':_2022,'_2023':_20
23})
data.head()
```

	_2020	_2021	_2022	_2023
0	100000.0	185000.0	36940.0	212000
1	44753.0	56000.0	80000.0	190000
2	164000.0	60000.0	97712.0	43064
3	47899.0	150000.0	16455.0	43064
4	300000.0	30000.0	15000.0	245700

```
# 2. Biểu diễn trực quan bằng Boxplot --> quan sát sự khác biệt giữa
các nhóm
sns.boxplot(data = data)
plt.show()
```



Hình 10: Biểu đồ boxplot giá trị biến salary_in_usd theo work_year

```
# 3. Phân tích ANOVA với alpha = 5%.
alpha = .05
# 3a) Kiểm định điều kiện _2020, _2021, _2022 có cùng phương sai
(Levene)
print('''Các giả thuyết kiểm định LEVENE:
    H0: VAR(_2020) = VAR(_2021) = VAR(_2022) = VAR(_2023)
    Ha: Các phương sai KHÔNG BẰNG NHAU''')

levene, p = sp.stats.levene(_2020,_2021,_2022,_2023)
print(f'\nTrị thống kê Levene = {levene:4f}; p = {p:4f}')
if  p>alpha:
  print(f'p > alpha cho nên KHÔNG bác bỏ H0: VAR(_2020) = VAR(_2021)
= VAR(_2022) = VAR(_2023)')
else:
  print(f'p < alpha cho nên bác bỏ H0 -> Các phương sai không bằng
nhau.')
```

```
Các giả thuyết kiểm định LEVENE:
    H0: VAR(_2020) = VAR(_2021) = VAR(_2022) = VAR(_2023)
    Ha: Các phương sai KHÔNG BẰNG NHAU

Trị thống kê Levene = 2.803982; p = 0.038285
p < alpha cho nên bác bỏ H0 -> Các phương sai không bằng nhau.
```

```
# 3b) Áp dụng One-way ANOVA
# Hàm f_oneway() chỉ trả về F-statistic và p-value, KHÔNG tạo ANOVA
table
f, p = sp.stats.f_oneway(_2020, _2021, _2022, _2023)
```

```
print('H0: KHÔNG có sự khác biệt giữa lương qua các năm.')
print('Ha: Có sự khác biệt giữa lương qua các năm.\n')
if (p < alpha):
    print(f'* Trị số p = {p} < {alpha} cho nên bác bỏ H0 ==> có sự khác biệt giữa lương qua các năm')
else:
    print(f'* Trị số p = {p} >= {alpha} cho nên KHÔNG bác bỏ H0 ==> KHÔNG có sự khác biệt giữa lương qua các năm')
## p-value có ý nghĩa về mặt thống kê (p < 0.05)
```

```
H0: KHÔNG có sự khác biệt giữa lương qua các năm.
Ha: Có sự khác biệt giữa lương qua các năm.

* Trị số p = 8.193496357672687e-77 < 0.05 cho nên bác bỏ H0 ==> có sự khác biệt giữa lương qua các năm
```

```
## 3c) Hậu kiểm Tukey HSD
df_melt = pd.melt(data.reset_index(),
                  id_vars = ['index'],
                  value_vars = ['_2020', '_2021', '_2022','_2023'])
df_melt=df_melt.dropna()
df_melt.columns = ['index', 'work_year', 'value']  # tên các cột
model           = ols('value ~ C(work_year)', data = df_melt).fit()

m_comp = pairwise_tukeyhsd(endog  = df_melt['value'],
                           groups = df_melt['work_year'],
                           alpha  = 0.05)
print(m_comp)
```

```
     Multiple Comparison of Means - Tukey HSD, FWER=0.05
=================================================================
group1 group2  meandiff  p-adj     lower      upper    reject
-----------------------------------------------------------------
 _2020  _2021  6937.8102 0.8134 -13236.3359 27111.9563  False
 _2020  _2022 42935.1331    0.0  25084.4126 60785.8536   True
 _2020  _2023 61226.5837    0.0   43661.886 78791.2815   True
 _2021  _2022 35997.3229    0.0  25266.5198 46728.1261   True
 _2021  _2023 54288.7735    0.0  44040.8215 64536.7256   True
 _2022  _2023 18291.4506    0.0  14208.2445 22374.6567   True
-----------------------------------------------------------------
```

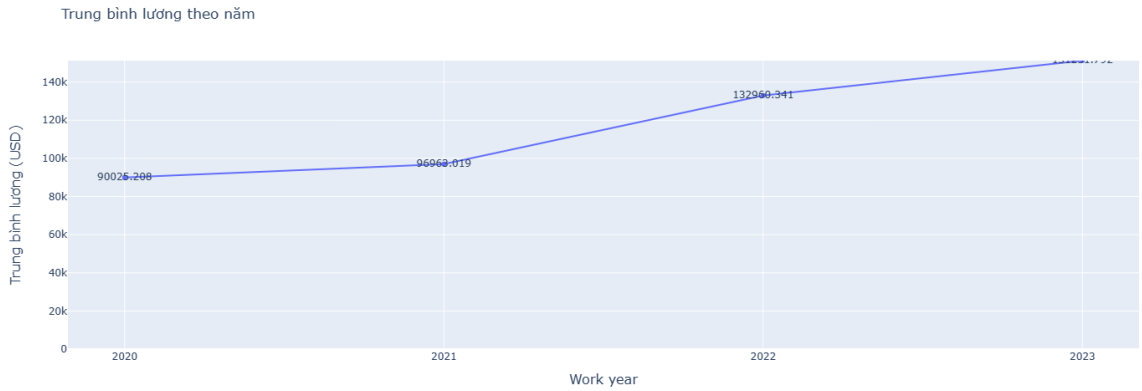
Nhận xét: Ngoại trừ cặp (_2020, _2021) có reject = False, tất cả so sánh những cặp khác đều bác bỏ H0 (reject = True), nghĩa là có sự khác biệt đáng kể về mặt thống kê.

3. Phân tích đa biến

- **Vẽ biểu đồ lineplot biểu diễn trung bình lương qua các năm.**

```
# work_year & salary_in_usd - lineplot
# Lineplot thể hiện trung bình lương theo các năm
salary_by_year = df.groupby('work_year')['salary_in_usd'].mean()

fig = px.line(df,
```

```
                x=salary_by_year.index.astype('str'),
                y=salary_by_year.values,
                text=np.round(salary_by_year.values,3),
                title='Trung bình lương theo năm')
fig.update_layout(
    xaxis=dict(
        title='Work year',
        titlefont_size=16
    ),
    yaxis=dict(
        title='Trung bình lương (USD)',
        titlefont_size=16
    ),
    yaxis_range=[0, max(salary_by_year.values)]
)
fig.show()
```



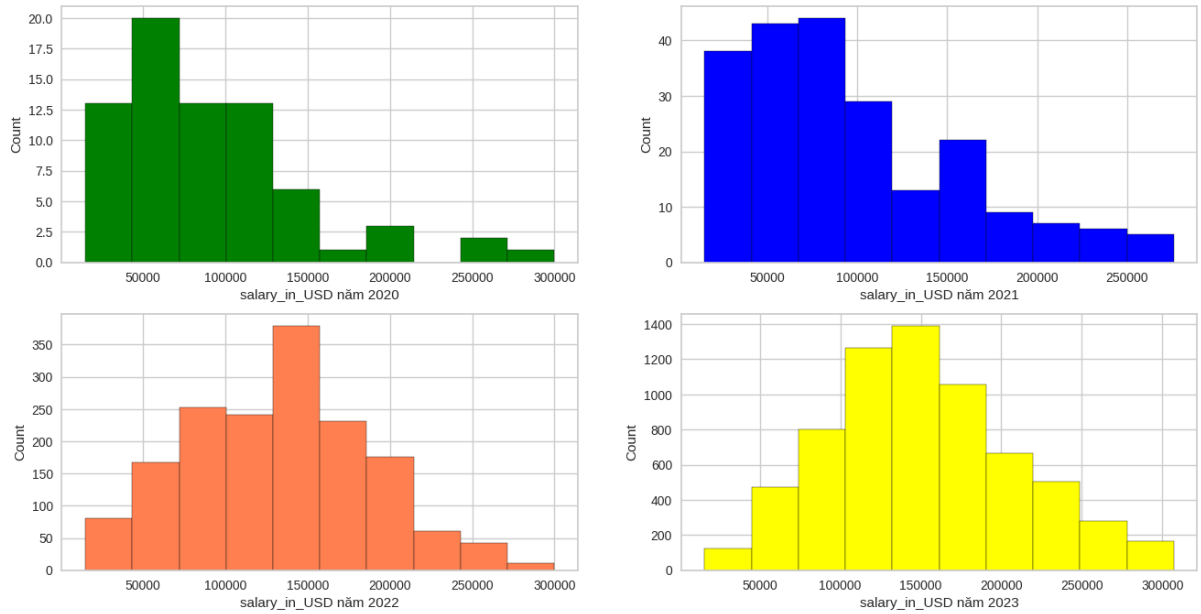
Hình 11: Biểu đồ lineplot biểu diễn trung bình lương qua các năm

Nhận xét: Xu hướng lương trung bình ngày càng tăng qua các năm, vào năm 2020, mức lương trung bình là khoảng 100 nghìn USD và năm 2023 là khoảng 150 nghìn USD. Tăng 52% trong 3 năm. Chỉ có năm 2021, trung bình lương giảm.

- **Biểu đồ tần số lương cho mỗi năm**

```
# work_year & salary_in_usd - subplot, hist
# Biểu đồ tần số lương cho mỗi năm
plt.figure(figsize=(16, 8))

plt.subplot(2, 2, 1)
plt.hist(df[df['work_year']==2020]['salary_in_usd'],  density =
False, color = 'green', edgecolor='k')
plt.xlabel('salary_in_USD năm 2020')
plt.ylabel('Count')

plt.subplot(2, 2, 2)
plt.hist(df[df['work_year']==2021]['salary_in_usd'],  density =
False, color = 'blue', edgecolor='k')
plt.xlabel('salary_in_USD năm 2021')
```

```
plt.ylabel('Count')

plt.subplot(2, 2, 3)
plt.hist(df[df['work_year']==2022]['salary_in_usd'],  density =
False, color = 'coral', edgecolor='k')
plt.xlabel('salary_in_USD năm 2022')
plt.ylabel('Count')

plt.subplot(2, 2, 4)
plt.hist(df[df['work_year']==2023]['salary_in_usd'],  density =
False, color = 'yellow', edgecolor='k')
plt.xlabel('salary_in_USD năm 2023')
plt.ylabel('Count')

plt.show()
```



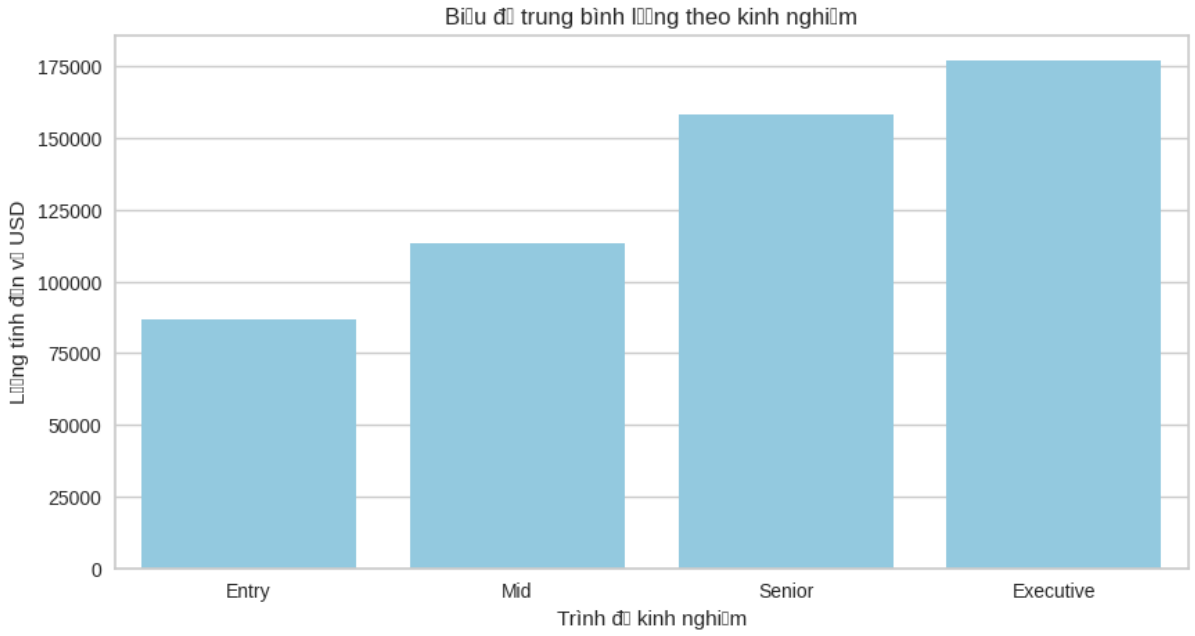
Hình 12: Biểu đồ phân phối lương cho mỗi năm

Nhận xét: Nhìn 4 biểu đồ, ta thấy đều bị lệch sang trái. Năm 2020 và 2021, độ cao của các thanh hạ dần từ trái sang phải, lương của 2 năm này tập trung nhiều nhất ở khoảng 50 nghìn đến 100 nghìn USD. Nhưng mà đến năm 2022 và 2023, biểu đồ ít lệch sang trái hơn mà đã nhích sang phải, tập trung nhiều ở khoảng 100 nghìn đến 200 nghìn USD.

- **Biểu đồ trung bình lương theo kinh nghiệm**

```
# experience_level & salary_in_usd - subplot, hist
experience_level_order = ['Entry', 'Mid', 'Senior', 'Executive']

# experience_level & salary_in_usd
# Biểu đồ trung bình lương theo kinh nghiệm
data = df.groupby('experience_level')['salary_in_usd'].mean()
```

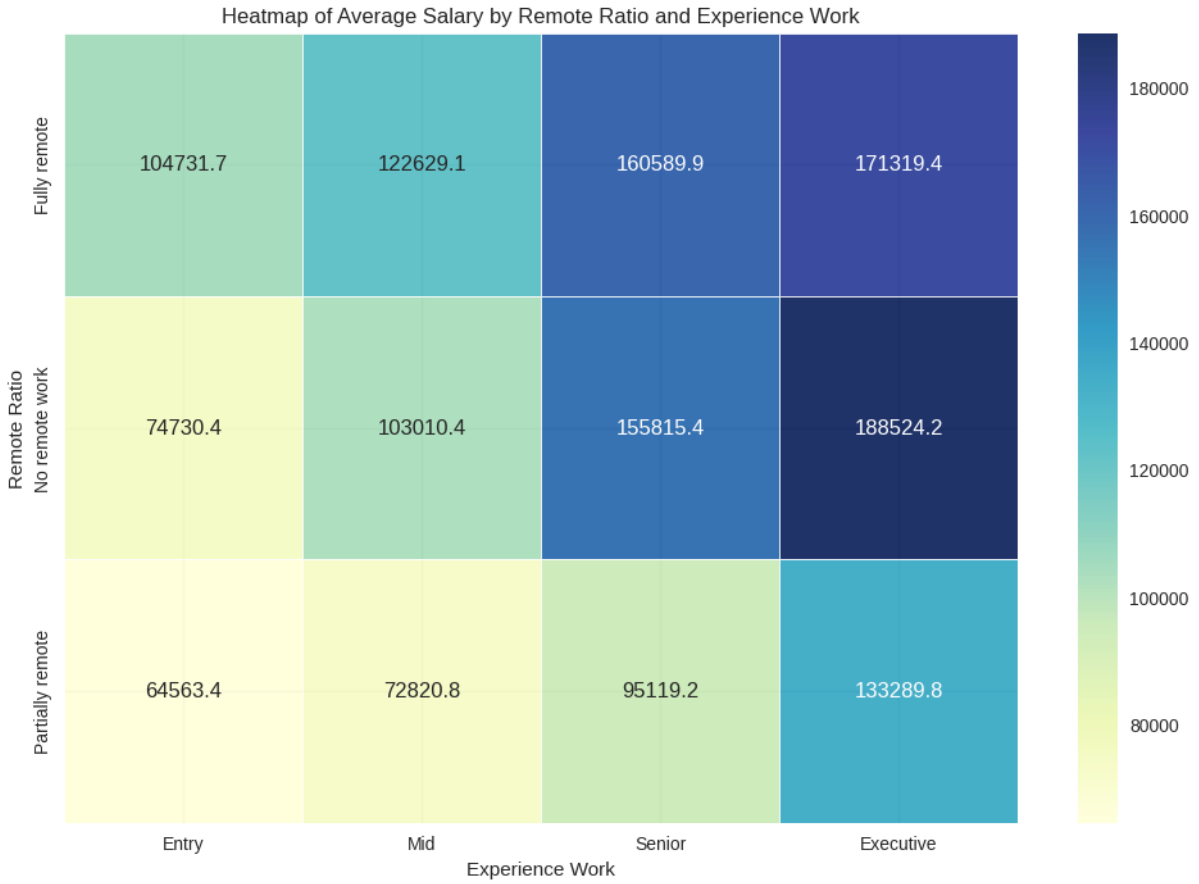
```
plt.figure(figsize=(10,5))
ax = sns.barplot(x=data.index, y=data.values,
order=experience_level_order, color='skyblue')
ax.set_xlabel('Trình độ kinh nghiệm')
ax.set_ylabel('Lương tính đơn vị USD')
ax.set_title('Biểu đồ trung bình lương theo kinh nghiệm')

plt.show()
```



Hình 13: Biểu đồ trung bình lương theo kinh nghiệm

```
## Heatmap of Average Salary by Job and Experience Work
average_salary_experience =
df.groupby(['experience_level','remote_ratio'])[['salary_in_usd']].mean(
).reset_index()
average_salary_experience['experience_level'] =
pd.Categorical(average_salary_experience['experience_level'],
categories=experience_level_order, ordered=True)
average_salary_experience =
average_salary_experience.sort_values(by='experience_level')

heatmap_data = average_salary_experience.pivot('remote_ratio',
'experience_level', 'salary_in_usd')

plt.figure(figsize=(12, 8))
sns.heatmap(heatmap_data, annot=True, cmap="YlGnBu", fmt=".1f",
            linewidths=0.5, alpha = 0.9)

plt.title('Heatmap of Average Salary by Remote Ratio and Experience
Work')
plt.xlabel('Experience Work')
plt.ylabel('Remote Ratio')
```

```
plt.show()
```



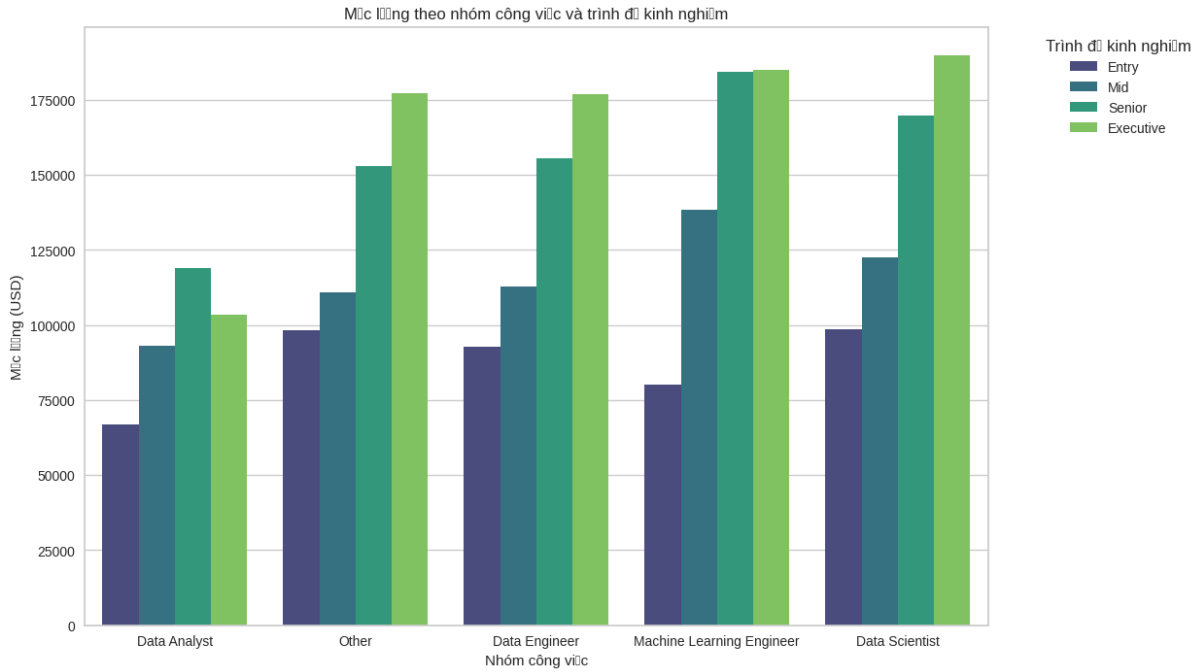
Hình 14: Heatmap lương trung bình theo Remote Ratio và Experience Level

Nhận xét: Mức lương tăng theo cả 2 chiều trình độ kinh nghiệm và hình phức làm việc (tăng từ làm việc từ xa một phần, không làm việc từ xa đến làm việc từ xa hoàn toàn). Nhìn vào heatmap, ta có thể thấy những ô trên cùng đường chéo có màu gần giống nhau, cho thấy lương trung bình của những ô này giống nhau. Vậy thì nếu trình độ kinh nghiệm cao hơn nhưng hình thức làm việc thấp hơn thì mức lương dự đoán sẽ gần bằng nhau. Có một trường hợp đặc biệt là lương của trình độ Executive của hình thức không làm việc từ xa cao hơn cả hình thức làm việc từ xa hoàn toàn, có thể là do các vị trí Executive thường đòi hỏi mức độ trách nhiệm và cam kết cao. Một số công việc yêu cầu sự hiện diện và tương tác trực tiếp nhiều hơn, điều này có thể làm tăng khả năng những người làm việc từ xa một phần có mức lương cao hơn.

- **Mức lương trung bình cho mỗi nhóm công việc và trình độ kinh nghiệm**

```
# Tính mức lương trung bình cho mỗi nhóm công việc và trình độ kinh
nghiệm
salary_by_experience = df.groupby(['job_title_clean',
'experience_level'])['salary_in_usd'].mean().reset_index()
```

```
salary_by_experience['experience_level'] =
pd.Categorical(salary_by_experience['experience_level'],
categories=experience_level_order, ordered=True)
salary_by_experience =
salary_by_experience.sort_values(by='experience_level')


# Vẽ biểu đồ
plt.figure(figsize=(12, 8))
sns.barplot(x='job_title_clean', y='salary_in_usd',
hue='experience_level', data=salary_by_experience, palette='viridis')
plt.title('Mức lương theo nhóm công việc và trình độ kinh nghiệm')
plt.xlabel('Nhóm công việc')
plt.ylabel('Mức lương (USD)')
plt.legend(title='Trình độ kinh nghiệm', bbox_to_anchor=(1.05, 1),
loc='upper left')
plt.show()
```



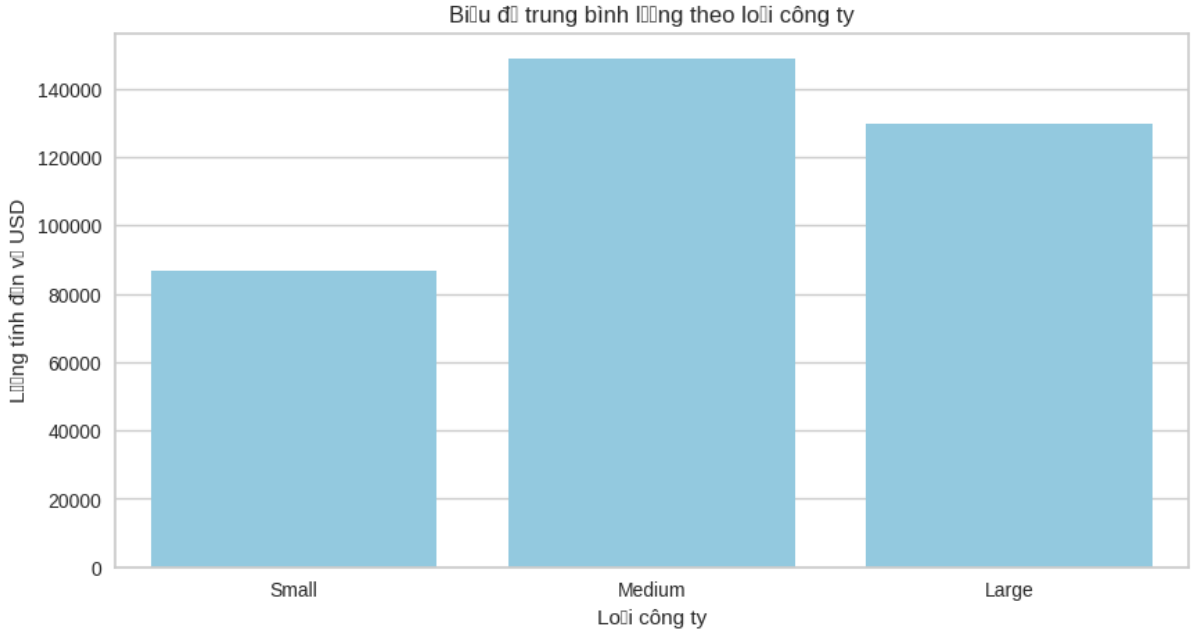
Hình 15: Lương trung bình theo Jobtile và Experience Level

Nhận xét: Hầu hết các nhóm công việc đều có mức lương tăng theo trình độ kinh nghiệm. Khi kinh nghiệm càng cao thì lương tăng cao một cách rõ rệt, biến động nhiều. Riêng chỉ có nhóm 'Data Analyst' lại khác biệt một chút khi trung bình lương của trình độ Senior lại cao nhất, và lương của trình độ kinh nghiệm khác nhau không có biến động nhiều như các nhóm công việc khác, có thể là do nhu cầu thị trường thay đổi, nhưng điều này cũng phụ thuộc vào rất nhiều yếu tố khác.

- **Biểu đồ trung bình lương theo loại công ty**

```
company_size_order = ['Small', 'Medium', 'Large']
```

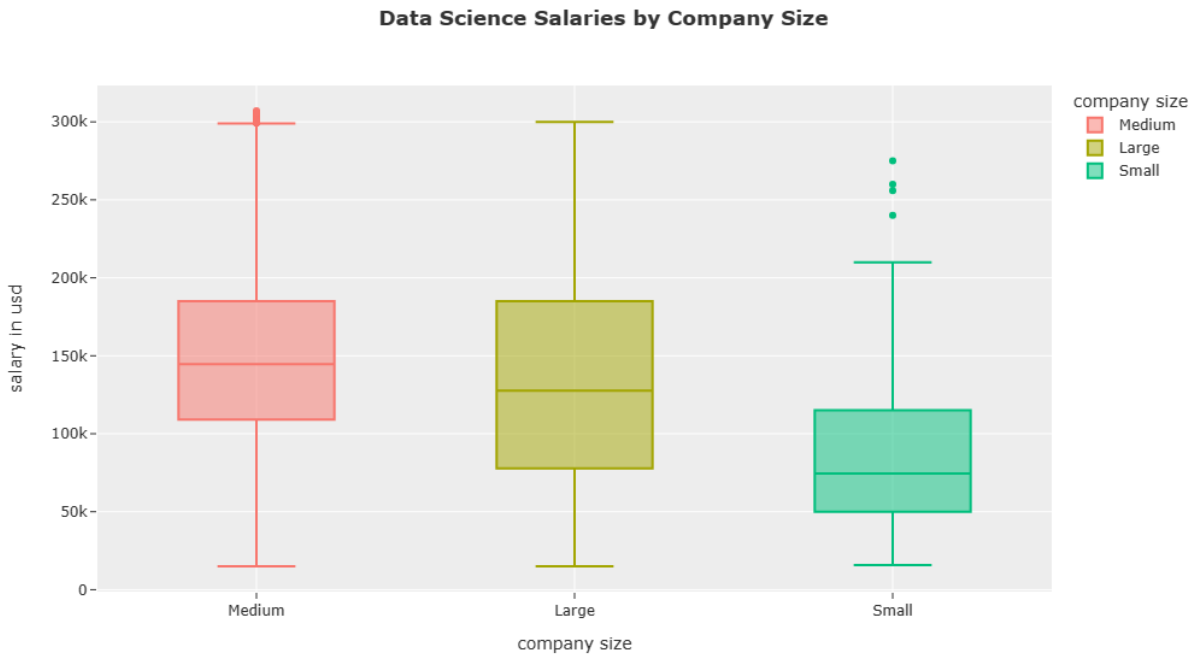
```
# Biểu đồ trung bình lương theo loại công ty
data = df.groupby('company_size')['salary_in_usd'].mean()

plt.figure(figsize=(10,5))
ax = sns.barplot(x=data.index, y=data.values,
order=company_size_order, color='skyblue')
ax.set_xlabel('Loại công ty')
ax.set_ylabel('Lương tính đơn vị USD')
ax.set_title('Biểu đồ trung bình lương theo loại công ty')

plt.show()
```



Hình 16: Biểu đồ trung bình lương theo loại công ty

- **Biểu đồ boxplot theo Salaries và Company Size**

```
# Company Size & Salary - boxplot
px.box(df,x='company_size',y='salary_in_usd',color='company_size',
       template='ggplot2',labels={'company_size':'company size',
                                  'salary_in_usd':'salary in usd'},
       title='<b>Data Science Salaries by Company Size',
       width=1000, height=600)
```

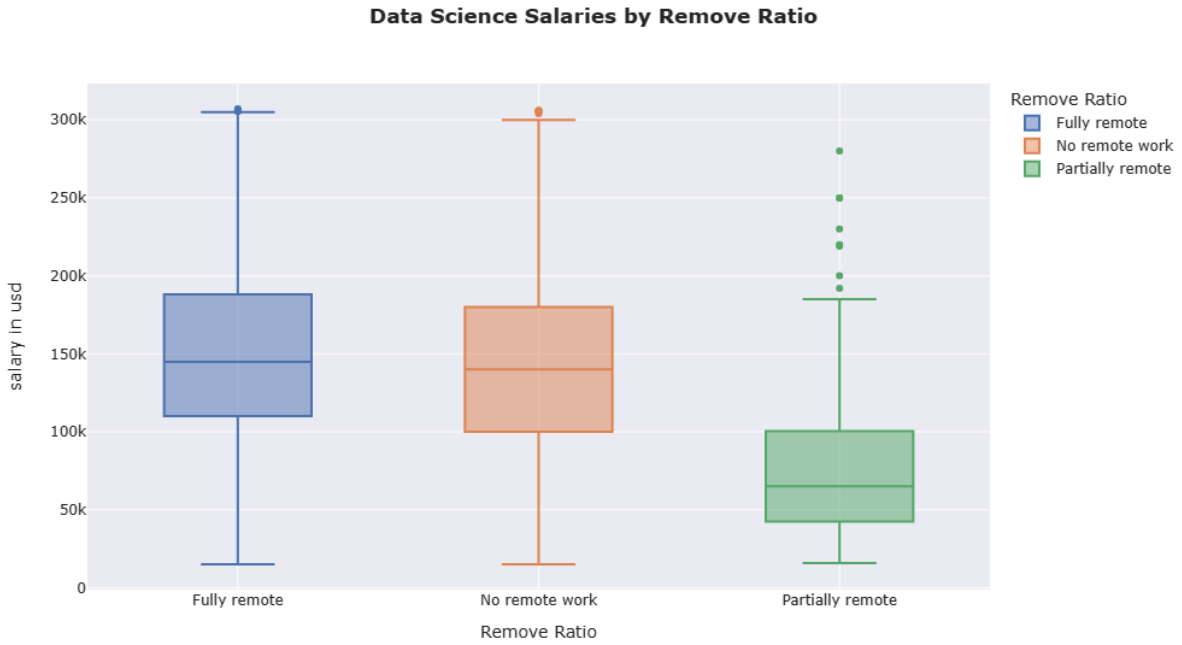


Hình 17: Biểu đồ boxplot theo Salaries và Company Size

Nhận xét: Mức lương trung bình cao nhất được quan sát thấy ở các doanh nghiệp quy mô vừa, trong khi mức lương thấp nhất được thấy ở các doanh nghiệp quy mô nhỏ. Điều này có thể gợi ý rằng các doanh nghiệp cỡ trung bình, thường đang trong giai đoạn tăng trưởng và mở rộng, sẵn sàng đầu tư nhiều hơn vào nhân tài. Mặt khác, các doanh nghiệp nhỏ, có thể có ngân sách hạn chế, có xu hướng đưa ra mức lương trung bình thấp hơn. Ngược lại, các doanh nghiệp quy mô lớn có thể có cơ cấu trả lương ổn định và phân bổ ngân sách chặt chẽ, có khả năng dẫn đến mức lương trung bình tương đối thấp hơn so với các doanh nghiệp quy mô vừa. Tuy nhiên, những phát hiện này cần xem xét lại một cách thận trọng, vì các doanh nghiệp quy mô lớn và doanh nghiệp quy mô nhỏ không được thể hiện đầy đủ trong tập dữ liệu, chỉ chiếm lần lượt 8,7% và 2,1%. Do đó, dữ liệu có thể không phản ánh đầy đủ sự phân bổ tiền lương giữa các quy mô công ty khác nhau.

- **Biểu đồ boxplot theo Salaries và Remove Ratio**

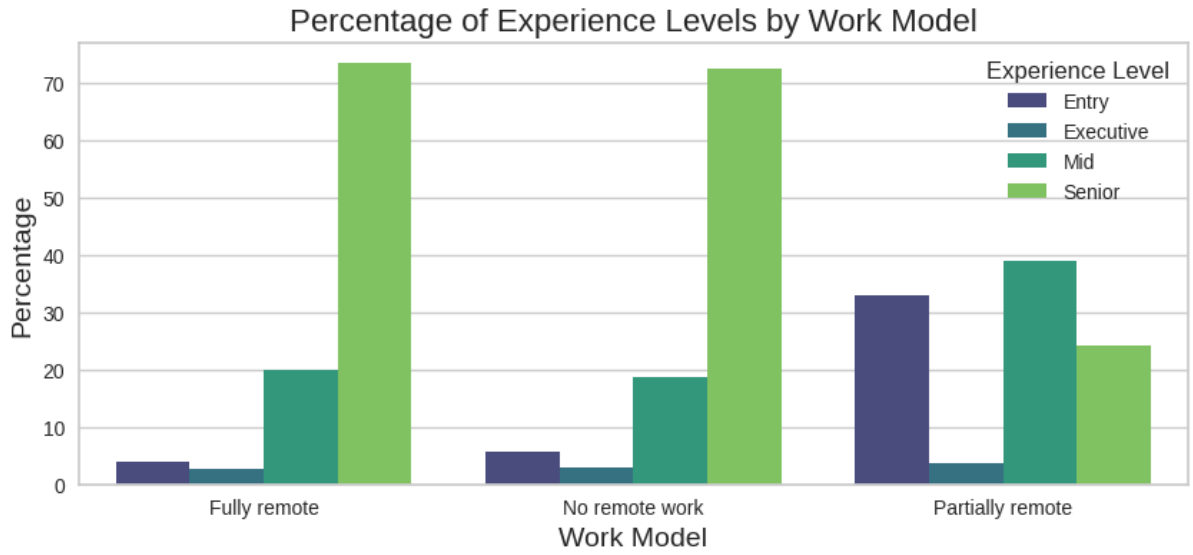
```
## Boxplot of salary_in_usd by Remote ratio
px.box(df,x='remote_ratio',y='salary_in_usd',color='remote_ratio',
       template='seaborn',labels={'remote_ratio':'Remove Ratio',
                                  'salary_in_usd':'salary in usd'},
       title='<b>Data Science Salaries by Remove Ratio',
       width=1000, height=600)
```



Hình 18: Biểu đồ boxplot theo Salaries và Remove Ratio

```
# Nhóm dữ liệu theo mô hình công việc và mức độ kinh nghiệm, tính toán
số lượng cho mỗi kết hợp
grouped_data = df.groupby(['remote_ratio',
'experience_level']).size().reset_index(name='count')

# Tính tổng số cho từng mô hình công việc để tìm tỷ lệ phần trăm sau này
total_counts =
grouped_data.groupby('remote_ratio')['count'].sum().reset_index(name='to
tal_count')

# Merging the total counts back into the grouped data
merged_data = pd.merge(grouped_data, total_counts, on='remote_ratio')

# Calculating the percentage of each experience level for each work
model
merged_data['percentage'] = (merged_data['count'] /
merged_data['total_count']) * 100


# Plotting the percentage of experience levels for each work model
plt.figure(figsize=(10,4))

sns.barplot(data=merged_data, x='remote_ratio', y='percentage',
hue='experience_level', palette='viridis')

plt.title('Percentage of Experience Levels by Work Model', fontsize=16)
plt.ylabel('Percentage', fontsize=14)
plt.xlabel('Work Model', fontsize=14)
plt.tick_params(axis='x')
```

```
plt.legend(title='Experience Level')
plt.show()
```



Hình 19: Percentage of Experience Levels by Work Model

Nhận xét: Biểu đồ cho thấy những vị trí làm việc từ xa có xu hướng đưa ra mức lương trung bình cao hơn một chút so với những vị trí yêu cầu có mặt tại chỗ. Điều này có thể là do các công ty chủ yếu hoạt động từ xa có chi phí thấp hơn, vì họ có thể không cần đầu tư nhiều vào cơ sở vật chất và nguồn lực tại chỗ. Các công ty này có thể có xu hướng phân bổ một phần tiền để chi trả lương cho nhân viên. Ngược lại, các vị trí áp dụng mô hình làm việc kết hợp có mức lương trung bình thấp nhất trong ba loại. Ta thấy rằng tỷ lệ phần trăm vị trí ở trình độ Entry và Mid là cao nhất đối với công việc làm việc từ xa một phần, điều này khá hợp lý. Tuy nhiên, điều quan trọng là phải xem xét điều này một cách thận trọng vì các vị trí làm việc từ xa một phần chỉ chiếm 2,7% dữ liệu. Việc thiếu dữ liệu như vậy có thể làm sai lệch con số lương trung bình.

- **Biểu đồ địa lý mức lương theo company_location**

```
average_salary_by_company =
df.groupby('company_location')['salary_in_usd'].mean().reset_index()
fig =
px.choropleth(locations=average_salary_by_company['company_location'],
                    color=average_salary_by_company['salary_in_usd'],
                    color_continuous_scale=px.colors.sequential.YlGn,
                    # template='plotly_dark',
                    title = 'Biểu đồ địa lý mức lương theo
company_location')

fig.update_layout(font = dict(size= 17, family="Segoe UI"))
```

```
fig.show()
```

Hình 20: Biểu đồ địa lý mức lương theo company_location

Nhận xét: Trong đồ thị này, mức lương trung bình theo quốc gia được vẽ trên bản đồ. Các quốc gia màu xanh trời nhạt là những quốc gia không có thông tin về công việc dữ liệu ở đó. Chúng ta có thể thấy rằng mức lương cao nhất trên toàn thế giới là ở Bắc Mỹ (Mỹ, Canada và Mexico), ở Trung Đông (cụ thể là Qatar, Israel và Ả Rập Saudi) và ở Châu Đại Dương (Úc và New Zealand). Cần nhiều dữ liệu hơn để phân tích tốt hơn về khía cạnh này vì có nhiều quốc gia không có thông tin và nhiều quốc gia chỉ có 1 hoặc 2 việc làm (vì vậy mức lương không đại diện cho những gì đang xảy ra ở quốc gia đó).

- **Tổng kết**

  - Tập dữ liệu chủ yếu đại diện cho năm 2023 (76,1%) và các doanh nghiệp vừa (89,2%). Các vị trí từ xa và cấp cao đặc biệt phổ biến, chiếm lần lượt 58,8% và 72,2% dữ liệu.
  - Thông tin chi tiết về phân phối tiền lương: Dữ liệu tiền lương hơi lệch trái với mức lương trung bình khoảng 150 nghìn USD và độ lệch chuẩn là khoảng 64 nghìn USD. Sự phân bổ không đồng đều trong một số số liệu nhất định cho thấy các khu vực cần khám phá chi tiết.
  - Các công việc liên quan đến dữ liệu không ngừng tăng lên trong những năm qua, số lượng việc làm tăng theo cấp số nhân và mức lương thực sự tốt và cũng tăng theo thời gian.
  - Những công việc phổ biến nhất không phải là những công việc được trả nhiều tiền nhất mà trung bình tất cả các vị trí đều được trả hơn 100 nghìn USD mỗi năm.

"Data Scientist", "Data Engineer" và "Data Analyst" là 3 công việc phổ biến hơn và chúng không ngừng gia tăng.

- Nếu ai đó muốn có công việc với mức lương cao hơn có thể, anh ấy/cô ấy phải tìm kiếm một vị trí hoàn toàn từ xa hoặc hoàn toàn chuyên nghiệp, trong một công ty quy mô vừa hoặc lớn, với vị trí toàn thời gian hoặc làm việc theo hợp đồng, bán thời gian. công việc hoặc một công việc tự do nó không mang lại lợi nhuận. Và lý tưởng nhất là công việc phải ở Bắc Mỹ, Trung Đông hoặc Châu Đại Dương.
- Mối tương quan giữa trình độ kinh nghiệm và tiền lương: Có một mối liên hệ vừa phải giữa kinh nghiệm của một cá nhân và mức lương của họ, cho thấy rằng nhiều kinh nghiệm hơn thường dẫn đến mức lương cao hơn. Tuy nhiên, có những điểm mâu thuẫn được lưu ý, chẳng hạn như vai trò 'Data Analyst' không nhất thiết phải có thâm niên cao hơn nhưng vẫn được trả lương cao, cho thấy giá trị của các kỹ năng chuyên môn.
- Ảnh hưởng của mô hình làm việc đến lương: Các vị trí từ xa dẫn đầu về mức lương trung bình đưa ra một chút, có thể phản ánh khả năng tiết kiệm chi phí của công ty. Ngược lại, mô hình làm việc từ xa một phần, chiếm 2,7% tập dữ liệu, có mức lương trung bình thấp nhất, kết quả có thể bị ảnh hưởng bởi sự chiếm ưu thế của các vị trí cấp đầu vào trong danh mục này.
- Tác động của quy mô công ty đến tiền lương: Các doanh nghiệp quy mô vừa nổi lên như những công ty trả lương cao nhất, theo sau là các doanh nghiệp quy mô nhỏ. Các doanh nghiệp lớn có mức lương trung bình thấp hơn so với các doanh nghiệp vừa. Tuy nhiên, sự thiếu đại diện của các công ty lớn (8,7%) và quy mô nhỏ (2,1%) trong dữ liệu cảnh báo việc đưa ra các kết luận sâu rộng.
- Chức danh công việc và Mức lương thay đổi: Sự chênh lệch về lương rõ rệt được ghi nhận giữa các chức danh công việc, với các vai trò như "Nhà khoa học nghiên cứu" và "Kỹ sư ML" vượt xa đáng kể về mức lương "Người quản lý dữ liệu" và "Nhà phân tích dữ liệu". Điều này chỉ ra rằng các kỹ năng và trách nhiệm chuyên môn có thể đòi hỏi phí bảo hiểm thị trường cao hơn.
- Sự thay đổi mức lương dựa trên vị trí: Hoa Kỳ, Canada và Vương quốc Anh là những khu vực hàng đầu về mức lương trung bình.
- Mặc dù các xu hướng và mối tương quan nhất định xuất hiện, chẳng hạn như kinh nghiệm ảnh hưởng đến mức lương và các vị trí từ xa được trả lương cao hơn

một chút, nhưng những hạn chế của tập dữ liệu đòi hỏi phải diễn giải thận trọng. Các nghiên cứu bổ sung bao gồm một tập hợp rộng hơn các yếu tố như ngành, vị trí địa lý và quy mô công ty có thể cung cấp sự hiểu biết toàn diện hơn về động thái tiền lương được quan sát.

4. **Thu gọn dữ liệu**

Ta tiến hành bỏ cột employment_type vì cột này biến thiên quá ít

```
df.drop(df[['employment_type']],axis=1, inplace=True)
```

Bỏ cột employee_residence vì cột này có tương quan cao với cột company_location.

```
df.drop(df[['employee_residence']],axis=1, inplace=True)
```

5. **Chuyển dạng dữ liệu**

Các giá trị dữ liệu biến đổi cụ thể như sau:

Company_location	USA → 1 Other → 0
Company_size	Small → 1 Medium → 2 Large → 3
Experience_level	Entry → 1 Mid → 2 Senior → 3 Executive → 4
Job_title (Random Forest)	Data Analyst → 0 Data Engineer → 1 Data Scientist → 2 Machine Learning Engineer → 3 Other → 4
Job_title (Linear Regression và SVM)	Dummy Encoding

Ta thực hiện các bước chuyển dạng sau:

```
# company_location
def location(title):
    if any(keyword in title.lower() for keyword in ['usa']):
```

```
        return 1
    else:
        return 0
df['usa'] = df['company_location'].apply(location)
```

```
# Company size
mapping_dict = {'Small': 1, 'Medium': 2, 'Large': 3}
df['company_size_clean'] = df['company_size'].map(mapping_dict)
```

```
# experience_level
mapping_dict = {'Entry': 1, 'Mid': 2, 'Senior': 3, 'Executive': 4}
df['experience_level_clean'] =
df['experience_level'].map(mapping_dict)
```

```
# loại bỏ những cột đã được thay bằng cột mới
df.drop(df[['experience_level', 'job_title', 'remote_ratio',
'company_location', 'company_size']],axis=1, inplace = True)
```

Sau khi chuyển dạng dữ liệu sang cột mới, ta tiến hành drop các cột categorical cũ.

5.1. Chuyển dạng dữ liệu dùng cho mô hình Random Forest

```
# job_title_clean cho mô hình rừng ngẫu nhiên
enc             = LabelEncoder()
job_title_rd  = pd.Series(enc.fit_transform(df['job_title_clean']))
job_title_rd  = pd.DataFrame(job_title_rd, columns =
['job_title_rd'])
print(job_title_rd)

## Bảng mã
print(job_title_rd.nunique())
mappings = {index: label for index, label in enumerate(enc.classes_)}
print(mappings)
```

```
      job_title_rd
0                2
1                2
2                4
3                4
4                3
...            ...
8641             4
8642             2
8643             2
8644             0
8645             2

[8646 rows x 1 columns]
job_title_rd    5
dtype: int64
{0: 'Data Analyst', 1: 'Data Engineer', 2: 'Data Scientist', 3: 'Machine Learning Engineer', 4: 'Other'}
```

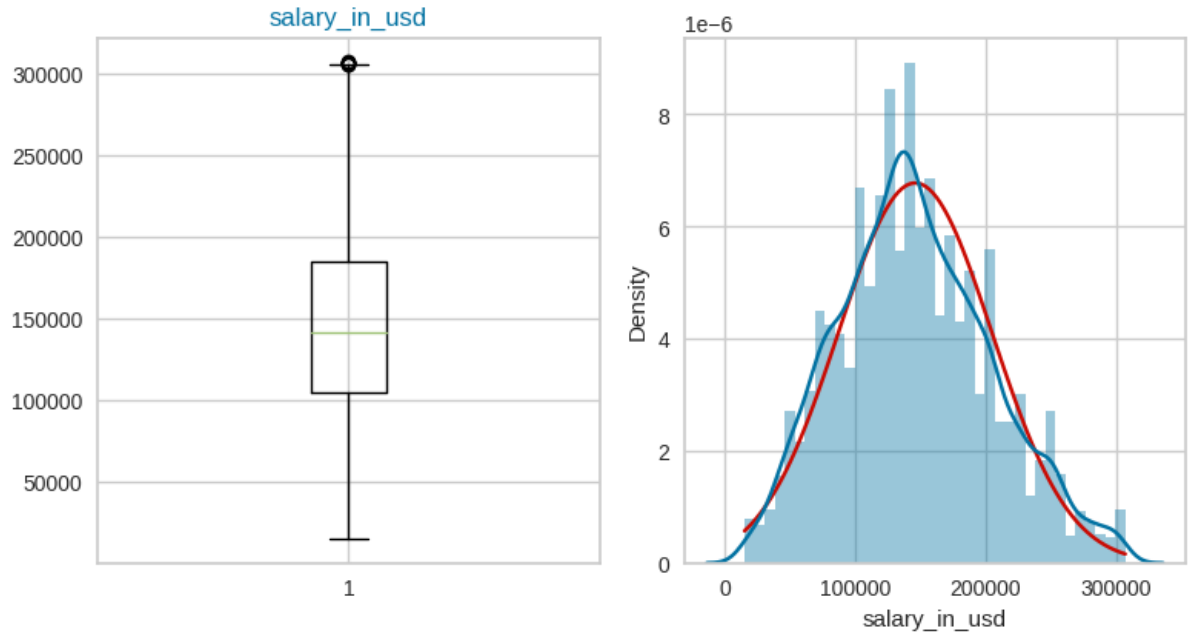
Ta sử dụng Label Encoding để đưa các ngành thành dạng số như: {'*Data Analyst*': 0}; {'*Data Engineer*': 1}; {'*Data Scientist*': 2}; {'*Machine Learning Engineer*': 3}; {'*Other*': 4}

Sau đó ta tiến hành kết hợp features trong bộ dữ liệu đã xử lý với dữ liệu của df_rd, khi đó ta xóa cột 'job_title_clean' để đảm bảo bộ dữ liệu chỉ còn biến số.

- **Kiểm tra bộ dữ liệu để quyết định sử dụng chuẩn hóa MinMaxScaler hay StandardScaler**

Ta tiến hành kiểm tra bộ dữ liệu có phân phối chuẩn hay không, còn outliers nữa hay không. Vì bộ dữ liệu chỉ có một biến số là salary_in_usd nên ta chỉ xét biến này.

```
# Kiểm tra KHÔNG có phân phối CHUẨN, KHÔNG có outliers
fig = plt.figure(figsize = ([8, 8]))

plt.subplot(2, 2, 1)
plt.title('salary_in_usd', color = 'b')
plt.boxplot(df['salary_in_usd'])

plt.subplot(2, 2, 2)
# Tính mean và standard deviation của dữ liệu
mean, std_dev = np.mean(df['salary_in_usd']),
np.std(df['salary_in_usd'])

# Tạo dải giá trị để vẽ đường phân phối chuẩn
x_range = np.linspace(min(df['salary_in_usd']),
max(df['salary_in_usd']), 100)
normal_distribution = norm.pdf(x_range, mean, std_dev)

# Vẽ đường phân phối chuẩn
plt.plot(x_range, normal_distribution, 'r', label='Normal
Distribution')

sns.distplot(df['salary_in_usd'], kde = True, hist = True)

plt.tight_layout()
plt.show()
```



Hình 21: Kiểm tra phân phối chuẩn và outliers

Từ biểu đồ bên trái, ta thấy cột salary_in_usd không chứa outliers.

Từ biểu đồ bên phải, ta thấy đường đường cong phân phối đặc trưng của cột salary_in_usd (được biểu diễn bằng màu xanh) gần với đường phân phối chuẩn (được biểu diễn bằng màu đỏ), nên thực hiện kiểm định phân phối chuẩn shapiro-wilk:

```
# Kiểm định shapiro phân phối chuẩn
statistic, p_value = shapiro(df['salary_in_usd'])

# In kết quả
print(f'Statistic: {statistic}, p-value: {p_value}')

# Kiểm tra giả định null
if p_value > 0.05:
    print('Không có bằng chứng để bác bỏ giả định null.')
else:
    print('Bác bỏ giả định null, mẫu dữ liệu không tuân theo phân
phối chuẩn.')
```

```
Statistic: 0.9901731610298157, p-value: 5.226472893088336e-24
Bác bỏ giả định null, mẫu dữ liệu không tuân theo phân phối chuẩn.
```

Ta có kết luận mẫu dữ liệu không tuân theo phân phối chuẩn và không còn Outliers. Như vậy, ta tiến hành chuẩn hóa bằng MinMaxScaler, đưa các giá trị về khoảng từ 0 đến 1.

```
## Chuẩn hóa dữ liệu MinMaxScaler() vì không có phân phối chuẩn và
outliers
scaler       = MinMaxScaler()
MinMax_array = scaler.fit_transform(df_rd) # dạng array
```

```
df_rd     = pd.DataFrame(MinMax_array, columns = df_rd.columns)
df_rd
```

	work_year	salary_in_usd	remote_ratio_clean	usa	company_size_clean	experience_level_clean	job_title_rd
0	1.000000	0.674427	0.0	1.0	0.5	1.000000	0.50
1	1.000000	0.599110	0.0	1.0	0.5	1.000000	0.50
2	1.000000	0.096077	0.0	0.0	0.5	0.333333	1.00
3	1.000000	0.096077	0.0	0.0	0.5	0.333333	1.00
4	1.000000	0.789798	0.0	1.0	0.5	0.666667	0.75
...	...	...	...	...	...	...	...
8641	0.333333	0.513523	1.0	1.0	1.0	0.666667	1.00
8642	0.333333	0.465594	1.0	1.0	1.0	0.333333	0.50
8643	0.000000	0.308114	1.0	1.0	0.0	0.000000	0.50
8644	0.000000	0.290996	1.0	1.0	1.0	0.000000	0.00
8645	0.333333	0.272732	0.5	0.0	1.0	0.666667	0.50

5.2. Chuyển dạng dữ liệu dùng cho mô hình Linear Regression và SVM

```
# job_title_clean cho mô hình hồi quy và svm
job_title_lr_svm = pd.get_dummies(df['job_title_clean'])
job_title_lr_svm.drop(job_title_lr_svm[['Other']],axis=1, inplace =
True)

# Ghép bộ dữ liệu gốc với cột job_title được dummy và bỏ đi dư thừa
df_lr_svm  = pd.concat([df, job_title_lr_svm], axis = 1)
df_lr_svm.drop(df_lr_svm[['job_title_clean']],axis=1, inplace = True)
```

Thay vì sử dụng Label Encoding như ở trên, trong trường hợp này ta sử dụng phương pháp Dummy Encoding để đưa cột job_title_clean thành dạng số. Khi đó, kết hợp 2 bộ dữ liệu và tiến hành xóa cột job_title_clean như cách trên ta đã làm.

- **Chuẩn hóa MinMaxScaler**

```
## Chuẩn hóa dữ liệu MinMaxScaler() vì không có phân phối chuẩn và
outliers
scaler       = MinMaxScaler()
MinMax_array = scaler.fit_transform(df_lr_svm) # dạng array
df_lr_svm    = pd.DataFrame(MinMax_array, columns =
df_lr_svm.columns)
df_lr_svm
```

	work_year	salary_in_usd	remote_ratio_clean	usa	company_size_clean	experience_level_clean	Data Analyst	Data Engineer	Data Scientist	Machine Learning Engineer
0	1.000000	0.674427	0.0	1.0	0.5	1.000000	0.0	0.0	1.0	0.0
1	1.000000	0.599110	0.0	1.0	0.5	1.000000	0.0	0.0	1.0	0.0
2	1.000000	0.096077	0.0	0.0	0.5	0.333333	0.0	0.0	0.0	0.0
3	1.000000	0.096077	0.0	0.0	0.5	0.333333	0.0	0.0	0.0	0.0
4	1.000000	0.789798	0.0	1.0	0.5	0.666667	0.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...
8641	0.333333	0.513523	1.0	1.0	1.0	0.666667	0.0	0.0	0.0	0.0
8642	0.333333	0.465594	1.0	1.0	1.0	0.333333	0.0	0.0	1.0	0.0
8643	0.000000	0.308114	1.0	1.0	0.0	0.000000	0.0	0.0	1.0	0.0
8644	0.000000	0.290996	1.0	1.0	1.0	0.000000	1.0	0.0	0.0	0.0
8645	0.333333	0.272732	0.5	0.0	1.0	0.666667	0.0	0.0	1.0	0.0

- **Giảm chiều bằng phương pháp Principal Component Analysis (PCA)**

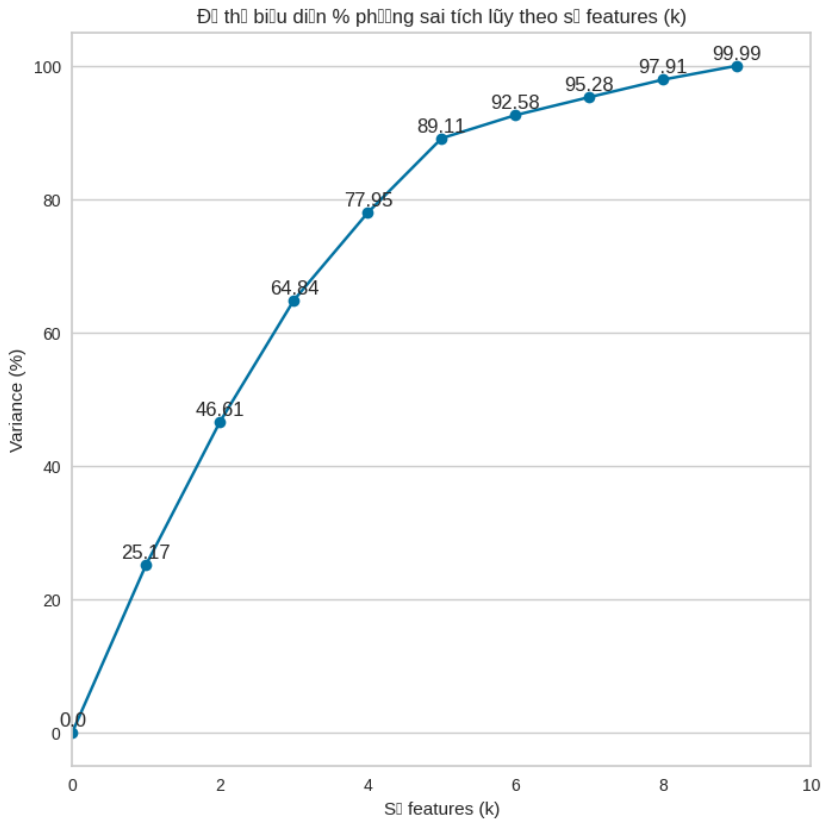
Kiểm tra biến target và các features để tiến hành giảm chiều

```
## Mô tả dữ liệu
## Biến phân lớp (target variable): 'Class' --> cột cuối cùng trong
file
target = 'salary_in_usd'
print('* Biến phân lớp:', target)

## Danh sách các features
nb_features = df_lr_svm.shape[1] - 1
features    = df_lr_svm.drop(target, axis=1).columns
print('* Số lượng features = %2d' %nb_features)
print('  Các features:', ', '.join(features))
```

```
* Biến phân lớp: salary_in_usd
* Số lượng features =  9
  Các features: work_year, remote_ratio_clean, usa,
company_size_clean, experience_level_clean, Data Analyst, Data
Engineer, Data Scientist, Machine Learning Engineer
```

Ta tiến hành tìm số chiều k bằng cách biểu diễn % phương sai sau đó tiến hành chọn k theo điểm "gẫy".

```
## Áp dụng PCA (chưa xác định k --> giữ nguyên số chiều)
pca = PCA().fit(df_lr_svm[features])
## Vẽ đồ thị biểu diễn % phương sai tích lũy theo số features -->
chọn k theo điểm "gẫy"
points = np.cumsum(pca.explained_variance_ratio_) * 100 # Các điểm dữ
liệu
points = np.insert(points, 0, 0) # Thêm điểm k = 0, variance = 0
x_i = np.arange(0, nb_features + 1)
y_i = (points[-13:])//0.01/100

plt.figure(figsize = (8, 8))
plt.plot(points, marker = 'o')
plt.xlabel('Số features (k)')
plt.ylabel('Variance (%)')
plt.title('Đồ thị biểu diễn % phương sai tích lũy theo số features
(k)')
plt.xlim([0, nb_features + 1])
plt.grid(axis = 'x')
for i in x_i:
    plt.text(i, y_i[i] + 1, y_i[i], ha = 'center', va = 'baseline') #
tung độ của text cao hơn point 1 đơn vị

plt.show()
```



Hình 22: Đồ thị biểu diễn phần trăm phương sai tích lũy theo số features (k)

Ta tiến hành kiểm chứng, ta tính phương sai tích lũy theo giá trị của k

```
## Kiểm chứng: Tính phương sai tích lũy theo giá trị của k
var = 0.0
for k in range(1, nb_features + 1):
    pca = PCA(k)
    pca.fit(df_lr_svm[features])

    newVar = pca.explained_variance_ratio_.sum() * 100
    print('   * k = %2d' %k, ': phương sai tích lũy ~ %.2f%%'
%newVar,
          '--> tăng ~ %.2f%%' %(newVar - var))
    var = newVar
```

```
   * k =  1 : phương sai tích lũy ~ 25.18% --> tăng ~ 25.18%
   * k =  2 : phương sai tích lũy ~ 46.61% --> tăng ~ 21.44%
   * k =  3 : phương sai tích lũy ~ 64.84% --> tăng ~ 18.23%
   * k =  4 : phương sai tích lũy ~ 77.96% --> tăng ~ 13.12%
   * k =  5 : phương sai tích lũy ~ 89.12% --> tăng ~ 11.16%
   * k =  6 : phương sai tích lũy ~ 92.59% --> tăng ~ 3.47%
   * k =  7 : phương sai tích lũy ~ 95.29% --> tăng ~ 2.70%
   * k =  8 : phương sai tích lũy ~ 97.92% --> tăng ~ 2.63%
   * k =  9 : phương sai tích lũy ~ 100.00% --> tăng ~ 2.08%
```

Ta chọn k = 6, thực hiện PCA với k = 6.

```
## Thực hiện PCA với k = 6
k   = 6
pca = PCA(k)
pca.fit(df_lr_svm[features])
## Chiếu dữ liệu vào không gian mới (Transform data)
PC_name = []
for i in range(k):
  name = f'Principal Component {i+1}'
  PC_name.append(name)
P            = pca.transform(df_lr_svm[features])
principalDf_lr_svm = pd.DataFrame(data = P, columns = PC_name)
df_lr_svm_pca  = pd.concat([principalDf_lr_svm,
df_lr_svm['salary_in_usd']], axis = 1)
```

Sau khi tiến hành giảm chiều, ta có được dữ liệu như sau:

	Principal Component 1	Principal Component 2	Principal Component 3	Principal Component 4	Principal Component 5	Principal Component 6	salary_in_usd
0	0.692303	-0.511517	-0.122474	-0.122468	-0.156623	-0.352753	0.674427
1	0.692303	-0.511517	-0.122474	-0.122468	-0.156623	-0.352753	0.599110
2	-0.104217	-0.240063	0.178701	0.194139	0.889088	-0.082022	0.096077
3	-0.104217	-0.240063	0.178701	0.194139	0.889088	-0.082022	0.096077
4	-0.205225	-0.420525	0.374262	0.836680	-0.230115	-0.016253	0.789798
...	...	...	...	...	...	...	...
8641	-0.017115	0.624865	-0.038075	0.107946	-0.185831	0.161137	0.513523
8642	0.794612	0.529516	-0.304791	-0.062577	-0.086642	0.523027	0.465594
8643	0.770422	0.576648	-0.292404	-0.066863	-0.011035	0.802182	0.308114
8644	-0.161960	0.868973	0.629439	-0.408544	0.060526	0.886747	0.290996
8645	0.777657	0.176535	-0.237554	0.071900	0.875060	0.003323	0.272732

5.3. Tiền xử lý cho mô hình phân cụm

- **Giảm chiều cho mô hình phân cụm**

Ta tạo 1 biến khác là df_km_hac để tiền hành giảm chiều PCA:

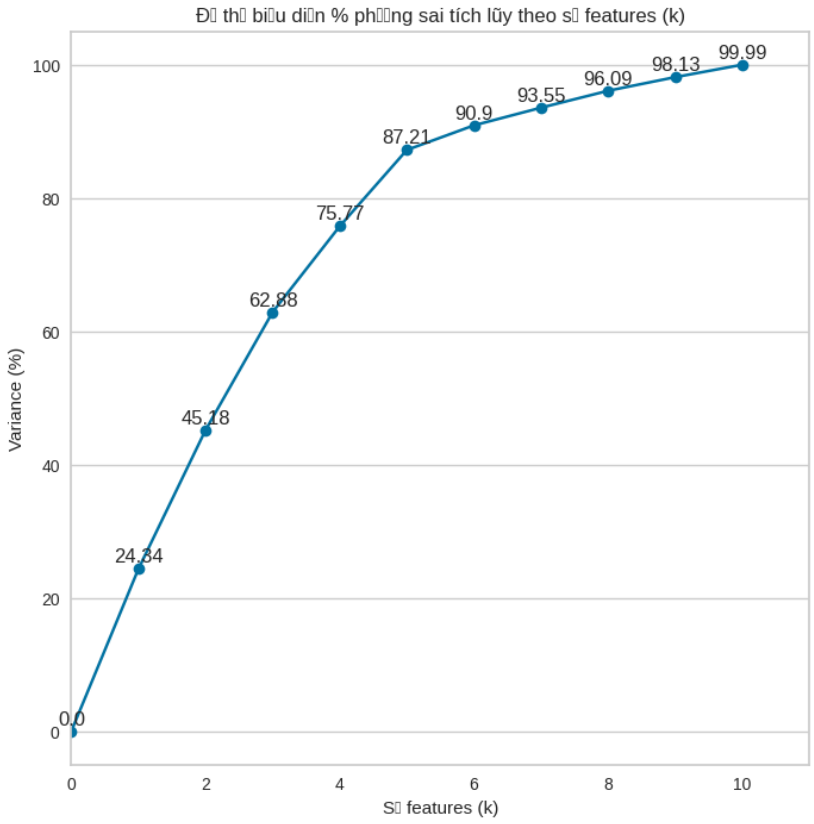
```
## Mô tả dữ liệu
df_km_hac = df_lr_svm.copy()
## Phân cụm không sử dụng biến target
## Danh sách các features
nb_features = df_km_hac.shape[1]
features    = df_km_hac.columns
print('* Số lượng features = %2d' %nb_features)
print('  Các features:', ', '.join(features))
```

```
* Số lượng features = 10
  Các features: work_year, salary_in_usd, remote_ratio_clean, usa,
company_size_clean, experience_level_clean, Data Analyst, Data
Engineer, Data Scientist, Machine Learning Engineer
```

Ta tiến hành tìm số chiều k bằng cách biểu diễn % phương sai sau đó tiến hành chọn k theo điểm "gẫy".

```
## Áp dụng PCA (chưa xác định k --> giữ nguyên số chiều)
pca = PCA().fit(df_km_hac[features])
```

```
## Vẽ đồ thị biểu diễn % phương sai tích lũy theo số features -->
chọn k theo điểm "gẫy"
points = np.cumsum(pca.explained_variance_ratio_) * 100 # Các điểm dữ
liệu
points = np.insert(points, 0, 0) # Thêm điểm k = 0, variance = 0
x_i = np.arange(0, nb_features + 1)
y_i = (points[-13:])//0.01/100

plt.figure(figsize = (8, 8))
plt.plot(points, marker = 'o')
plt.xlabel('Số features (k)')
plt.ylabel('Variance (%)')
plt.title('Đồ thị biểu diễn % phương sai tích lũy theo số features
(k)')
plt.xlim([0, nb_features + 1])
plt.grid(axis = 'x')
for i in x_i:
    plt.text(i, y_i[i] + 1, y_i[i], ha = 'center', va = 'baseline') #
tung độ của text cao hơn point 1 đơn vị

plt.show()
```



Hình 23: Biểu đồ biểu diễn % phương sai tích lũy theo số features (k) – phân cụm

Ta tiến hành kiểm chứng, ta tính phương sai tích lũy theo giá trị của k

```
## Kiểm chứng: Tính phương sai tích lũy theo giá trị của k
var = 0.0
for k in range(1, nb_features + 1):
    pca = PCA(k)
```

```
    pca.fit(df_km_hac[features])

    newVar = pca.explained_variance_ratio_.sum() * 100
    print('   * k = %2d' %k, ': phương sai tích lũy ~ %.2f%%'
%newVar,
          '--> tăng ~ %.2f%%' %(newVar - var))
    var = newVar
```

```
* k =  1 : phương sai tích lũy ~ 24.35% --> tăng ~ 24.35%
* k =  2 : phương sai tích lũy ~ 45.18% --> tăng ~ 20.83%
* k =  3 : phương sai tích lũy ~ 62.89% --> tăng ~ 17.70%
* k =  4 : phương sai tích lũy ~ 75.77% --> tăng ~ 12.89%
* k =  5 : phương sai tích lũy ~ 87.21% --> tăng ~ 11.44%
* k =  6 : phương sai tích lũy ~ 90.90% --> tăng ~ 3.69%
* k =  7 : phương sai tích lũy ~ 93.55% --> tăng ~ 2.65%
* k =  8 : phương sai tích lũy ~ 96.09% --> tăng ~ 2.54%
* k =  9 : phương sai tích lũy ~ 98.14% --> tăng ~ 2.05%
* k = 10 : phương sai tích lũy ~ 100.00% --> tăng ~ 1.86%
```

Ta chọn k = 6, thực hiện PCA với k = 6

```
## Thực hiện PCA với k = 6
k    = 6
pca = PCA(k)
pca.fit(df_km_hac[features])
## Chiếu dữ liệu vào không gian mới (Transform data)
PC_name = []
for i in range(k):
  name = f'Principal Component {i+1}'
  PC_name.append(name)
P            = pca.transform(df_km_hac[features])
df_km_hac_pca = pd.DataFrame(data = P, columns = PC_name)
```

	Principal Component 1	Principal Component 2	Principal Component 3	Principal Component 4	Principal Component 5	Principal Component 6
0	0.719852	-0.516875	-0.110809	-0.075257	-0.198858	-0.362722
1	0.715500	-0.509512	-0.103909	-0.088494	-0.180407	-0.331234
2	-0.128909	-0.180942	0.246076	-0.064220	0.964890	-0.006685
3	-0.128909	-0.180942	0.246076	-0.064220	0.964890	-0.006685
4	-0.170667	-0.461942	0.330709	0.918387	-0.090053	-0.085494
...	...	...	...	...	...	...
8641	-0.026220	0.606789	-0.082945	0.173445	-0.182852	0.128059
8642	0.783618	0.538245	-0.317691	-0.053520	-0.081408	0.451412
8643	0.747364	0.603831	-0.289059	-0.104188	0.032348	0.778523
8644	-0.204256	0.908782	0.609607	-0.378532	-0.065085	0.726479
8645	0.755120	0.228206	-0.189061	-0.165121	0.895686	-0.066486

- **Giảm chiều để biểu diễn trực quan mô hình phân cụm**

Để có thể quan sát trực quan bằng đồ thị, ta đưa mô hình về 2 chiều để có thể biểu diễn trên trục tọa độ.

```
## Thực hiện PCA với k = 2
k    = 2
pca = PCA(k)
pca.fit(df_km_hac_pca)
```

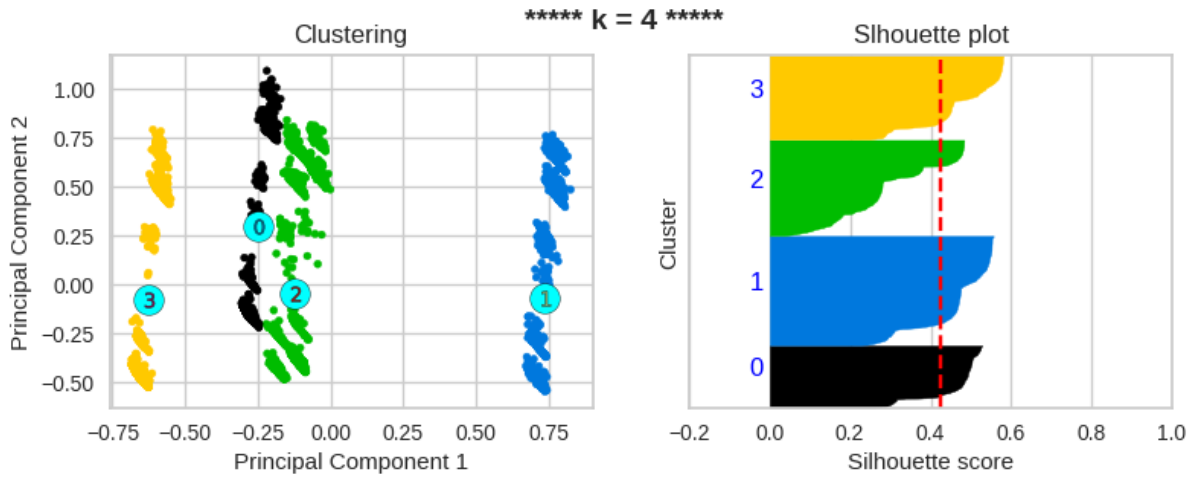
```
## Chiếu dữ liệu vào không gian mới (Transform data)
PC_name = []
for i in range(k):
  name = f'Principal Component {i+1}'
  PC_name.append(name)
P            = pca.transform(df_km_hac_pca)
df_km_hac_pca_vs = pd.DataFrame(data = P, columns = PC_name)
df_km_hac_pca_vs
```

	Principal Component 1	Principal Component 2
0	0.719852	-0.516875
1	0.715500	-0.509512
2	-0.128909	-0.180942
3	-0.128909	-0.180942
4	-0.170667	-0.461942
...	...	...
8641	-0.026220	0.606789
8642	0.783618	0.538245
8643	0.747364	0.603831
8644	-0.204256	0.908782
8645	0.755120	0.228206

CHƯƠNG 4: XÂY DỰNG MÔ HÌNH

1. Mô hình phân cụm

1.1. K-Means



Hình 24: Biểu đồ trực quan kết quả phân cụm K-Means (k=4)

Thực hiện phân lớp bằng phương pháp K-Means với k = 4

```
k      = 4
model_km = KMeans(n_clusters = k)
model_km.fit(df_km_hac_pca)
## Các trọng tâm
print(f'*** {k} trọng tâm:')
print(model_km.cluster_centers_)
```

```
*** 4 trọng tâm:
[[-0.62879982 -0.07859973 -0.54430292 -0.14612286  0.01389453  0.0027938 ]
 [-0.24950896  0.29606826  0.74909044 -0.38279318 -0.08306995 -0.02402971]
 [ 0.73457154 -0.06840214 -0.1675631  -0.09281139  0.00984812  0.00644237]
 [-0.12302699 -0.04414209  0.18668708  0.48260579  0.03025401  0.00572764]]
```

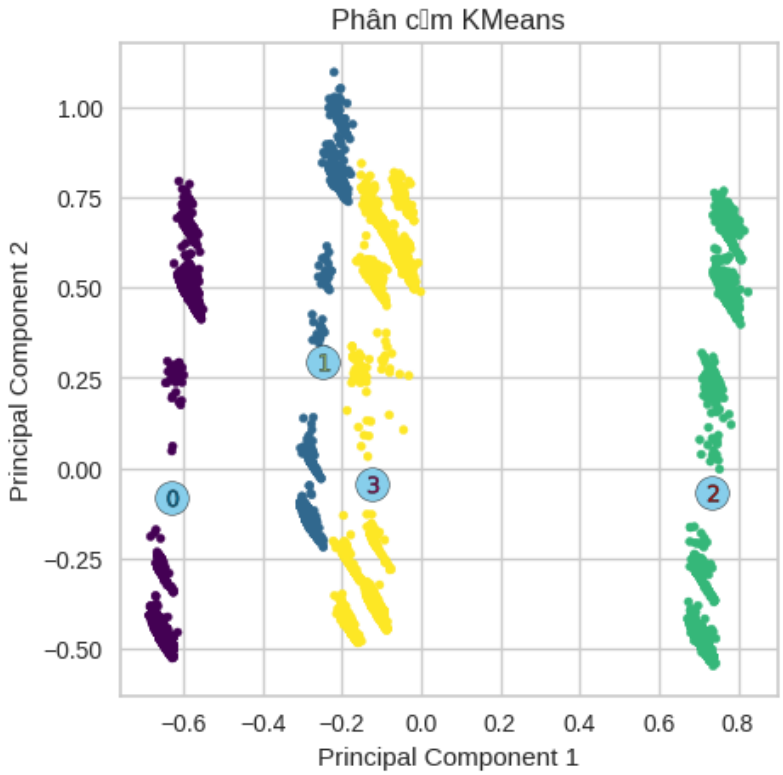
Lúc này ta được kết quả sau khi phân cụm.

```
## Kết quả gom cụm
df_kmeans = df_km_hac_pca.copy()
df_kmeans['cluster'] = model_km.labels_
print(df_kmeans.head(10))

print('\nKích thước các clusters:')
print(df_kmeans['cluster'].value_counts())
print(f'Inertia / #samples = {model_km.inertia_/data.shape[0]:.2f}')
```

```
   Principal Component 1  Principal Component 2  Principal Component 3  \
0               0.719852              -0.516875              -0.110809
1               0.715500              -0.509512              -0.103909
2              -0.128909              -0.180942               0.246076
3              -0.128909              -0.180942               0.246076
4              -0.170667              -0.461942               0.330709
5              -0.193102              -0.423990               0.366278
6              -0.131924              -0.355556               0.218771
7              -0.133902              -0.352209               0.221908
8              -0.177334              -0.450663               0.341280
9              -0.200818              -0.410937               0.378511

   Principal Component 4  Principal Component 5  Principal Component 6  \
0              -0.075257              -0.198858              -0.362722
1              -0.088494              -0.180407              -0.331234
2              -0.064220               0.964890              -0.006685
3              -0.064220               0.964890              -0.006685
4               0.918387              -0.090053              -0.085494
5               0.850155               0.005057               0.076810
6              -0.000175               0.012865               0.276604
7              -0.006191               0.021252               0.290917
8               0.898110              -0.061788              -0.037261
9               0.826688               0.037766               0.132628

   cluster
0        2
1        2
2        3
3        3
4        3
5        3
6        3
7        3
8        3
9        3

Kích thước các clusters:
2    2690
3    2355
0    2077
1    1524
Name: cluster, dtype: int64
Inertia / #samples = 1324.56
```

Biểu diễn trực quan sau khi tiến hành phân cụm

```
# Biểu diễn trực quan
plt.figure(figsize = (5, 5))

plt.title('Phân cụm KMeans')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.scatter(df_km_hac_pca_vs['Principal Component 1'],
df_km_hac_pca_vs['Principal Component 2'], marker = '.', c =
df_kmeans['cluster'], cmap='viridis')

# # Các trọng tâm
centroids = model_km.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1], marker = 'o', c =
'skyblue', alpha = 1, s = 200, edgecolor = 'k')
for i, center in enumerate(centroids):
    plt.scatter(center[0], center[1], marker = '$%d$' % i, alpha = 1,
s = 50, edgecolor = 'k')
plt.show()
```

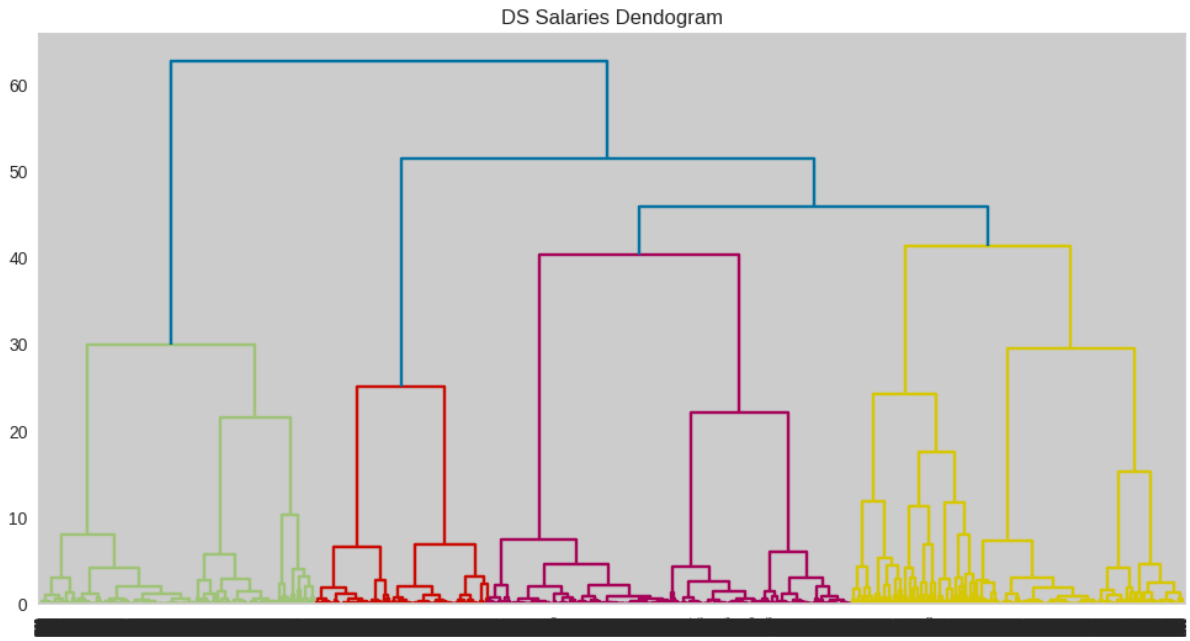


Hình 25: Biểu đồ trực quan các cụm được gom theo K-Means (k=4)

1.2. Hierarchical Agglomerative Clustering (HAC)

Xây dựng Dendrogram

```
# Xây dựng Dendrogram
plt.figure(figsize = (12, 6))
plt.title("DS Salaries Dendogram")
dg = hierarchy.dendrogram(hierarchy.linkage(df_km_hac_pca,
                                            method = 'ward'))


plt.axhline(y = 200, color = 'r', linestyle = '--')
plt.show()
```



Hình 26: Dendogram

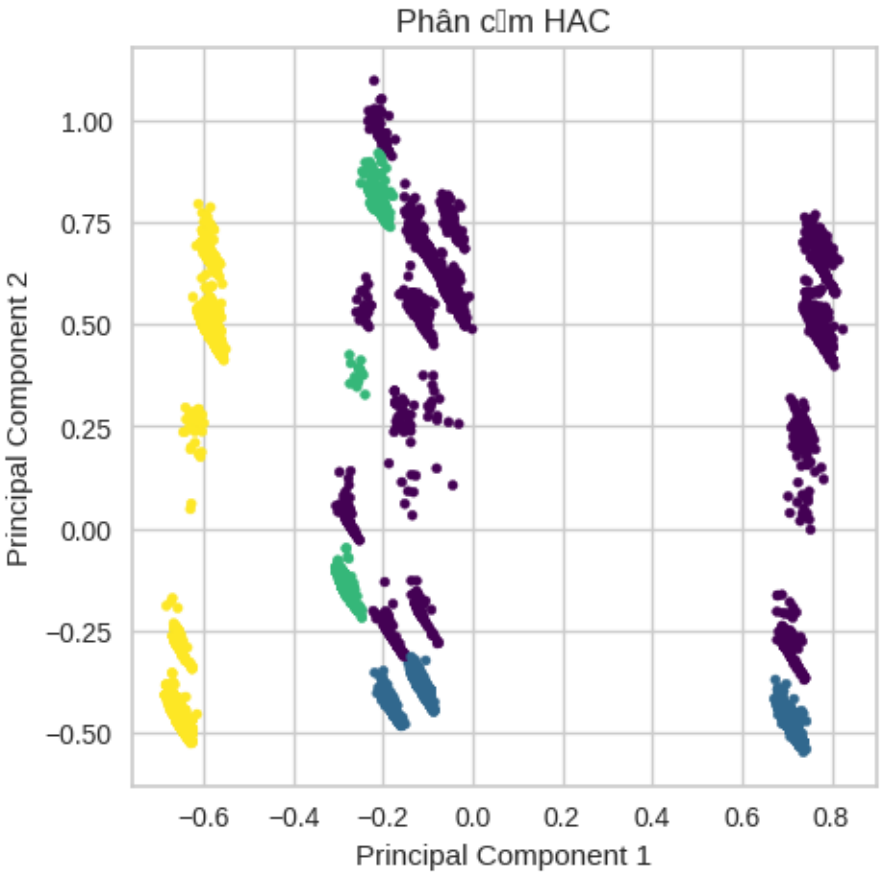
```
## Xây dựng mô hình AgglomerativeClustering
k      = 4
model_hac = AgglomerativeClustering(n_clusters = k, metric =
'euclidean', linkage = 'ward')
model_hac.fit(df_km_hac_pca)
```

```
## Kết quả gom cụm
df_hac = df_km_hac_pca.copy()
df_hac['cluster'] = model_hac.labels_
print(df_hac.head(10))

print('\nKích thước các clusters:')
print(df_hac['cluster'].value_counts())
```

```
   Principal Component 1  Principal Component 2  Principal Component 3  \
0               0.719852              -0.516875              -0.110809
1               0.715500              -0.509512              -0.103909
2              -0.128909              -0.180942               0.246076
3              -0.128909              -0.180942               0.246076
4              -0.170667              -0.461942               0.330709
5              -0.193102              -0.423990               0.366278
6              -0.131924              -0.355556               0.218771
7              -0.133902              -0.352209               0.221908
8              -0.177334              -0.450663               0.341280
9              -0.200818              -0.410937               0.378511

   Principal Component 4  Principal Component 5  Principal Component 6  \
0              -0.075257              -0.198858              -0.362722
1              -0.088494              -0.180407              -0.331234
2              -0.064220               0.964890              -0.006685
3              -0.064220               0.964890              -0.006685
4               0.918387              -0.090053              -0.085494
5               0.850155               0.005057               0.076810
6              -0.000175               0.012865               0.276604
7              -0.006191               0.021252               0.290917
8               0.898110              -0.061788              -0.037261
9               0.826688               0.037766               0.132628

   cluster
0        1
1        1
2        0
3        0
4        1
5        1
6        1
7        1
8        1
9        1

Kích thước các clusters:
1    2738
0    2524
3    2077
2    1307
Name: cluster, dtype: int64
```

Biểu diễn trực quan sau khi tiến hành phân cụm

```
# Biểu diễn trực quan
plt.figure(figsize = (5, 5))

plt.title('Phân cụm HAC')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.scatter(df_km_hac_pca_vs['Principal Component 1'],
df_km_hac_pca_vs['Principal Component 2'], marker = '.', c =
df_hac['cluster'], cmap='viridis')

plt.show()
```



Hình 27: Biểu đồ trực quan các cụm được gom theo HAC (k=4)

1.3. Đánh giá mô hình phân cụm

Sau khi xây dựng 2 mô hình K-means và HAC, ta tiến hành đánh giá 2 mô hình bằng Silhouette Score để đánh giá hiệu năng của mô hình, chỉ số càng gần 1 thì mô hình đang phân cụm chính xác.

- Silhouette Score cho mô hình K-means:

```
# Tính toán Silhoutte Score để đánh giá hiệu năng của việc phân cụm
score_km = silhouette_score(df_km_hac_pca, model_km.labels_,
metric='euclidean')
# Silhouette score nằm gần 1 nghĩa là ví dụ đang được phân cụm chính
xác, xa các cụm khác
print('Silhouetter Score của KMean: %.3f' % score_km)
```

```
Silhouetter Score của KMean: 0.425
```

- Silhouette Score cho mô hình HAC:

```
# Tính toán Silhoutte Score để đánh giá hiệu năng của việc phân cụm
score_hac = silhouette_score(df_km_hac_pca, model_hac.labels_,
metric='euclidean')
# Silhouette score nằm gần 1 nghĩa là ví dụ đang được phân cụm chính
xác, xa các cụm khác
print('Silhouetter Score của HAC: %.3f' % score_hac)
```

```
Silhouetter Score của HAC: 0.364
```

Như vậy, ta chọn mô hình K-means để phân cụm.

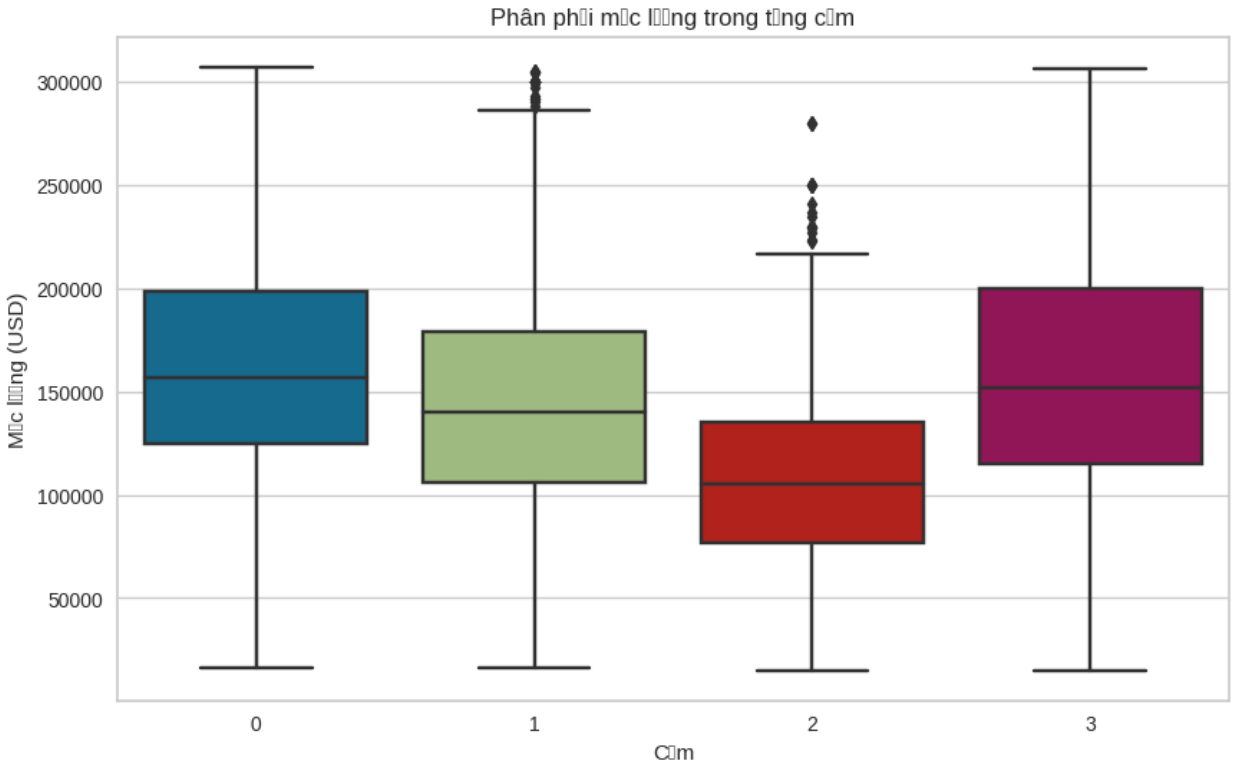
1.4. Phân tích sau khi phân cụm

- **Salary in usd**

```
# Tính các chỉ số liên quan trong phân phối mức lương trong từng cụm
summary_stats =
df_cluster.groupby('cluster')['salary_in_usd'].describe()

# Chuyển đổi DataFrame thành định dạng bảng và hiển thị
print(tabulate(summary_stats, headers='keys', tablefmt='pretty'))
```

```
+---------+-------+--------------------+--------------------+---------+---------+---------+---------+----------+
| cluster | count |        mean        |        std         |   min   |   25%   |   50%   |   75%   |   max    |
+---------+-------+--------------------+--------------------+---------+---------+---------+---------+----------+
|    0    | 2690.0 | 158692.10669144982 | 58413.547541125256 | 16000.0 | 125000.0 | 156400.0 | 198200.0 | 307100.0 |
|    1    | 2077.0 | 144628.85026480502 | 55477.558255040756 | 16228.0 | 105700.0 | 139700.0 | 178600.0 | 305000.0 |
|    2    | 1524.0 | 107423.41929133858 | 42237.75057767051  | 15000.0 | 76750.0  | 105000.0 | 135000.0 | 280000.0 |
|    3    | 2355.0 | 157367.79023354565 | 60739.544990301634 | 15000.0 | 115000.0 | 152000.0 | 200000.0 | 306000.0 |
+---------+-------+--------------------+--------------------+---------+---------+---------+---------+----------+
```

Biểu đồ box plot mức lương của mỗi cụm:



Hình 28: Biểu đồ box plot giữa salary_in_usd và cụm

Dựa trên biểu đồ và thống kê về phân phối của mức lương theo từng cụm, có thể thấy rằng mức lương của những người làm trong ngành khoa học dữ liệu được phân phối theo 4 cụm chính, với các đặc điểm sau:

- Mức lương trung bình: Cụm 3 có mức lương trung bình cao nhất là 163,383, cao hơn đáng kể so với cụm 0 (107,423) và cụm 2 (147,738). Cụm 1 có mức lương cũng khá cao là 158,692.
- Phân bố lương: Cả 4 cụm đều có phân bố lương khá rộng, đều từ khoảng 15,000; 16,000 đến khoảng 280,000 và 300,000. Trong đó, cụm 2 có phân bố tương đối tập trung ổn định ở mức lương tầm trung. Cụm 1 và 3 thì tập trung ở mức lương trung bình đến cao trong khi cụm 0 là cụm phân bố về mức lương thấp hơn so với các cụm còn lại. Riêng cụm 3 cho thấy có sự đang dạng về lương hơn cụm 1 với mức 75th percentile cao nhất và 25th percentile thấp hơn hơn một chút của cụm 1.

Để hiểu rõ hơn về các yếu tố ảnh hưởng đến sự khác biệt về mức lương giữa các cụm, cần tiến hành phân tích sâu sắc hơn về dữ liệu kinh nghiệm làm việc, trình độ nghề nghiệp, vị trí công việc cụ thể, và điều tra thêm về thị trường lao động.

- **Remote Ratio**

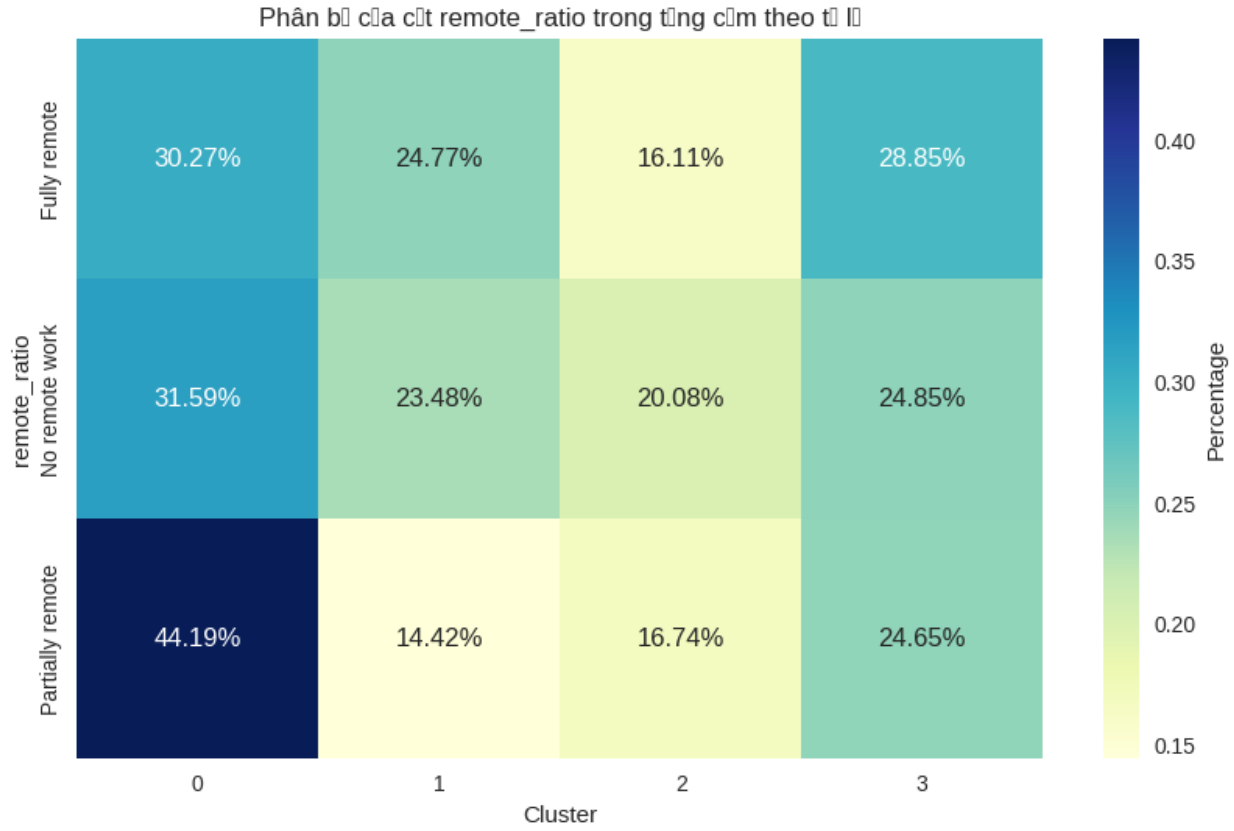
```
remote_ratio_cross_tab = pd.crosstab(df_cluster['remote_ratio'],
df_cluster['cluster'])
print("\nPhân phối của cột 'remote_ratio' trong từng cụm:")
print(remote_ratio_cross_tab)
```

```
Phân phối của cột 'remote_ratio' trong từng cụm:
cluster              0     1    2     3
remote_ratio
Fully remote      1563  1279  832  1490
No remote work    1032   767  656   812
Partially remote    95    31   36    53
```

```
# Tính tỉ lệ theo cụm (tổng theo cụm là 1)
remote_ratio_percentage = remote_ratio_cross_tab.apply(lambda x:
x/x.sum(), axis=1)

# In phân phối tỉ lệ của từng thành phần theo cụm
print("\nPhân phối của cột 'remote_ratio' trong từng cụm theo phần
trăm:")
print(remote_ratio_percentage)
```

```
Phân phối của cột 'remote_ratio' trong từng cụm theo phần trăm:
cluster                 0         1         2         3
remote_ratio
Fully remote      0.302672  0.247676  0.161115  0.288536
No remote work    0.315886  0.234772  0.200796  0.248546
Partially remote  0.441860  0.144186  0.167442  0.246512
```

Phân bố của cột remote_ratio trong từng cụm theo tỷ lệ:



Hình 29: Phân bố của cột remote_ratio trong từng cụm theo tỉ lệ

Thông qua biểu đồ có thể nhận thấy phần lớn kết quả của hình thức hoàn toàn làm việc từ xa (Fully remote) đều được xếp vào cụm 1 và 2 trong đó cụm 2 là nhiều nhất. trong khi cụm 1 mới là cụm được xếp vào nhiều nhất đối với hai hình thức làm việc còn lại là làm việc một phần (Partially remote) và không làm việc từ xa (No remote work). Còn đối với cụm 0 và 3, không cho thấy khác biệt về sự phân bố nào với cụm 2 trong hai hình thức làm việc vừa rồi. Riêng trong loại hình thức làm việc là hoàn toàn làm việc từ xa thì hai cụm này chiếm dữ liệu tương đối thấp hơn hai cụm còn lại.

Như vậy, đối với yếu tố hình thức làm việc thì cụm 0 và cụm 3 chưa thể hiện đặc điểm quá rõ ràng nhưng đối với cụm 1 cho thấy nó có thể đại diện cho đối tượng có hình thức làm việc là không làm việc từ xa bởi sự chênh lệch của cụm này so với 3 cụm còn lại. Và cụm 2 có thể đại diện cho những đối tượng có hình thức làm việc từ xa hoàn toàn.

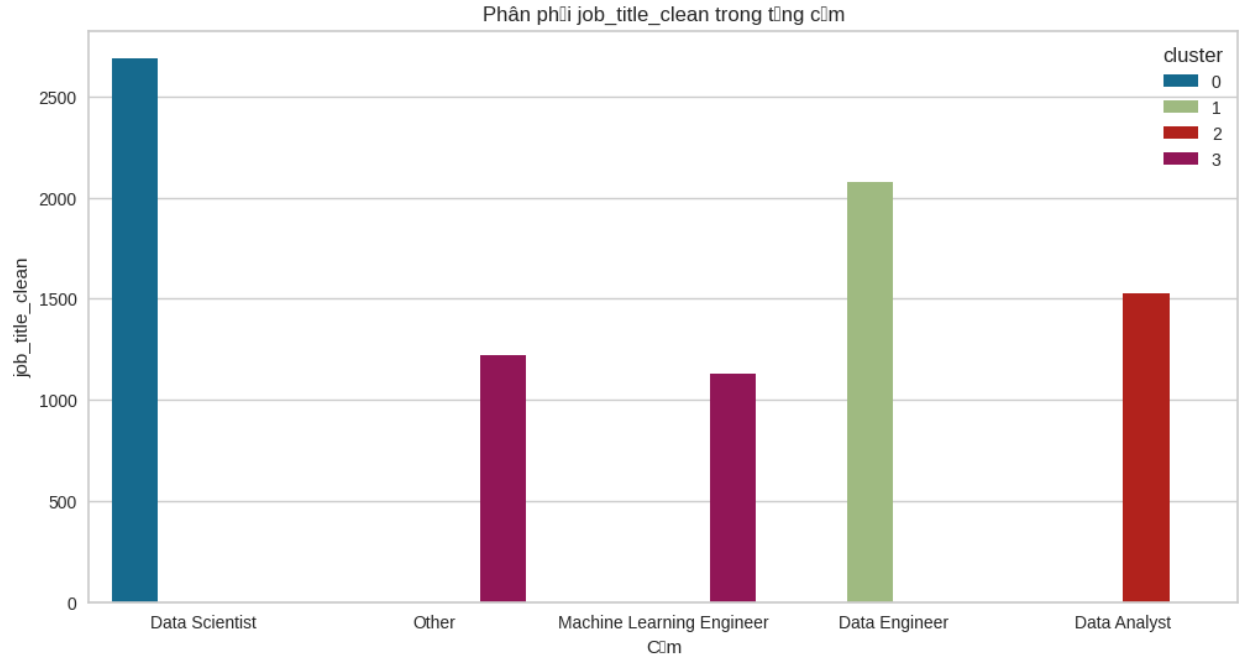
- **Job Title**

```
job_title_clean_cross_tab =
pd.crosstab(df_cluster['job_title_clean'], df_cluster['cluster'])
print("\nPhân phối của cột 'job_title_clean':")
print(job_title_clean_cross_tab)
```

```
Phân phối của cột 'job_title_clean':
cluster                          0     1     2     3
job_title_clean
Data Analyst                     0     0  1524     0
Data Engineer                    0  2077     0     0
Data Scientist                2690     0     0     0
Machine Learning Engineer        0     0     0  1132
Other                            0     0     0  1223
```

Biểu diễn trực quan tương quan giữa job_title_clean với các cụm:

```
fig, axes = plt.subplots( figsize=(12, 6))

sns.countplot(data=df_cluster, x='job_title_clean', hue='cluster')
plt.title('Phân phối job_title_clean trong từng cụm')
plt.xlabel('Cụm')
plt.ylabel('job_title_clean')

plt.show()
```



Hình 30: Phân phối Job_title trong từng cụm

Kết quả biểu đồ đã cho thấy sự khác biệt giữa các cụm đôi với yếu tố nghề nghiệp trong khoa học dữ liệu. Lần lượt, cụm 0 đại diện cho những đối tượng có nhóm nghề là Data Analyst. Cũng phù hợp với phân tích từ trước đó khi cụm 0 là cụm có mức lương trung bình thấp nhất và đã được trực quan trong phần phân tích đa biến. Cụm 1 mang đặc trưng của người làm nhóm Data Scientist. Cụm 2 mang đặc trưng của người làm nhóm Data Engineer và nhóm Other. Và cụm 3 mang đặc trưng của người làm nhóm Machine Learning Engineer, nhóm có mức lương trung bình cao nhất, phù hơi với phân tích của nhóm trong phần tương quan đa biến.

- **Company Size**

```
company_size_cross_tab = pd.crosstab(df_cluster['company_size'],
df_cluster['cluster'])
print("\nPhân phối của cột 'company_size':")
print(company_size_cross_tab)
```

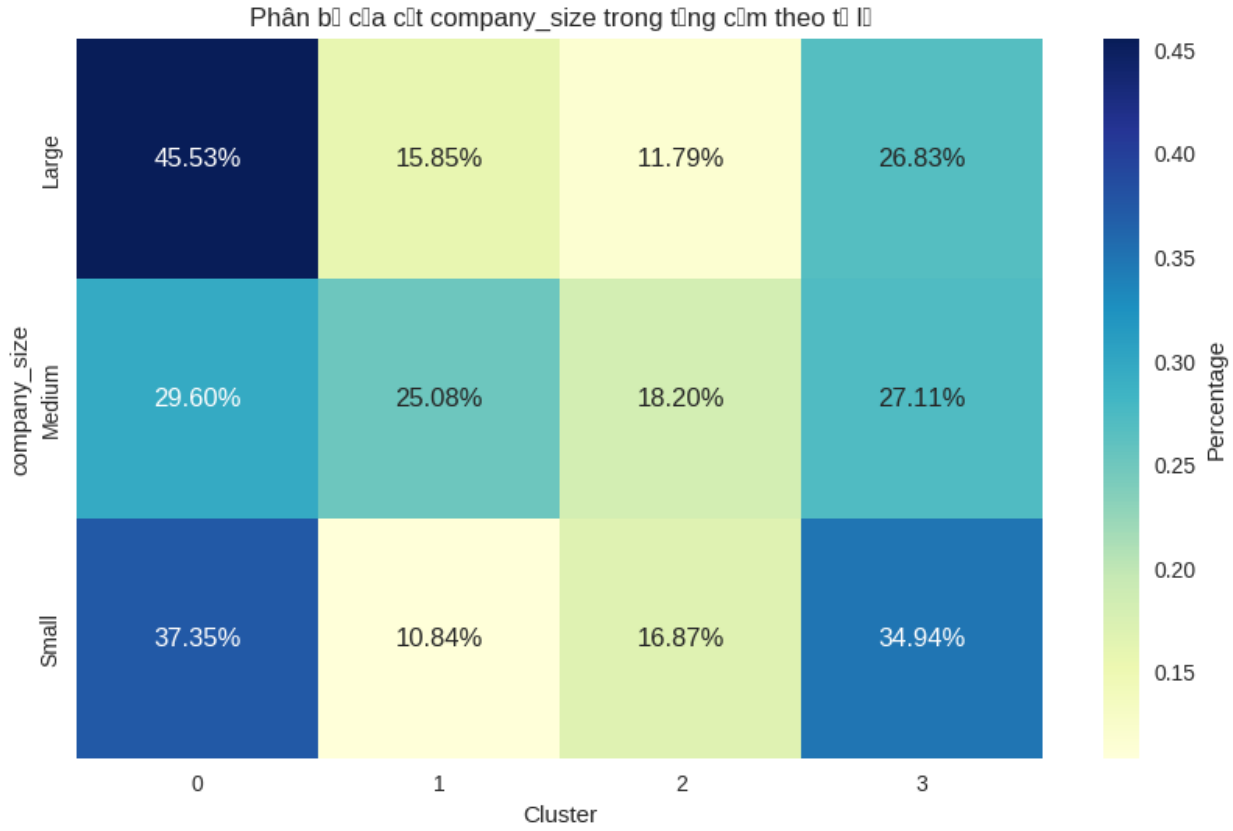
```
Phân phối của cột 'company_size':
cluster          0     1     2     3
company_size
Large          336   117    87   198
Medium        2292  1942  1409  2099
Small           62    18    28    58
```

```
# Tính tỉ lệ theo cụm (tổng theo cụm là 1)
company_size_percentage = company_size_cross_tab.apply(lambda x:
x/x.sum(), axis=1)

# In phân phối tỉ lệ của từng thành phần theo cụm
print("\nPhân phối của cột 'company_size' trong từng cụm theo phần
trăm:")
print(company_size_percentage)
```

```
Phân phối của cột 'company_size' trong từng cụm theo phần trăm:
cluster                0         1         2         3
company_size
Large           0.455285  0.158537  0.117886  0.268293
Medium          0.296048  0.250840  0.181994  0.271119
Small           0.373494  0.108434  0.168675  0.349398
```

Phân bố của cột company_size trong từng cụm theo tỷ lệ:

```
plt.figure(figsize=(10, 6))
sns.heatmap(company_size_percentage, annot=True, fmt=".2%",
cmap="YlGnBu", cbar_kws={'label': 'Percentage'})
plt.title("Phân bố của cột company_size trong từng cụm theo tỷ lệ")
plt.ylabel("company_size")
plt.xlabel("Cluster")
plt.show()
```



Hình 31: Phân bố của cột company_size trong từng cụm theo tỷ lệ

Dựa trên biểu đồ về tương quan giữa quy mô công ty và các cụm, ta nhận thấy được các đặc điểm phân cụm như sau. Đối tượng làm việc tại công ty có quy mô công ty lớn (Large) phân bố chủ yếu ở cụm 1. Còn những người làm việc trong công ty có quy mô vừa thì ba cụm 0, 2, 3 có phân bố đều nhau, riêng cụm 1 thì nhỉnh hơn một chút. Đối với trường hợp công ty có quy mô nhỏ thì cụm 2 có phân bố thấp nhất, cụm 0 và 3 thì tương đối ngang nhau và cụm 1 tập trung phần lớn, khá giống với 2 quy mô trên.

- **Experience Level**

```
experience_level_cross_tab =
pd.crosstab(df_cluster['experience_level'], df_cluster['cluster'])
print("\nPhân phối của cột 'experience_level':")
print(experience_level_cross_tab)
```

```
Phân phối của cột 'experience_level':
cluster              0     1    2     3
experience_level
Entry              139   100  131    97
Executive           73    92   14    69
Mid                457   425  411   430
Senior            2021  1460  968  1759
```

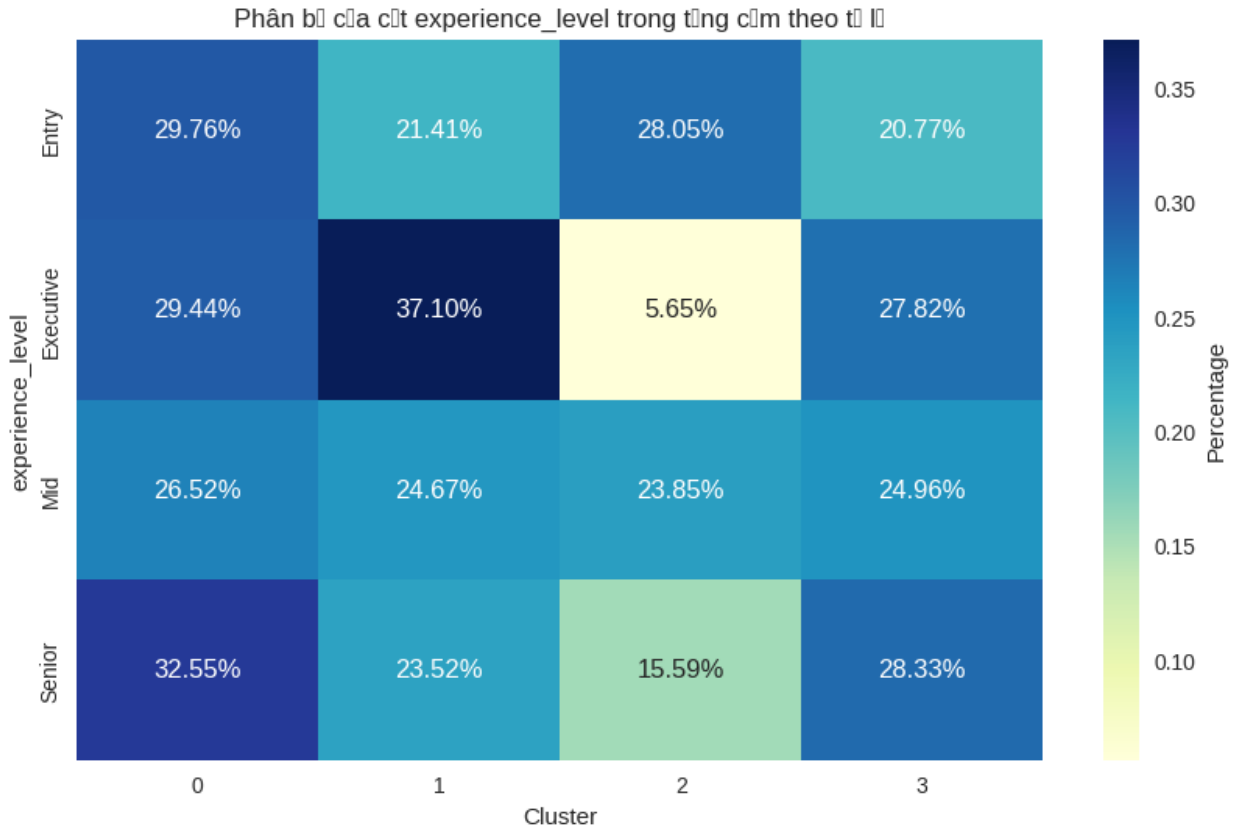
```
# Tính tỉ lệ theo cụm (tổng theo cụm là 1)
experience_level_percentage = experience_level_cross_tab.apply(lambda
x: x/x.sum(), axis=1)
```

```
# In phân phối tỉ lệ của từng thành phần theo cụm
print("\nPhân phối của cột 'experience_level' trong từng cụm theo
phần trăm:")
print(experience_level_percentage)
```

```
Phân phối của cột 'experience_level' trong từng cụm theo phần trăm:
cluster                0         1         2         3
experience_level
Entry             0.297645  0.214133  0.280514  0.207709
Executive         0.294355  0.370968  0.056452  0.278226
Mid               0.265235  0.246663  0.238537  0.249565
Senior            0.325548  0.235180  0.155928  0.283344
```

Biểu diễn trực quan tương quan giữa exeperience_lever với các cụm:

```
plt.figure(figsize=(10, 6))
sns.heatmap(experience_level_percentage, annot=True, fmt=".2%",
cmap="YlGnBu", cbar_kws={'label': 'Percentage'})
plt.title("Phân bố của cột experience_level trong từng cụm theo tỷ lệ
")
plt.ylabel("experience_level")
plt.xlabel("Cluster")
plt.show()
```



Hình 32: Phân bố của cột experience_level trong từng cụm theo tỉ lệ

Kết luận có thể rút ra được từ biểu đồ là với mức kinh nghiệm Entry thì phân bố chủ yếu ở cụm 1 và 3 với cụm 3 tập trung nhiều hơn. Còn với mức kinh nghiệm Executive thì

phần lớn tập trung tại cụm 2 sau đó là cụm 1 và nhỏ nhất là cụm 3. Riêng mức kinh nghiệm Mid thì có phân bố tương đối đồng giữa ba cụm 1, 2, 3 và đều nhỉnh hơn cụm 0 một chút. Cuối cùng với mức kinh nghiệm Senior có phân bố tại cụm 1 cao nhất trong bốn mức kinh nghiệm và nhỏ nhất là cụm 3 nhưng không thấp bằng Executive. Hai cụm còn lại của mức kinh nghiệm này có mức phân bố ngang nhau.

- **Kết luận**

Thông qua phân tích tương quan giữa các cụm và các biến về mức lương, hình thức làm việc, nghề nghiệp, quy mô công ty và mức kinh nghiệm, có thể thấy rằng các cụm mang những đặc điểm như sau:

- **Cụm 0:** Đây là cụm có mức lương thấp nhất, bao gồm chủ yếu các Data Analyst làm việc tại các công ty nhỏ và vừa.
- **Cụm 1:** Đây là cụm có mức lương cao, bao gồm các Data Scientist làm việc tại các công ty lớn và nhỏ; có kinh nghiệm thuộc mức Senior.
- **Cụm 2:** Đây là cụm có mức lương trung bình, bao gồm các Data Engineer và Other; có mức kinh nghiệm Executive.
- **Cụm 3:** Đây là cụm có mức lương cao nhất, bao gồm các Machine Learning Engineer làm việc tại các công ty vừa và nhỏ.

Sự phân cụm này mang lại cái nhìn tổng quan về ngành khoa học dữ liệu, giúp hiểu rõ hơn về các đặc điểm và xu hướng của ngành. Những thông tin này sẽ là cơ sở để đưa ra quyết định chính xác và có ý nghĩa trong quản lý nhân sự và phát triển nguồn nhân lực trong lĩnh vực khoa học dữ liệu.

2. Mô hình dự đoán

2.1. Hàm đánh giá

Ta xây dựng hàm đánh giá thể hiện các chỉ số để có thể đánh giá các mô hình, các chỉ số bao gồm:

- **Mean Squared Error** (MSE - Sai số bình phương trung bình): MSE tính bằng cách lấy trung bình của bình phương chênh lệch giữa giá trị dự đoán và giá trị thực tế. MSE là phổ biến và nhận giá trị lớn nếu có các điểm dữ liệu có sự chênh lệch lớn.

- **Mean Absolute Error** (MAE - Sai số tuyệt đối trung bình): MAE tính bằng cách lấy trung bình của giá trị tuyệt đối của chênh lệch giữa giá trị dự đoán và giá trị thực tế. MAE nhìn chung là ít nhạy cảm hơn với các giá trị ngoại lệ so với MSE.
- **R-squared** (Coefficient of Determination - Hệ số xác định): R-squared đo lường mức độ biến thiên của biến phụ thuộc có thể được giải thích bằng mô hình. Giá trị R-squared nằm trong khoảng từ 0 đến 1, với 1 là hoàn hảo.
- **Median Absolute Error** (Sai số tuyệt đối trung bình độ lệch): Được tính bằng cách lấy trung bình của giá trị tuyệt đối của độ lệch giữa giá trị dự đoán và giá trị thực tế, nhưng sử dụng giá trị trung bình độ lệch thay vì giá trị trung bình.
- **Explained Variance Score**: Được tính bằng cách đo lường mức độ biến thiên của mô hình so với biến thiên tổng cộng của dữ liệu. Giá trị nằm trong khoảng từ 0 đến 1, với 1 là hoàn hảo.
- **Max Error** (Sai số lớn nhất): Là giá trị lớn nhất của chênh lệch giữa giá trị thực tế và giá trị dự đoán.

```
def evaluate_regression_model(y_true, y_pred):
    results = {}

    # Mean Squared Error (MSE)
    mse = mean_squared_error(y_true, y_pred)
    results['MSE'] = mse

    # Mean Absolute Error (MAE)
    mae = mean_absolute_error(y_true, y_pred)
    results['MAE'] = mae

    # R-squared
    r_squared = r2_score(y_true, y_pred)
    results['R-squared'] = r_squared

    # Median Absolute Error
    median_ae = median_absolute_error(y_true, y_pred)
    results['Median Absolute Error'] = median_ae

    # Explained Variance Score
    explained_variance = explained_variance_score(y_true, y_pred)
    results['Explained Variance Score'] = explained_variance

    # Max Error
    max_err = max_error(y_true, y_pred)

    return results
```

2.2. Mô hình Random Forest

Ta quan sát biến target và các biến features

```
## Mô tả dữ liệu
## Biến phân lớp (target variable): 'Class' --> cột cuối cùng trong
file
target = 'salary_in_usd'
print('* Biến target:', target)

## Danh sách các features
nb_features = df_rd.shape[1] - 1
features    = df_rd.drop(target, axis=1).columns
print('* Số lượng features = %2d' %nb_features)
print('  Các features:', ', '.join(features))
```

```
* Biến target: salary_in_usd
* Số lượng features =  6
  Các features: work_year, remote_ratio_clean, usa,
company_size_clean, experience_level_clean, job_title_rd
```

Chia tập dữ liệu thành 3 tập train, tập val và tập test:

```
X = df_rd[features]
y = df_rd[target]
# Tạo tập huấn luyện và tập dữ liệu còn lại
X_train, X_temp, y_train, y_temp = train_test_split(X, y,
test_size=0.3, random_state=42)

# Chia tập dữ liệu còn lại thành tập kiểm tra và tập validation
X_test, X_val, y_test, y_val = train_test_split(X_temp, y_temp,
test_size=0.5, random_state=42)
```

Ta xây dựng mô hình, khởi tạo mô hình Random Forest:

```
# Tạo mô hình Random Forest
rf_model = RandomForestRegressor()
```

Ta sử dụng Grid Search để tìm ra được tham số tốt nhất cho mô hình.

GridSearchCV được khởi tạo, với **estimator** là mô hình Random Forest, **param_grid** là tập hợp các giá trị tham số cần tìm kiếm, **scoring** là độ đo đánh giá (ở đây là negative mean squared error), và **cv** là số lượng folds trong cross-validation (ở đây là 5).

```
# Định nghĩa các tham số cần tìm kiếm
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

```
# Sử dụng GridSearchCV để tìm kiếm các tham số tốt nhất
grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid,
scoring='neg_mean_squared_error', cv=5)
grid_search.fit(X_train, y_train)
```

Sau đó in ra tham số tối ưu nhất được tìm kiếm thông qua Grid Search. Bên cạnh đó ta sử dụng mô hình đã tối ưu hóa để dự đoán trên tập test.

```
# In ra các tham số tốt nhất
print("Best parameters: ", grid_search.best_params_)

# Dự đoán trên tập val
y_pred = grid_search.predict(X_val)

# Đưa ra các chỉ số đánh giá của mô hình
evaluate_regression_model(y_val, y_pred)
```

```
Best parameters:  {'max_depth': 10, 'min_samples_leaf': 2,
'min_samples_split': 10, 'n_estimators': 50}
```

Đánh giá mô hình bằng các chỉ số.

```
# Đưa ra các chỉ số đánh giá của mô hình
evaluate_regression_model(y_val, y_pred)
```

```
{'MSE': 0.026742496450317767,
 'MAE': 0.12788695799479288,
 'R-squared': 0.3383301815091375,
 'Median Absolute Error': 0.10615658245708715,
 'Explained Variance Score': 0.3384064151350954}
```

2.3. Mô hình Linear Regression

Ta quan sát biến target và các biến features

```
## Mô tả dữ liệu
## Biến phân lớp (target variable): 'Class' --> cột cuối cùng trong
file
target = 'salary_in_usd'
print('* Biến target:', target)

## Danh sách các features
nb_features = df_lr_svm_pca.shape[1] - 1
features    = df_lr_svm_pca.drop(target, axis=1).columns
print('* Số lượng features = %2d' %nb_features)
print('  Các features:\n', ',\n'.join(features))
```

```
* Biến target: salary_in_usd
* Số lượng features =  6
  Các features:
 Principal Component 1,
Principal Component 2,
Principal Component 3,
Principal Component 4,
Principal Component 5,
Principal Component 6
```

Chia tập dữ liệu ra thành 3 tập train, tập val và tập test

```
X = df_lr_svm_pca[features]
y = df_lr_svm_pca[target]
# Tạo tập huấn luyện và tập dữ liệu còn lại
X_train, X_temp, y_train, y_temp = train_test_split(X, y,
test_size=0.3, random_state=42)

# Chia tập dữ liệu còn lại thành tập kiểm tra và tập validation
X_test, X_val, y_test, y_val = train_test_split(X_temp, y_temp,
test_size=0.5, random_state=42)
```

Ta xây dựng mô hình:

```
# Tạo mô hình Linear Regression
lr_model = LinearRegression()

# Huấn luyện mô hình
lr_model.fit(X_train, y_train)

# Dự đoán trên tập val
y_pred = lr_model.predict(X_val)
```

Sau khi train mô hình và dự báo dựa trên tập test, ta đưa ra các chỉ số đánh giá cho mô hình Linear Regression

```
# Các chỉ số đánh giá mô hình
evaluate_regression_model(y_val, y_pred)
```

```
{'MSE': 0.026829413269906118,
 'MAE': 0.12951995545164466,
 'R-squared': 0.3361796629011359,
 'Median Absolute Error': 0.1076230993269851,
 'Explained Variance Score': 0.3365634052333927}
```

2.4. Mô hình SVM

SVM và Linear Regression sử dụng chung tập tiền xử lý. Nên ta xây dựng mô hình, khởi tạo mô hình SVM:

```
# Định nghĩa mô hình SVM
```

```
svm_model = SVR()
```

Ta sử dụng Grid Search để tìm ra được tham số tốt nhất cho mô hình. **GridSearchCV** được khởi tạo, với **estimator** là mô hình SVM, **param_grid** là tập hợp các giá trị tham số cần tìm kiếm, **scoring** là độ đo đánh giá (ở đây là negative mean squared error), và **cv** là số lượng folds trong cross-validation (ở đây là 5).

```
# Định nghĩa các tham số cần tìm kiếm
param_grid = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': ['scale', 'auto', 0.1, 1],
    'epsilon': [0.1, 0.2, 0.5]
}

# Sử dụng GridSearchCV để tìm kiếm các tham số tốt nhất
grid_search_svm = GridSearchCV(estimator=svm_model,
param_grid=param_grid, scoring='neg_mean_squared_error', cv=5)
grid_search_svm.fit(X_train, y_train)
```

Sau đó in ra tham số tối ưu nhất được tìm kiếm thông qua Grid Search. Bên cạnh đó ta sử dụng mô hình đã tối ưu hóa để dự đoán trên tập test.

```
# In ra các tham số tốt nhất
print("Best parameters: ", grid_search_svm.best_params_)

# Dự đoán trên tập val
y_pred = grid_search_svm.predict(X_val)
```

```
Best parameters:  {'C': 10, 'epsilon': 0.1, 'gamma': 'auto',
'kernel': 'rbf'}
```

Đánh giá mô hình bằng các chỉ số.

```
# Đánh giá mô hình bằng các chỉ số
evaluate_regression_model(y_val, y_pred)
```

```
{'MSE': 0.026576916307882385,
 'MAE': 0.1274178271118674,
 'R-squared': 0.34242700855721997,
 'Median Absolute Error': 0.10461847022200094,
 'Explained Variance Score': 0.3465907058465425}
```

2.5. Lựa chọn giải pháp

Ta đánh giá hiệu suất của hai mô hình Linear Regression và SVM bằng cách sử dụng phương pháp cross-validation.

Tạo danh sách các mô hình ứng viên

```
## Các mô hình ứng viên
models = [LinearRegression(),
          SVR(C = 10, epsilon = 0.1, gamma = 'auto', kernel = 'rbf')]
```

Ta chọn số folds là 10, tạo scorer sử dụng Mean Squared Error để đánh giá mô hình. Sử dụng **cross_val_score** để thực hiện cross-validation trên mô hình hiện tại.

```
## Mô tả dữ liệu
## Biến phân lớp (target variable): 'Class' --> cột cuối cùng trong
file
target = 'salary_in_usd'
nb_features = df_lr_svm_pca.shape[1] - 1
features    = df_lr_svm_pca.drop(target, axis=1).columns
X = df_lr_svm_pca[features]
y = df_lr_svm_pca[target]
folds = 10
for model in models:
    model_name = model.__class__.__name__
    mse_scorer = make_scorer(mean_squared_error)
    mse = cross_val_score(model, X, y, scoring = mse_scorer, cv =
folds)
    print(f'Mean Squared Error (trung bình) của mô hình {model_name}
= {mse.mean():.4f}')
```

```
Mean Squared Error (trung bình) của mô hình LinearRegression = 0.0273
Mean Squared Error (trung bình) của mô hình SVR = 0.0273
```

Tương tự, ta thực hiện Cross validation cho mô hình Random Forest

```
## Cross validation cho mô hình Random Forest
target = 'salary_in_usd'
nb_features = df_rd.shape[1] - 1
features    = df_rd.drop(target, axis=1).columns
X = df_rd[features]
y = df_rd[target]

model_rd = RandomForestRegressor(max_depth= 20, min_samples_leaf = 2,
min_samples_split = 10, n_estimators = 200)
mse = cross_val_score(model_rd, X, y, scoring = mse_scorer, cv =
folds)

print(f'Mean Squared Error (trung bình) của mô hình Random Forest =
{mse.mean():.4f}')
```

```
Mean Squared Error (trung bình) của mô hình Random Forest = 0.0272
```

Dựa vào các kết quả trên, ta thấy rằng mô hình Random Forest là mô hình có chỉ số MSE trung bình thấp nhất, tuy nhiên chỉ số MSE trung bình của mô hình SVM và Linear

Regression chỉ chênh lệch 0.0001 nên ta xét nên các chỉ số khác. Nhìn lại kết quả đánh giá của 3 mô hình ở trên:

Chỉ số đánh giá	Linear Regression	SVM	Random Forest
Mean Squared Error	0.0268	0.0266	0.0267
Mean Absolute Error	0.1295	0.1274	0.1279
R-squared	0.3362	0.3424	0.3383
Median Absolute Error	0.1076	0.1046	0.1062
Explained Variance Score	0.3366	0.3466	0.3384

Nhận xét:

- **Mean Squared Error (MSE)**: Tất cả ba mô hình đều có MSE khá gần nhau, nhưng SVM có MSE thấp nhất (0.0266). Điều này có thể là một lợi thế.
- **Mean Absolute Error (MAE)**: SVM có MAE thấp nhất (0.1274), có thể cho thấy nó có khả năng dự đoán tốt trung bình.
- **R-squared (Coefficient of Determination)**: SVM có R-squared cao nhất (0.3424), làm cho nó có khả năng giải thích phần lớn biến thiên trong biến phụ thuộc.
- **Median Absolute Error**: SVM có Median Absolute Error thấp nhất, có thể chỉ ra rằng nó đạt được hiệu suất ổn định hơn đối với các giá trị ngoại lệ.
- **Explained Variance Score**: SVM cũng có Explained Variance Score cao nhất (0.3466), đo lường khả năng giải thích của mô hình.

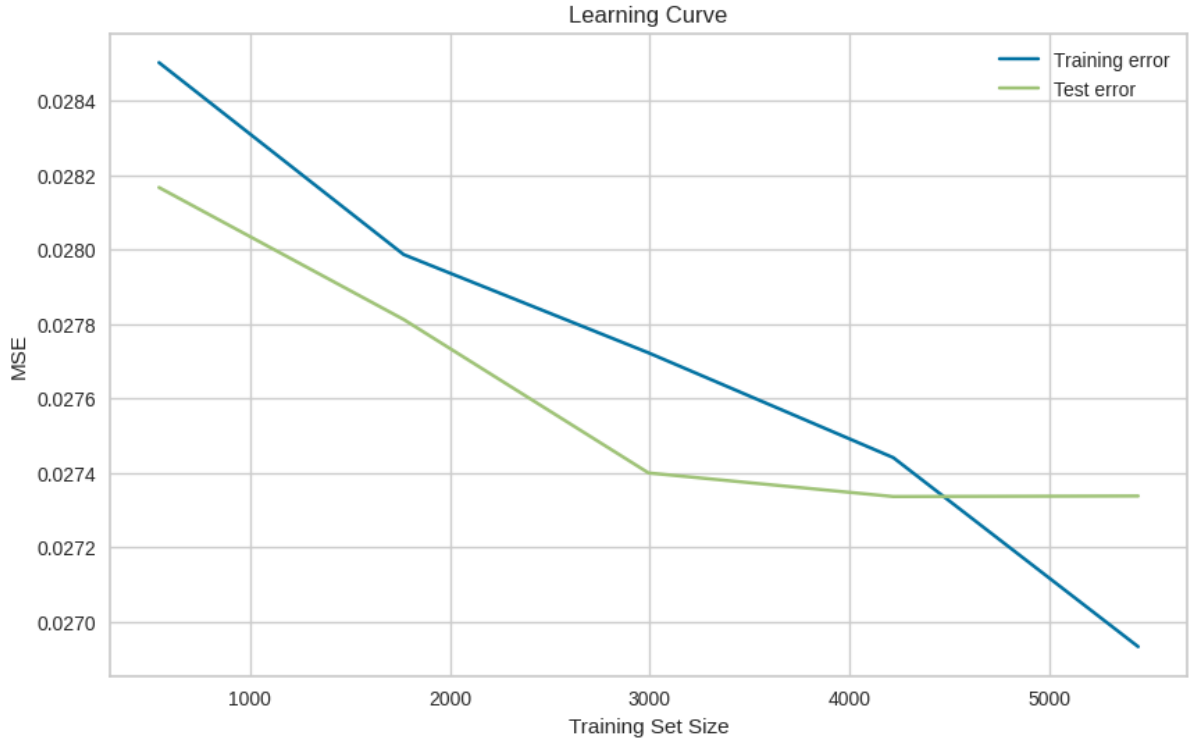
Dựa trên các kết quả này, có thể nói rằng SVM có vẻ là mô hình tốt nhất trong 3 mô hình SVM, Linear Regression và Random Forest dựa trên các chỉ số được đánh giá.

2.6. Đánh giá mô hình lựa chọn

Kiểm tra model fit của mô hình SVM, ta dùng các chỉ số như bias và variance.

```
# Tạo mô hình SVM
svm_model = SVR(C = 10, epsilon = 0.1, gamma = 'auto', kernel =
'rbf')

# Hàm tính bias, variance, và MSE
def bias_variance_analysis(model, X, y):
    # Tính toán các giá trị dự đoán trên tập huấn luyện
    y_pred = cross_val_predict(model, X, y, cv=10)
```

```
    # Tính MSE
    mse = mean_squared_error(y, y_pred)

    # Tính bias
    bias = np.mean((y - np.mean(y_pred))**2)

    # Tính variance
    variance = np.var(y_pred)

    return bias, variance, mse

# Đánh giá bias, variance, và MSE
bias, variance, mse = bias_variance_analysis(svm_model, X_train,
y_train)

print(f'Bias: {bias}')
print(f'Variance: {variance}')
print(f'MSE: {mse}')

# Vẽ đồ thị learning curve để quan sát trade-off giữa bias và
variance
train_sizes, train_scores, test_scores = learning_curve(svm_model,
X_train, y_train, cv=10, scoring='neg_mean_squared_error')

train_scores_mean = -np.mean(train_scores, axis=1)
test_scores_mean = -np.mean(test_scores, axis=1)

plt.figure(figsize=(10, 6))
plt.plot(train_sizes, train_scores_mean, label='Training error')
plt.plot(train_sizes, test_scores_mean, label='Cross - validation
error')
plt.xlabel('Training Set Size')
plt.ylabel('MSE')
plt.title('Learning Curve')
plt.legend()
plt.show()
```



Hình 33: Learning Curve

Bias: nghĩa là độ lệch, biểu thị sự chênh lệch giữa giá trị trung bình mà mô hình dự đoán và giá trị thực tế của dữ liệu.

Variance: nghĩa là phương sai, biểu thị độ phân tán của các giá trị mà mô hình dự đoán so với giá trị thực tế.

- Bias thấp đồng nghĩa với mô hình có khả năng giữ nguyên đặc điểm của dữ liệu huấn luyện, đối với dữ liệu mới. Giá trị này khá thấp, cho thấy mô hình có khả năng diễn giải tốt đặc điểm của dữ liệu huấn luyện.
- Variance thấp là một điều tích cực, vì nó cho thấy mô hình ít nhạy cảm với biến động trong dữ liệu. Sự ổn định này giúp đảm bảo rằng mô hình có khả năng tổng quát hóa tốt trên dữ liệu mới.

Vì độ chệch thấp, phương sai thấp (Low bias, Low Variance) nên đây là trường hợp mô hình khớp tốt (good fitting) vì phân phối của giá trị dự báo trùng với phân phối của ground truth.

Đánh giá mô hình trên tập test:

```
y_pred = grid_search_svm.predict(X_test)
# Đánh giá mô hình bằng các chỉ số
evaluate_regression_model(y_test, y_pred)
```

```
{'MSE': 0.024780993523897003,
 'MAE': 0.12430273672372051,
 'R-squared': 0.31843558526319315,
 'Median Absolute Error': 0.10382795420597607,
 'Explained Variance Score': 0.31843776371527277}
```

Đánh giá mô hình SVM với các chỉ số như trên:

- MSE khá thấp (0.0248), điều này thường được coi là tích cực vì nó cho thấy mức độ chính xác tốt trên tập kiểm tra.
- MAE cũng thấp (0.1243), điều này chỉ ra rằng mô hình có khả năng dự đoán tốt trung bình.
- R-squared là 0.3184, nghĩa là mô hình giải thích được khoảng 31.84% biến thiên.
- Median Absolute Error thấp (0.1038), điều này chỉ ra rằng mô hình có hiệu suất ổn định hơn đối với các giá trị ngoại lệ.
- Explained Variance Score cũng khá cao (0.3184), điều này chỉ ra mức độ giải thích tốt của mô hình.

Dựa vào các chỉ số này, mô hình SVM có vẻ có hiệu suất tốt trong việc dự đoán trên tập kiểm tra.

KẾT LUẬN

Với học phần "Lập trình phân tích dữ liệu", nhóm tiếp tục sử dụng bộ dữ liệu Data Science Job Salaries, bên cạnh biểu diễn trực quan các dữ liệu các biến và phát triển đồ án theo hướng trực quan hơn và dễ dàng tiếp cận sử dụng thông tin hơn. Nhóm đã xây dựng được các mô hình phân cụm, mô hình hồi quy với kết quả khả quan.

Trong đề tài, nhóm đã áp dụng được nguyên tắc thiết kế biểu đồ và sử dụng các biểu đồ đúng với mục đích quan sát dữ liệu mà nhóm đề ra. Khắc phục những hạn chế của đồ án trước, trong môn lập trình phân tích dữ liệu, nhóm đã thực hiện kiểm định ANOVA, áp dụng các kiến thức được học về PCA để giảm chiều dữ liệu. Bên cạnh đó, ngoài mô hình hồi quy có giám sát được nhóm nêu ra trong mục tiêu đề tài để dự đoán mức lương ngành Khoa học dữ liệu, nhóm còn xây dựng thêm một số mô hình gom cụm để có thể hiểu hơn về các nhóm dữ liệu, thực hiện đánh giá các mô hình gom cụm để tìm ra được số cụm tối ưu nhất.

Tuy nhiên, đồ án của nhóm vẫn còn nhiều điểm hạn chế cả về chủ quan lẫn khách quan. Mô hình hồi quy của nhóm xây dựng chỉ có chỉ số R-squares là 31.84%, vẫn còn chưa thực sự tối ưu. Một phần vì nhóm chưa xử lý dữ liệu tốt, một phần vì bộ dữ liệu của nhóm phần lớn là các biến phân loại (ngoại trừ biến salary_in_usd) nên các biến trong mô hình hồi quy dùng nhiều biến giả dẫn đến kết quả mô hình không chính xác cao. Trong bộ dữ liệu thu thập vẫn còn nhiều thiếu sót như vị trí và thời gian thu thập số liệu chênh lệch nhiều dẫn đến không đồng đều giữa các giá trị trong tập dữ liệu.

Vì vậy, trong tương lai nhóm sẽ tiếp tục cải thiện các chỉ số trong mô hình hồi quy của mình và bộ dữ liệu trong tương lai sẽ được thu thập ngày càng đầy đủ hơn. Điều này sẽ giúp mô hình của nhóm có được độ chính xác cao hơn trong tương lai.

TÀI LIỆU THAM KHẢO

Bài 19: Support Vector Machine — Machine Learning cơ bản. Retrieved April 9, 2017, from https://machinelearningcoban.com/2017/04/09/smv/

Bài giảng học phần Khai phá dữ liệu, "Chương 4: Luật kết hợp", TS.Nguyễn An Tế, khoa Công nghệ thông tin kinh doanh, trường Công nghệ và Thiết kế UEH. (2022).

chungtoi. (2023, March 10). 9 Chỉ Tiêu đánh Giá độ Chính Xác Mô Hình Hồi Quy - Chạy định Lượng. Chạy định lượng. https://chaydinhluong.com/9-chi-tieu-danh-gia-do-chinh-xac-mo-hinh-hoi-quy/?fbclid=IwAR1wjsI1XnfwHEFDrQqNmxH4yHK35_bScS8qxmTGVMKw07HhJv3x9L9TZS8

Hải, H. P., Hòa, N. V., Nghị, Đ. T., Kỹ Thuật Công Nghệ Môi, K., Đại, T., & Giang, A. (n.d.). SO SÁNH MÔ HÌNH HỌC SÂU VỚI CÁC PHƯƠNG PHÁP HỌC TỰ ĐỘNG KHÁC TRONG PHÂN LỚP DỮ LIỆU BIỂU HIỆN GEN MICROARRAY. Edu.Vn. Retrieved November 26, 2023, from https://www.cit.ctu.edu.vn/~dtnghi/rech/dir_0/gen.pdf

What is random forest? (n.d.). Ibm.com. Retrieved November 26, 2023, from https://www.ibm.com/topics/random-forest?fbclid=IwAR1umQJR4-jHq7bWpJUeYvyGJSozm3XS2mX7V6arLFtpRCNK5tdawVI-5Hc

Wilke, C. O. (2019). Fundamentals of data visualization: A primer on making informative and compelling figures. O'Reilly Media.

(N.d.). Amazon.com. Retrieved November 26, 2023, from https://aws.amazon.com/vi/what-is/linear-regression/?fbclid=IwAR3_RzkItJYYL1U2keUI3F4KddzXjoYtWchvKilz9-Bkq5DFTcWWfa7ZHig

RPubs - PCA. (n.d.-b). Retrieved from https://rpubs.com/ngocnv/851193

PHỤ LỤC

STT	Họ và tên	MSSV	Nội dung	Đánh giá
1	Lê Trần Khánh Phú	31211024087	Tiền xử lý dữ liệu Giảm chiều dữ liệu (PCA) Xây dựng mô hình gom cụm Tổng hợp báo cáo word Chuẩn bị slide thuyết trình	100%
2	Phạm Minh Phước	31211027663	Tổng quan bộ dữ liệu Kiểm tra tình trạng bộ dữ liệu Định nghĩa, mục đích của mô hình hồi quy Tổng hợp báo cáo word Chuẩn bị slide thuyết trình	100%
3	Đỗ Quang Thiên Phú	31211024191	Tổng quan bộ dữ liệu Phân tích bộ dữ liệu nghiên cứu Đánh giá mô hình máy học Tổng hợp báo cáo word Chuẩn bị slide thuyết trình	100%
4	Trương Thanh Phong	31211027662	Xây dựng mô hình gom cụm Phân tích sau khi gom cụm Tổng hợp báo cáo word	100%
5	Nguyễn Tân Niên	31211027661	Xây dựng mô hình và đánh giá Tổng hợp báo cáo word Chuẩn bị slide thuyết trình	100%