PHẦN I. TỔNG QUAN NGHIÊN CỨU (OVERVIEW)

Ngày nay, nhiều thành phố lớn đã triển khai dịch vụ cho thuê xe đạp để nâng cao sự thoải mái khi đi lại. Điều quan trọng là cung cấp dịch vụ cho thuê xe đạp cho công chúng vào đúng thời điểm vì điều này giúp giảm thời gian chờ đợi. Cuối cùng, việc cung cấp nguồn cung cấp dịch vụ cho thuê xe đạp ổn định cho các thành phố trở thành một vấn đề quan trọng. Phần quan trọng là dự đoán số lượng xe đạp cần thiết mỗi giờ để đảm bảo nguồn cung cấp xe đạp cho thuê ổn định.

Dữ liệu được cung cấp trong **SeoulBikeData.csv** chứa thông tin về số lượng xe đạp đã được thuê và trả lại tại trạm ở Hàn Quốc. Dữ liệu bao gồm các số liệu hàng ngày về số lượng xe đạp đã được thuê và trả lại trong một khoảng thời gian 12 tháng, từ tháng 12 năm 2017 đến tháng 11 năm 2018.

Dữ liệu có các cột sau:

- **Date**: Ngày của giao dịch thuê/trả xe đạp.
- **Rented Bike Count**: Tổng số lượng xe đạp đã được thuê.
- **Return Bike Count**: Tổng số lượng xe đạp đã được trả.
- **Hour**: Giờ của giao dịch thuê/trả xe đạp.
- **Temperature (C)**: Nhiệt độ trong đơn vị Celsius.
- **Humidity (%)**: Độ ẩm theo phần trăm.
- **Wind speed (m/s)**: Tốc độ gió trong mẫu giây trên mét vuông.
- **Visibility (10m)**: Tầm nhìn trong khoảng 10 mét.
- **Dew point temperature (C)**: Nhiệt độ điểm sương trong đơn vị Celsius.
- **Solar Radiation (MJ/m2)**: Độ phóng xạ theo m2
- **Rainfall (mm)**: Số lượng mưa trong đơn vị milimét.
- **Snowfall (cm)**: Số lượng tuyết trong đơn vị centimét.
- **Seasons**: Mùa trong năm (ví dụ như Xuân, Hạ, Thu, Đông).
- **Holiday**: Xem ngày đó có phải là ngày lễ không (có hoặc không).
- **Functioning Day**: Xem ngày đó có phải là ngày làm việc không (có hoặc không).

Bộ dữ liệu này có thể được sử dụng để phân tích và lập mô hình nhu cầu thuê xe đạp ở Seoul, có tính đến các yếu tố thời tiết và mùa khác nhau.

PHẦN II. CƠ SỞ LÝ LUẬN (LITERATURE REVIEW)

2.1. Giới thiệu

Hiện nay, ở Seoul đang phát triển dịch vụ cho thuê xe đạp, gọi là "Ddareungi", được triển khai lần đầu tiên vào năm 2015 và đã thu hút được sự quan tâm nhờ vào hiệu quả giúp tăng cường sử dụng xe đạp, cải thiện mối liên kết đầu cuối với các phương thức vận tải khác và giảm thiểu tác động tiêu cực của hoạt

động vận tải đến môi trường. Theo nghiên cứu "The Meddin Bike-sharing World Map" được thực hiện bởi PBSC Urban Solutions, tính đến tháng 12 năm 2023, tổng số lượt sử dụng dịch vụ Ddareungi đã vượt mốc 100 triệu lượt và xu hướng này sẽ tiếp tục tăng trong tương lai.

Với tần suất giao nhận xe hàng ngày rất lớn, sự mất cân bằng về số lượng xe đạp ở các trạm dẫn đến suy giảm đáng kể hiệu quả cung cấp dịch vụ của hệ thống và số lượng người dùng tiềm năng. Do đó, việc dự đoán nhu cầu, xác định số lượng xe đạp sẽ được sử dụng trong tương lai là chìa khóa cho các hoạt động tái cân bằng.

2.2 Tổng quan tình hình nghiên cứu

Các nhà nghiên cứu dựa trên các mục tiêu khác nhau bằng cách sử dụng nhiều phương pháp để kiểm tra nhu cầu của người sử dụng xe đạp *(Fishman, 2015)*, chẳng hạn như "xác định mối quan hệ giữa thời tiết và hoạt động của người dùng", "kiểm tra tác động của địa hình đến hoạt động của nhà ga". Có rất nhiều nghiên cứu được thực hiện để dự đoán nhu cầu xe đạp toàn cầu nhưng nghiên cứu chuyên sâu tập trung vào các trạm nói riêng thì chưa có nhiều *(Wu. X., và cộng sự, 2019)*. Các nghiên cứu đó đã sử dụng mô hình hồi quy Random Forest Regression, Gradient Boosting Regression and Artificial Neural Network. Theo nghiên cứu *"Mobility Modeling and Prediction in Bike-Sharing Systems"*, lưu lượng trên mỗi trạm với mức độ chi tiết theo giờ được thể hiện chính xác hơn bằng phương pháp huấn luyện model (Train model).

Ngoài ra, nhiều nhà nghiên cứu đã cố gắng xác định các yếu tố có thể làm tăng nhu cầu về thuê xe đạp công cộng. Thời tiết được coi là một yếu tố quan trọng ảnh hưởng đến nhu cầu thuê xe đạp và một nghiên cứu gần đây, sử dụng dữ liệu từ *"40 Chương trình chia sẻ xe đạp công cộng ở 5 vùng khí hậu"* kết luận rằng biến số quan trọng nhất là thời gian trong ngày, tiếp theo là lượng mưa *(Richard, 2021)*. Nghiên cứu khác *(Saneinejad và cộng sự, 2012)* tập trung vào các yếu tố như gió, độ ẩm và nhiệt độ khi đạp xe. Trong đó, ngoài nhiệt độ, còn lại đều có mối tương quan nghịch biến với nhu cầu đi xe đạp. Một nghiên cứu tương tự cũng nhấn mạnh rằng lượng hành khách nếu nhiệt độ thấp và độ ẩm cao *(Gebhart và Noland, 2013)*. Một dự báo ngắn hạn về việc sử dụng bến tàu ở Tô Châu - Trung Quốc *(Xu, X., và cộng sự, 2019)* đã đưa ra được những kết luận như số lượng xe đạp được thuê vào ngày mưa ít hơn ngày không mưa. Những kết quả này cho thấy ý nghĩa về mặt hành vi của khách hàng.

Bên cạnh thời tiết, thời gian cũng là một yếu tố quan trọng ảnh hưởng đến số lượng thuê xe đạp. Một nghiên cứu về CityCycle của Brisbane cho thấy các ngày nghỉ lễ, cuối tuần sẽ có tác động tích cực và đáng kể đến việc sử dụng xe đạp công cộng. Xét về sự khác biệt về thời gian, việc sử dụng xe đạp công cộng khác nhau giữa các mùa, với mức độ sử dụng vào mùa hè cao hơn so với mùa đông *(Eren và Uz, 2020)*. Như vậy, có thể giả định việc thuê xe vào các ngày cuối tuần hay nghỉ lễ, đồng thời lượng thuê xe đạp vào mùa hè sẽ nhiều hơn.

2.3. Phương pháp luận

Bài nghiên cứu xây dựng mô hình dự đoán nhu cầu thuê xe đạp theo giờ. Dữ liệu được sử dụng bao gồm thời điểm, số lượng xe đạp được thuê mỗi giờ và thông tin thời tiết (Temperature, Humidity, Windspeed, Visibility, Dewpoint, Solar radiation, Snowfall, Rainfall).

Nhóm nghiên cứu đã sử dụng phương pháp Hồi quy tuyến tính (Linear regression), mô hình Hồi quy rừng ngẫu nhiên (Random Forest Regression), mô hình Gradient Boosting và mô hình XGBoost.

PHẦN III. ĐÁNH GIÁ TỔNG QUÁT DỮ LIỆU (OVERALL DATA EVALUATION)

Xuất thư viện (Importing libraries)

In [5]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import warnings
```

Xuất dữ liệu (Importing data)

In [6]:
```
data = pd.read_csv('E:/Desktop/Self-development/python/Project/Seoul Bike Sharing Demand
```

3.1. Mô tả dữ liệu (Data desciption)

In [7]:
```
# Data Shape
data.shape
```

Out[7]:
```
(8760, 14)
```

> *Nhận xét*
>
> Data gồm 14 chỉ số (trong đó có 1 cột là label, 13 chỉ số dùng để dự đoán label), với tổng 8760 samples.

In [8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Data columns (total 14 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Date                       8760 non-null   object 
 1   Rented Bike Count          8760 non-null   int64  
 2   Hour                       8760 non-null   int64  
 3   Temperature(°C)            8760 non-null   float64
 4   Humidity(%)                8760 non-null   int64  
 5   Wind speed (m/s)           8760 non-null   float64
 6   Visibility (10m)           8760 non-null   int64  
 7   Dew point temperature(°C)  8760 non-null   float64
 8   Solar Radiation (MJ/m2)    8760 non-null   float64
 9   Rainfall(mm)               8760 non-null   float64
 10  Snowfall (cm)              8760 non-null   float64
 11  Seasons                    8760 non-null   object 
 12  Holiday                    8760 non-null   object 
 13  Functioning Day            8760 non-null   object 
dtypes: float64(6), int64(4), object(4)
memory usage: 958.3+ KB
```

Nhận xét

- 4 cột dạng int64
- 6 cột dạng float64
- 4 cột dạng object

In [9]:
```
# Hình dáng dữ liệu theo bảng (Dataset View)
data.head(5)
```

Out[9]:

	Date	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfal
0	01/12/2017	254	0	-5.2	37	2.2	2000	-17.6	0.0	
1	01/12/2017	204	1	-5.5	38	0.8	2000	-17.6	0.0	
2	01/12/2017	173	2	-6.0	39	1.0	2000	-17.7	0.0	
3	01/12/2017	107	3	-6.2	40	0.9	2000	-17.6	0.0	
4	01/12/2017	78	4	-6.0	36	2.3	2000	-18.6	0.0	

In [10]:
```
# Danh sách các cột trong Dataset (Data Columns)
data.columns
```

Out[10]:
```
Index(['Date', 'Rented Bike Count', 'Hour', 'Temperature(°C)', 'Humidity(%)',
       'Wind speed (m/s)', 'Visibility (10m)', 'Dew point temperature(°C)',
       'Solar Radiation (MJ/m2)', 'Rainfall(mm)', 'Snowfall (cm)', 'Seasons',
       'Holiday', 'Functioning Day'],
      dtype='object')
```

In [11]:
```
# Kiểm tra trùng lặp
data.duplicated().sum()
```

Out[11]:
```
0
```

Nhận xét

Không có hai rows bất kì nào trùng nhau trong bảng.

In [12]:
```
# Đếm các giá trị trùng lặp, giá trị Null, giá trị NaN và giá trị duy nhất (Duplicated val

def data_stat(data) :
    stat_df = pd.DataFrame(index= data.columns)
    stat_df["DataType"] = data.dtypes
    stat_df["Not Null Values"] = data.count()
    stat_df["Duplicated Values"] = data.duplicated().sum()
    stat_df["Null Values"] = data.isnull().sum()
    stat_df["NaN Values"] = data.isna().sum()
    stat_df["Unique Values"] = data.nunique()
    return stat_df

data_stat(data).style.background_gradient()
```

Out[12]:

	DataType	Not Null Values	Duplicated Values	Null Values	NaN Values	Unique Values
Date	object	8760	0	0	0	365
Rented Bike Count	int64	8760	0	0	0	2166
Hour	int64	8760	0	0	0	24
Temperature(°C)	float64	8760	0	0	0	546
Humidity(%)	int64	8760	0	0	0	90
Wind speed (m/s)	float64	8760	0	0	0	65
Visibility (10m)	int64	8760	0	0	0	1789
Dew point temperature(°C)	float64	8760	0	0	0	556
Solar Radiation (MJ/m2)	float64	8760	0	0	0	345
Rainfall(mm)	float64	8760	0	0	0	61
Snowfall (cm)	float64	8760	0	0	0	51
Seasons	object	8760	0	0	0	4
Holiday	object	8760	0	0	0	2
Functioning Day	object	8760	0	0	0	2

Nhận xét

- Các cột không có dữ liệu trùng lặp cũng như không có dữ liệu Null, NaN.
- Các chỉ số Sessions, Holiday, Functioning Day có số giá trị unique values đặc biệt thấp.

In [13]:

```
print("General Statistics")
def data_stat(data) :
    stat_df = pd.DataFrame(index= data.columns)
    stat_df["DataType"] = data.dtypes
    stat_df["Not Null Values"] = data.count()
    stat_df["NaN Values"] = data.isnull().sum()
    stat_df["Unique Values"] = data.nunique()
    return stat_df

data_stat(data)
```

```
General Statistics
```

Out[13]:

	DataType	Not Null Values	NaN Values	Unique Values
Date	object	8760	0	365
Rented Bike Count	int64	8760	0	2166
Hour	int64	8760	0	24
Temperature(°C)	float64	8760	0	546
Humidity(%)	int64	8760	0	90
Wind speed (m/s)	float64	8760	0	65
Visibility (10m)	int64	8760	0	1789
Dew point temperature(°C)	float64	8760	0	556
Solar Radiation (MJ/m2)	float64	8760	0	345
Rainfall(mm)	float64	8760	0	61
Snowfall (cm)	float64	8760	0	51
Seasons	object	8760	0	4
Holiday	object	8760	0	2
Functioning Day	object	8760	0	2

Nhận xét

Các chỉ số Sessions. Holiday, Functioning Day có số giá trị unique values đặc biệt thấp

In [14]:
```
# Thống kê cơ bản với từng chỉ số (General statistics)
data.describe().T
```

Out[14]:

	count	mean	std	min	25%	50%	75%	max
Rented Bike Count	8760.0	704.602055	644.997468	0.0	191.00	504.50	1065.25	3556.00
Hour	8760.0	11.500000	6.922582	0.0	5.75	11.50	17.25	23.00
Temperature(°C)	8760.0	12.882922	11.944825	-17.8	3.50	13.70	22.50	39.40
Humidity(%)	8760.0	58.226256	20.362413	0.0	42.00	57.00	74.00	98.00
Wind speed (m/s)	8760.0	1.724909	1.036300	0.0	0.90	1.50	2.30	7.40
Visibility (10m)	8760.0	1436.825799	608.298712	27.0	940.00	1698.00	2000.00	2000.00
Dew point temperature(°C)	8760.0	4.073813	13.060369	-30.6	-4.70	5.10	14.80	27.20
Solar Radiation (MJ/m2)	8760.0	0.569111	0.868746	0.0	0.00	0.01	0.93	3.52
Rainfall(mm)	8760.0	0.148687	1.128193	0.0	0.00	0.00	0.00	35.00
Snowfall (cm)	8760.0	0.075068	0.436746	0.0	0.00	0.00	0.00	8.80

In [15]:
```
# Thống kê cơ bản với từng chỉ số (General statistics)
data.describe().T.style.background_gradient().format(precision = 2)
```

Out[15]:

	count	mean	std	min	25%	50%	75%	max
Rented Bike Count	8760.00	704.60	645.00	0.00	191.00	504.50	1065.25	3556.00
Hour	8760.00	11.50	6.92	0.00	5.75	11.50	17.25	23.00
Temperature(°C)	8760.00	12.88	11.94	-17.80	3.50	13.70	22.50	39.40
Humidity(%)	8760.00	58.23	20.36	0.00	42.00	57.00	74.00	98.00
Wind speed (m/s)	8760.00	1.72	1.04	0.00	0.90	1.50	2.30	7.40
Visibility (10m)	8760.00	1436.83	608.30	27.00	940.00	1698.00	2000.00	2000.00
Dew point temperature(°C)	8760.00	4.07	13.06	-30.60	-4.70	5.10	14.80	27.20
Solar Radiation (MJ/m2)	8760.00	0.57	0.87	0.00	0.00	0.01	0.93	3.52
Rainfall(mm)	8760.00	0.15	1.13	0.00	0.00	0.00	0.00	35.00
Snowfall (cm)	8760.00	0.08	0.44	0.00	0.00	0.00	0.00	8.80

Nhận xét

Khoảng giá trị và variance của các chỉ số numeric là khá khác nhau, có thể rất lớn như Rent Bike Count hoặc Visibility, rất nhỏ như Snowfall, Rainfall.

Do đó, ta cần scale lại data trước khi đưa vào mô hình để đảm bảo không có chỉ số nào dominate phần còn lại.

Nhận xét chung

- Có tất cả 14 chỉ số trong bảng với 8760 mẫu, bao gồm tất cả đều đủ giá trị không có missing values.
- Chủ yếu đều là giá trị số (float hoặc int), có 4 cột nhận giá trị là text đó là Date, Sessions, Holiday, Functioning Day.
- Tên của một số chỉ số còn bao gồm thêm cả đơn vị đo, gây khó khăn cho quá trình làm bài toán, sẽ được đổi tên cho đơn giản hơn.
- Các chỉ số Sessions. Holiday, Functioning Day có số giá trị unique values đặc biệt thấp.
- Cần xử lí hiệu quả các chỉ số ở dạng text để đưa về dạng numeric.
- Khoảng giá trị của các chỉ số numeric là khá khác nhau, có thể rất lớn như Rent Bike Count hoặc Visibility, rất nhỏ như Snowfall, Rainfall do đó ta cần scale lại data trước khi đưa vào mô hình để đảm bảo không có chỉ số nào dominate phần còn lại.

3.2. Tiền xử lý dữ liệu (Data preprocessing)

Để tiện cho sử dụng, ta quy ước tên các cột sẽ được sử dụng theo phong cách snake_case.

In [16]:
```
data_columns_changing = {"Date" : "date", "Rented Bike Count": "rented_bike_count", "Hou
                         "Temperature(°C)": "temperature", "Humidity(%)": "humidity", "Wind
                         "Visibility (10m)": "visibility", "Dew point temperature(°C)": "de
                         "Solar Radiation (MJ/m2)": "solar_radiation", "Rainfall(mm)": "rai
                         "Snowfall (cm)": "snowfall", "Seasons": "season", "Holiday": "holi
```

In [17]:
```
data.rename(columns= data_columns_changing, inplace= True)
```

In [18]:
```
data.head()
```

Out[18]:

	date	rented_bike_count	hour	temperature	humidity	wind_speed	visibility	dew_point_temperature	s
0	01/12/2017	254	0	-5.2	37	2.2	2000	-17.6	
1	01/12/2017	204	1	-5.5	38	0.8	2000	-17.6	
2	01/12/2017	173	2	-6.0	39	1.0	2000	-17.7	
3	01/12/2017	107	3	-6.2	40	0.9	2000	-17.6	
4	01/12/2017	78	4	-6.0	36	2.3	2000	-18.6	

In [19]:
```
data.tail()
```

Out[19]:

	date	rented_bike_count	hour	temperature	humidity	wind_speed	visibility	dew_point_temperature
8755	30/11/2018	1003	19	4.2	34	2.6	1894	-10.3
8756	30/11/2018	764	20	3.4	37	2.3	2000	-9.9
8757	30/11/2018	694	21	2.6	39	0.3	1968	-9.9
8758	30/11/2018	712	22	2.1	41	1.0	1859	-9.8
8759	30/11/2018	584	23	1.9	43	1.3	1909	-9.3

Chú ý rằng ngày tính theo thứ (Ví dụ thứ 2, thứ 3, chủ nhật) sẽ có ý nghĩa hơn là ngày tính theo số như bình thường (ngày 1, ngày 2, ngày 10,...) nên ta sẽ chuyển về dạng ngày tính theo thứ; tháng và năm thì vẫn như bình thường.

In [20]:
```
# Chuyển cột Date từ datatype Object(Dùng chung cho text) chuyển về dtype datetime
data["date"] = pd.to_datetime(data["date"], dayfirst = True) # Data gốc có dạng day-mont

# Chuyển ngày-tháng-năm thành 3 attributes riêng biệt
data["day_in_week"] = data["date"].dt.day_name() # ngày theo thứ
data["month"] = data["date"].dt.month_name()
data["year"] = data["date"].map(lambda x: x.year).astype("object")

# Bỏ đi column date ban đầu
data.drop(columns = ["date"], inplace = True)
```

In [21]:
```
# Kiểm tra lại data sau khi chuyển
data.head()
```

Out[21]:

	rented_bike_count	hour	temperature	humidity	wind_speed	visibility	dew_point_temperature	solar_radiatio
0	254	0	-5.2	37	2.2	2000	-17.6	0.
1	204	1	-5.5	38	0.8	2000	-17.6	0.
2	173	2	-6.0	39	1.0	2000	-17.7	0.
3	107	3	-6.2	40	0.9	2000	-17.6	0.
4	78	4	-6.0	36	2.3	2000	-18.6	0.

PHẦN IV. PHÂN TÍCH DỮ LIỆU (EXPLORATORY DATA ANALYSIS - EDA)

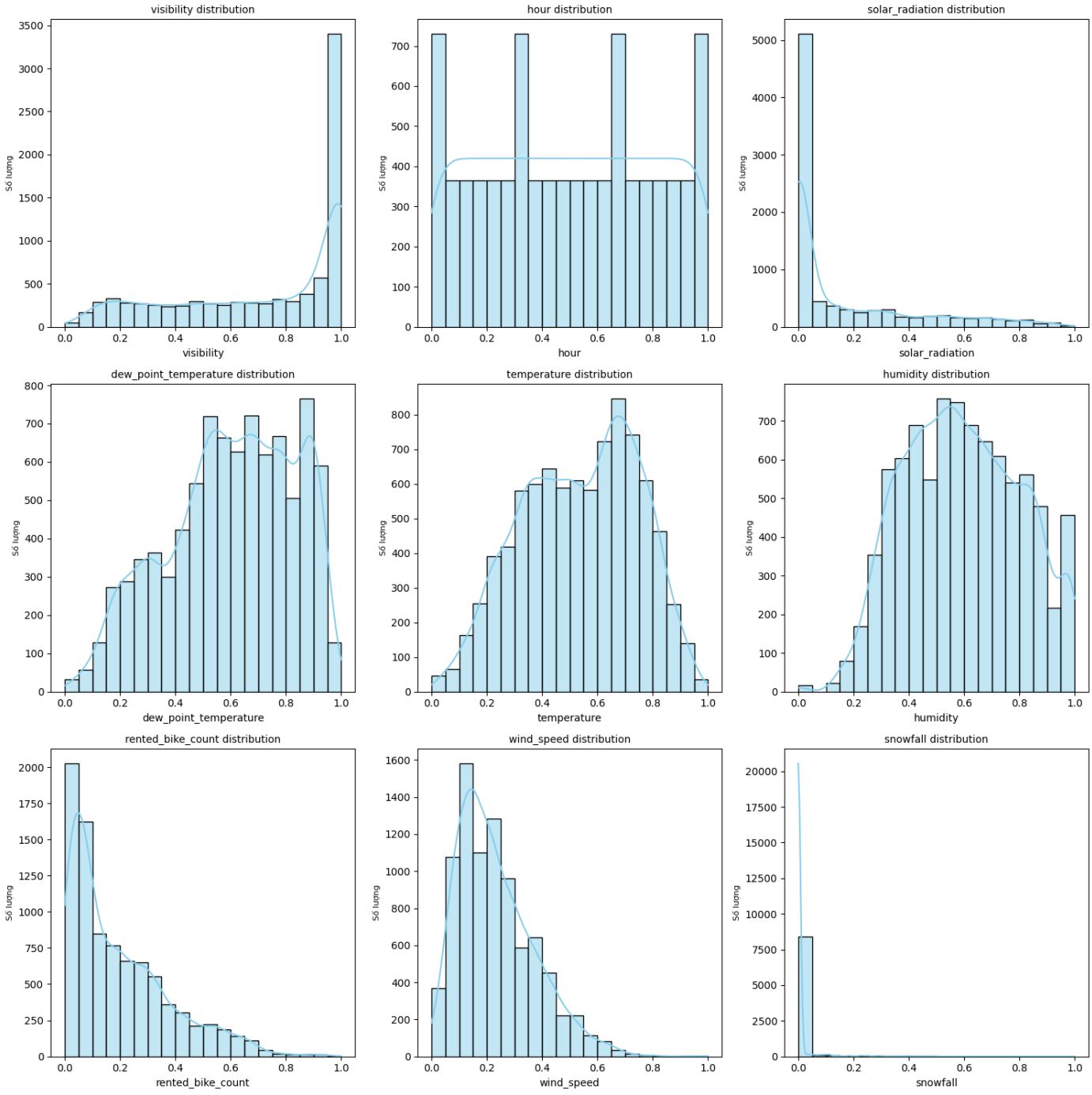
4.1. Phân bố theo biến (Distribution by attributes)

Chọn ra 9 biến cân bằng nhất (attributes balance) và vẽ bảng phân phối (distribution table)

In [22]:

```
from sklearn.preprocessing import MinMaxScaler

data_checking = data.loc[:, data.nunique() > 2] # Chỉ xem phân bố của những attributes co

# Xác định và chọn các cột dạng số để scaling (Identify and select numeric columns for s
numeric_columns = data_checking.select_dtypes(include = ['number']).columns

# Chuẩn hóa các dữ liệu dạng số về khoảng [0,1] (Standardize the numeric data to the ran
scaler = MinMaxScaler()
data_scaled = data_checking.copy()
data_scaled[numeric_columns] = scaler.fit_transform(data[numeric_columns])

# Tính toán độ lệch chuẩn (Calculate the standard deviation)
std_devs = data_scaled[numeric_columns].std()

# Sắp xếp các thuộc tính theo độ lệch chuẩn giảm dần (Sort attributes by standard deviat
sorted_attrs = std_devs.sort_values(ascending=False)

# Chọn 9 thuộc tính có độ lệch chuẩn cao nhất (Select the top 9 attributes with the high
top_9_high_std_attrs = sorted_attrs[:9]

# Tính điểm cân bằng của mỗi thuộc tính (Calculate balance score for each attribute)
balance_scores = data_scaled[numeric_columns].var() / data_scaled[numeric_columns].max()

# Sắp xếp các thuộc tính theo điểm cân bằng giảm dầnSort attributes by balance score in
sorted_balance_attrs = balance_scores.sort_values(ascending = False)  # Sort in descendi

# Chọn 9 thuộc tính cân bằng nhất (Select the top 9 most balanced attributes)
top_9_balanced_attrs = sorted_balance_attrs[:9]

# Tạo grid 3x3 chứa các subplot (Create a 3x3 grid of subplots)
fig, axes = plt.subplots(3, 3, figsize = (15, 15))
fig.subplots_adjust(hspace = 0.5)

"skyblue"
```

```
# Vẽ phân phối cho 9 thuộc tính cân bằng nhất với màu cụ thể(Draw the distributions for
for i, attr in enumerate(top_9_balanced_attrs.index):
    row, col = divmod(i, 3)
    ax = axes[row, col]
    sns.histplot(data = data_scaled, x = attr, bins = 20, kde = True, ax = ax, color = p
    ax.set_xlabel(attr, fontsize = 10)
    ax.set_ylabel(f"Số lượng", fontsize = 8)
    ax.set_title(f"{attr} distribution", fontsize = 10)

plt.tight_layout()
plt.savefig("most_balance_distribution.png")
plt.show()
```



4.2. Mối tương quan giữa các biến (Correlation matrix)

Tại phần này, nhóm mô tả correlation (tương quan) giữa các biến số (numeric attributes) và sử dụng các biểu đồ để khai phá dữ liệu (EDA)

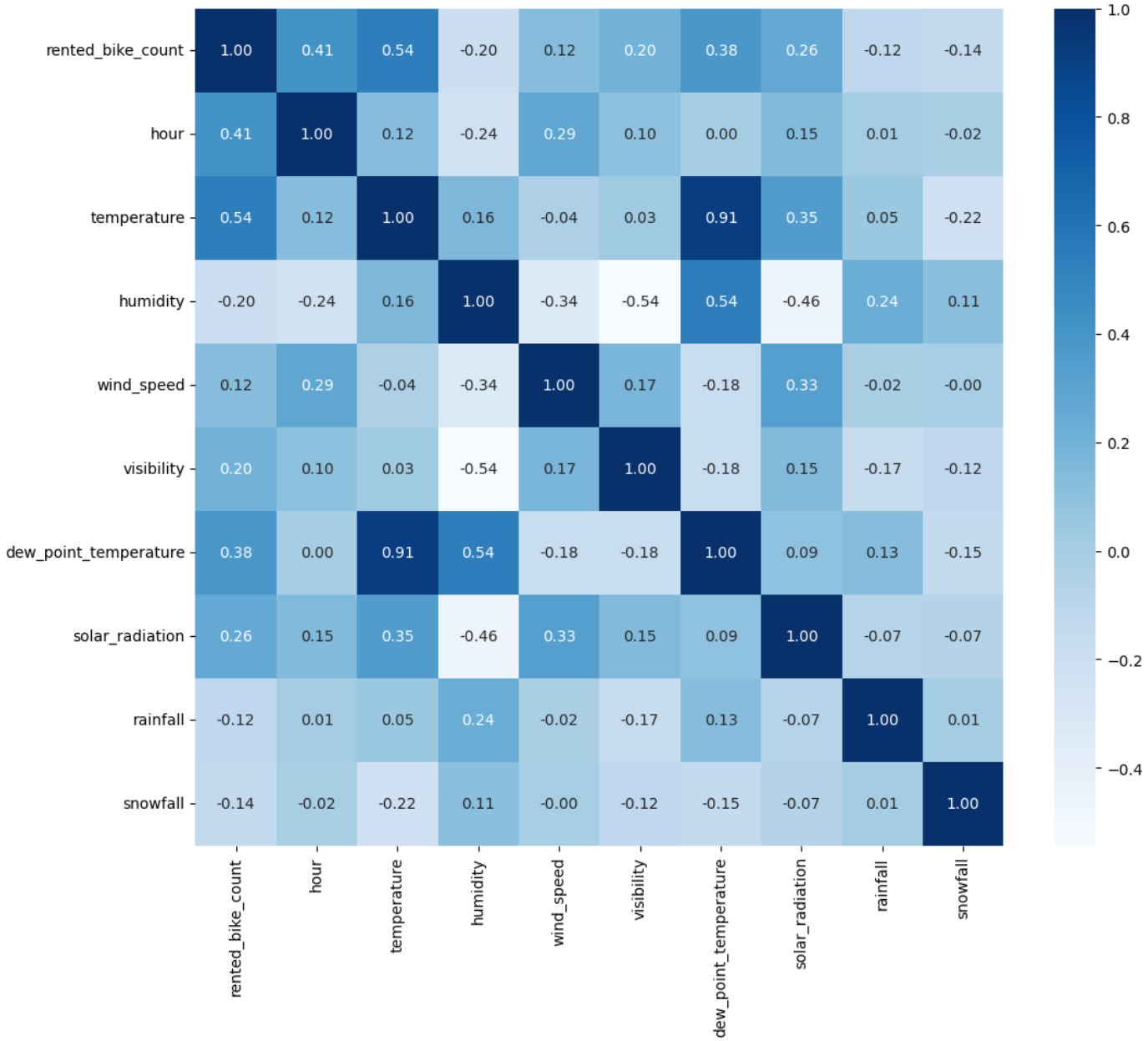
In [23]:
```
numeric_columns = data.select_dtypes(include = ['number'])
numeric_columns.corr()
```

Out[23]:

	rented_bike_count	hour	temperature	humidity	wind_speed	visibility	dew_point_
rented_bike_count	1.000000	0.410257	0.538558	-0.199780	0.121108	0.199280	
hour	0.410257	1.000000	0.124114	-0.241644	0.285197	0.098753	
temperature	0.538558	0.124114	1.000000	0.159371	-0.036252	0.034794	
humidity	-0.199780	-0.241644	0.159371	1.000000	-0.336683	-0.543090	
wind_speed	0.121108	0.285197	-0.036252	-0.336683	1.000000	0.171507	
visibility	0.199280	0.098753	0.034794	-0.543090	0.171507	1.000000	
dew_point_temperature	0.379788	0.003054	0.912798	0.536894	-0.176486	-0.176630	
solar_radiation	0.261837	0.145131	0.353505	-0.461919	0.332274	0.149738	
rainfall	-0.123074	0.008715	0.050282	0.236397	-0.019674	-0.167629	
snowfall	-0.141804	-0.021516	-0.218405	0.108183	-0.003554	-0.121695	

Tiếp theo, dùng heatmap để có một sự đánh giá trực quan tốt hơn

In [24]:
```
plt.figure(figsize = (12, 10))
sns.heatmap(numeric_columns.corr(), cmap = 'Blues', annot = True, fmt = '.2f')
plt.savefig('my_image.png')
plt.show()
```



Nhận xét

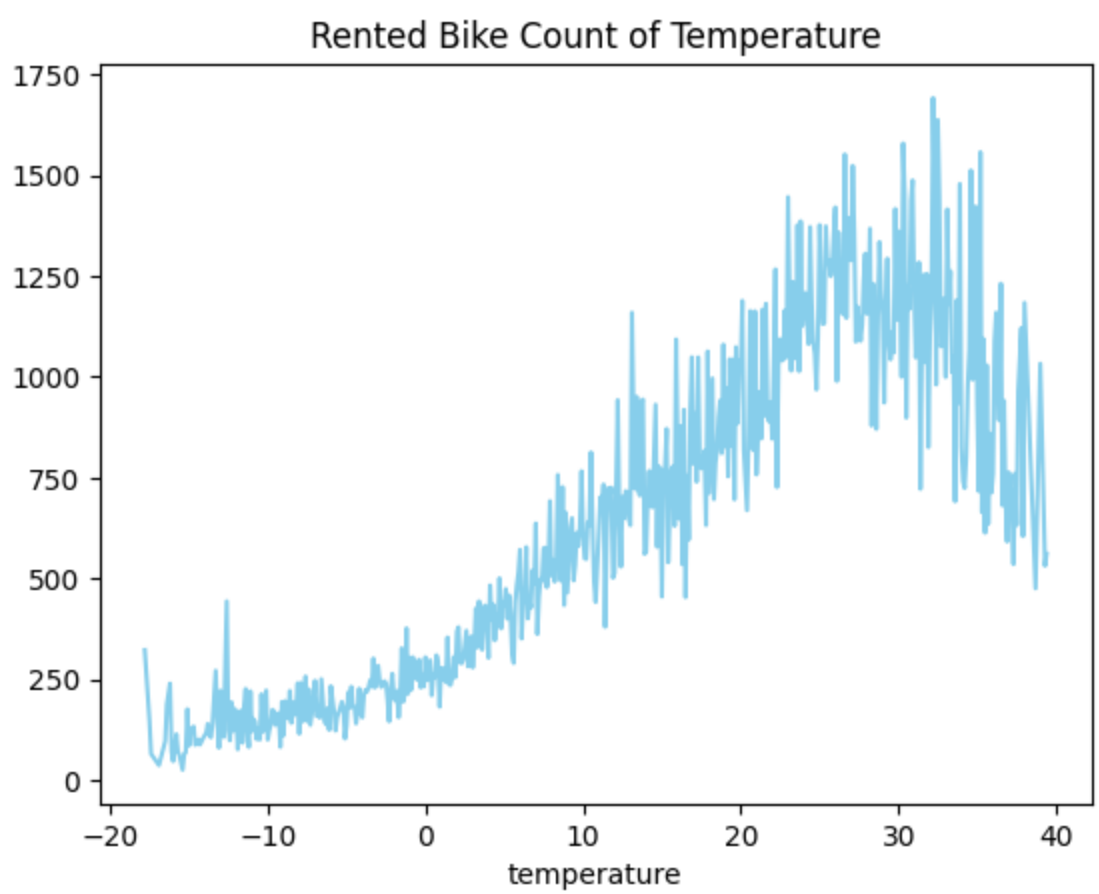
Dựa vào Correlation matrix, ta có thể thấy biến temperature có correlation với biến rented_bike_count tốt nhất (0.54). Correlation dương và khá mạnh này cho thấy rented_bike_count phụ thuộc lớn nhất vào các chỉ temperature (tức là temparature càng cao thì rented_bike_count càng lớn).

Ngoài ra, biến hour cũng có tương quan khá tốt với biến rented_bike_count (0.41).

Hầu hết các biến còn lại có correlation nhỏ, chứng tỏ các biến ít ảnh hưởng với nhau theo mối quan hệ tuyến tính.

In [25]:
```
groupby_temperature = numeric_columns.copy()
groupby_temperature.groupby('temperature').mean()['rented_bike_count'].plot.line(color =
plt.title('Rented Bike Count of Temperature')
```

Out[25]:
```
Text(0.5, 1.0, 'Rented Bike Count of Temperature')
```



Nhận xét

Số lượng xe được thuê tăng khi nhiệt độ tăng và đạt đỉnh ở khoảng 25 - 30 độ C.

Nhóm nhận định rằng đây là mức nhiệt độ dễ chịu và mát mẻ nên mọi người có xu hướng thuê xe đạp cao hơn để phục vụ nhu cầu đi dạo, đi chơi,...

Tuy nhiên , khi nhiệt độ quá cao (30 - 40 độ C) thì lượng thuê xe đạp giảm do thời tiết quá nóng và mọi người không có nhu cầu ra ngoài.

4.3. Thống kê mô tả (Descriptive statistics)

In [26]:
```
# Mô tả các biến
data.describe()
```

Out[26]:

	rented_bike_count	hour	temperature	humidity	wind_speed	visibility	dew_point_temperat
count	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000
mean	704.602055	11.500000	12.882922	58.226256	1.724909	1436.825799	4.073
std	644.997468	6.922582	11.944825	20.362413	1.036300	608.298712	13.060
min	0.000000	0.000000	-17.800000	0.000000	0.000000	27.000000	-30.600
25%	191.000000	5.750000	3.500000	42.000000	0.900000	940.000000	-4.700
50%	504.500000	11.500000	13.700000	57.000000	1.500000	1698.000000	5.100
75%	1065.250000	17.250000	22.500000	74.000000	2.300000	2000.000000	14.800
max	3556.000000	23.000000	39.400000	98.000000	7.400000	2000.000000	27.200

Nhận xét

3 biến Solar Radiation, Rainfall, Snowfall có mean thấp, 25%, 50% gần như sấp xỉ không.

Điều này chứng tỏ rằng dữ liệu được lấy từ khu vực có thời tiết khá tốt: bức xạ mặt trời (không nắng gắt), mưa và tuyết bằng 0 hoặc sấp xỉ bằng không (mức độ nhẹ).

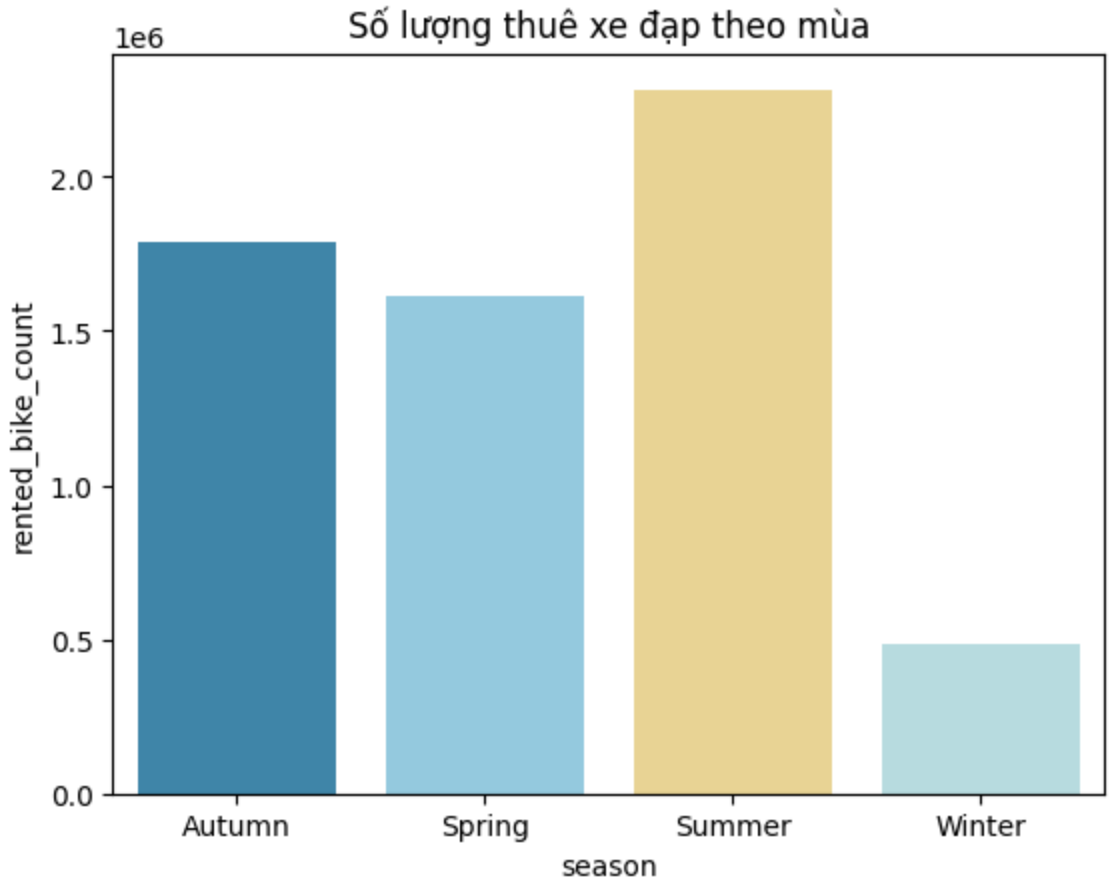
Nhiệt độ trung bình khoảng 13 độ C nên có thể dữ liệu được lấy từ vùng có khí hậu ôn hòa.

4.3.1. Xu hướng thuê xe đạp theo mùa như thế nào?

Bike_rented_count có thay đổi nhiều theo mùa không? Nhu cầu của người dân Seoul khi thuê xe đạp sẽ nhiều hơn vào mùa nóng hay mùa lạnh?

In [27]:
```
# Thống kê số lượng thuê xe theo từng mùa
season_trends = data.groupby('season')['rented_bike_count'].sum().reset_index()

# Bảng màu (color palette)
colors = ['#2D8BBA', '#87CEEB', '#F6DA86', '#B0E0E6']

# Vẽ bar graph
sns.barplot(x = 'season', y = 'rented_bike_count', data = season_trends, palette = color
plt.title('Số lượng thuê xe đạp theo mùa')
plt.show()
```

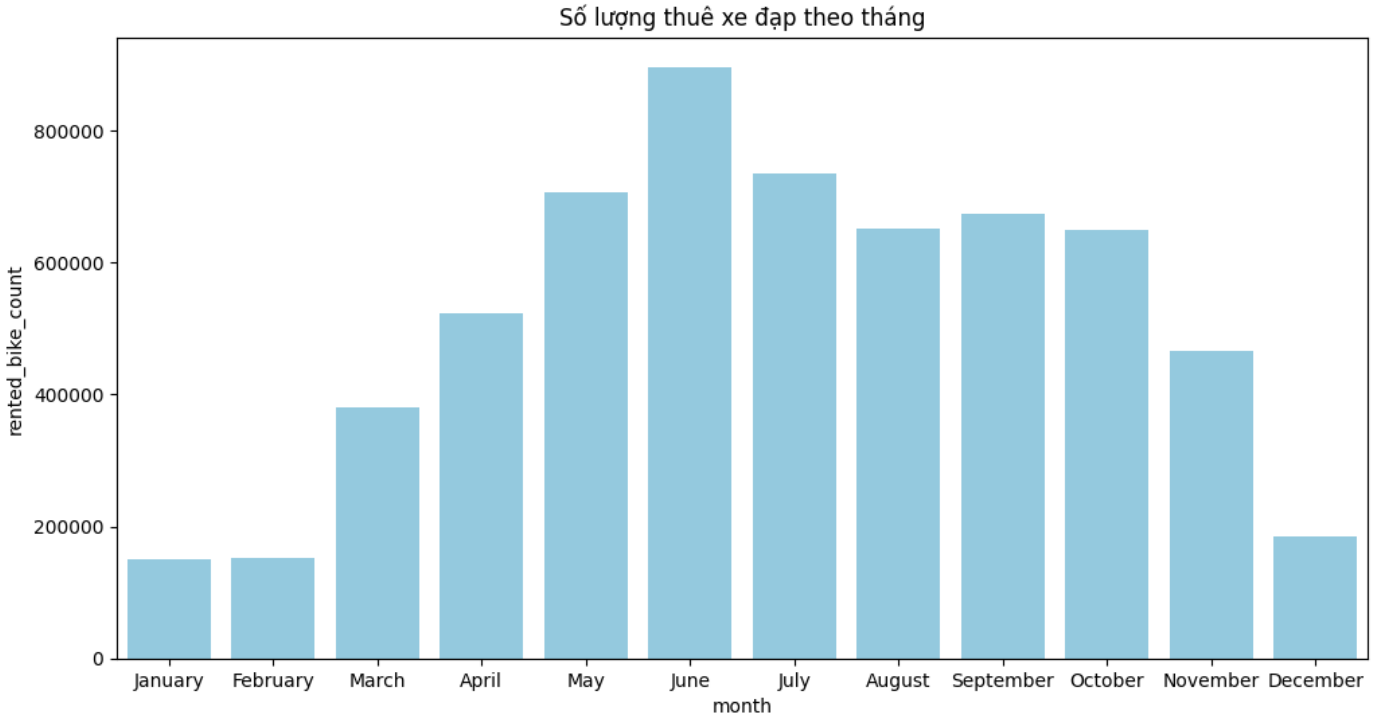
```
C:\Users\DMX\AppData\Local\Temp\ipykernel_17860\816290194.py:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x = 'season', y = 'rented_bike_count', data = season_trends, palette = col
ors)
```



Nhận xét

Có thể thấy số lượng sử dụng Rental Bike lớn hơn hẳn vào mùa nóng so với mùa lạnh (summer-winter), đúng với nhận xét từ Correlation matrix ở trên, đó là nhiệt độ càng cao, số lượng thuê xe đạp càng lớn

In [28]:
```
import calendar

# Tiếp tục thống kê chi tiết số lượng thuê xe theo từng tháng
season_trends = data.groupby('month')['rented_bike_count'].sum().reset_index()

# Để các cột sắp xếp theo thứ tự thời gian trong năm, không phải theo thứ tự alphabet cu
months_order = {month: index for index, month in enumerate(calendar.month_name[1:], 1)}
season_trends['month'] = season_trends['month'].map(months_order)
season_trends = season_trends.sort_values('month')

# Vẽ bar graph
plt.figure(figsize = (12, 6))
sns.barplot(x = 'month', y = 'rented_bike_count', data = season_trends, color = '#87CEEB
plt.xticks(range(12), calendar.month_name[1:])

plt.title('Số lượng thuê xe đạp theo tháng')
plt.show()
```

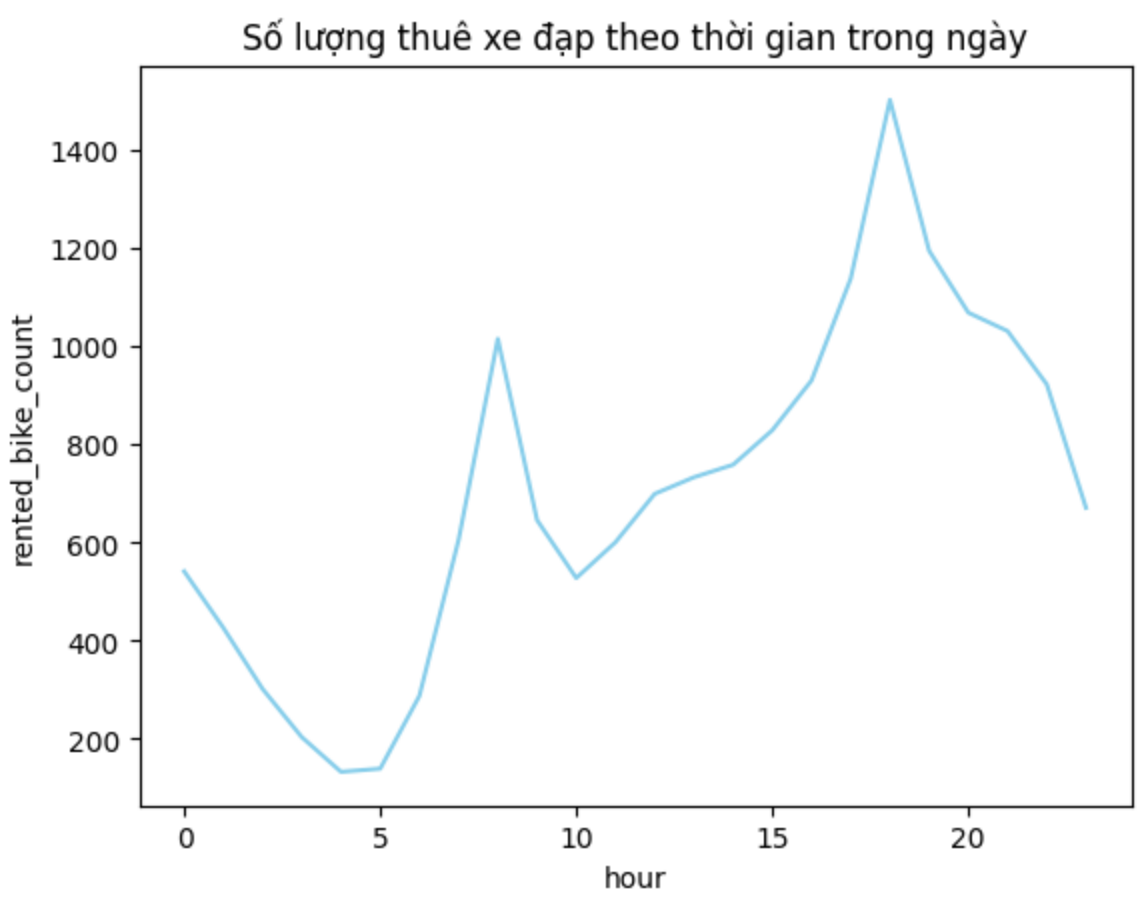


Nhận xét

Có thể thấy nhu cầu thuê xe phân bố theo tháng trong năm gần như tuân theo phân phối chuẩn, với giá trị lớn nhất vào giữa năm (các tháng mùa hè, nhiệt độ cao), giảm dần và ít nhất vào cuối năm và đầu năm.

4.3.2. Những giờ nào là giờ cao điểm, nhu cầu đối với việc thuê xe đạp là lớn nhất?

In [29]:
```
# Thống kê số lượng thuê xe theo hour (Grouping by hour)
hourly_pattern = data.groupby('hour')['rented_bike_count'].mean().reset_index()

sns.lineplot(x = 'hour', y = 'rented_bike_count', data = hourly_pattern, color = "#87cee
plt.title('Số lượng thuê xe đạp theo thời gian trong ngày')
plt.show()
```

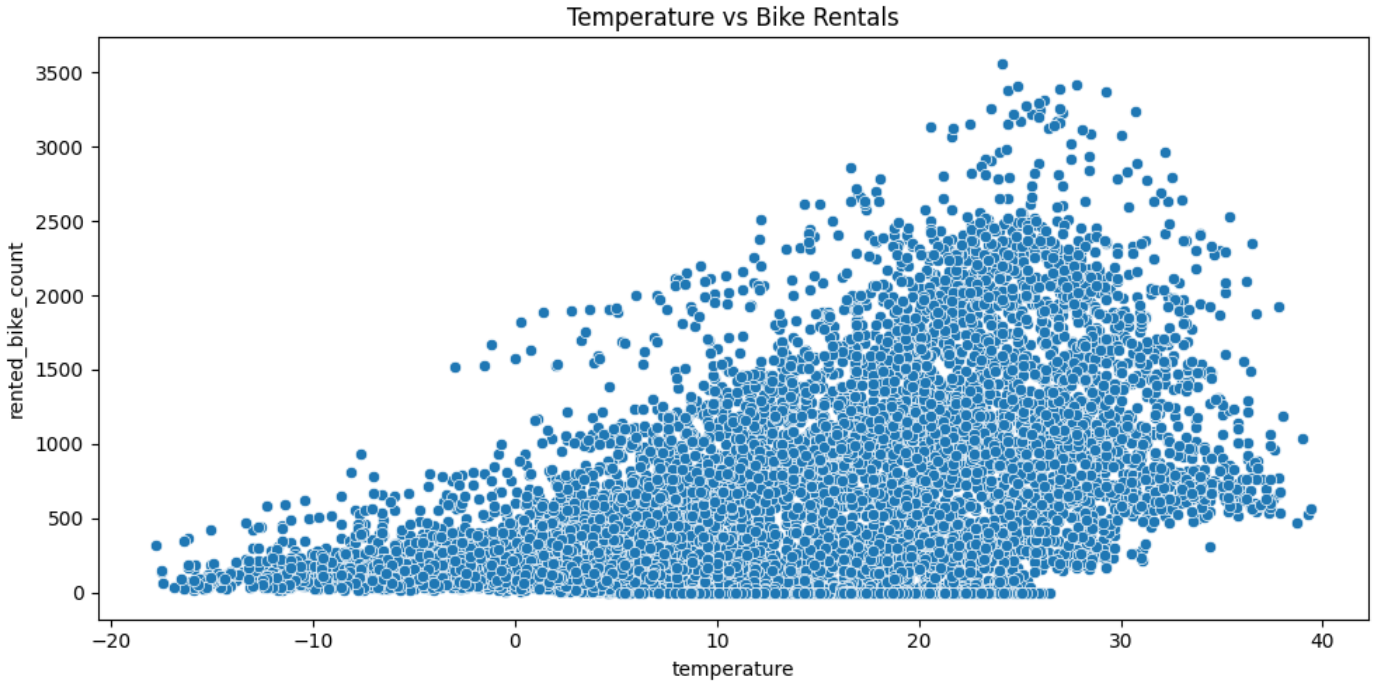


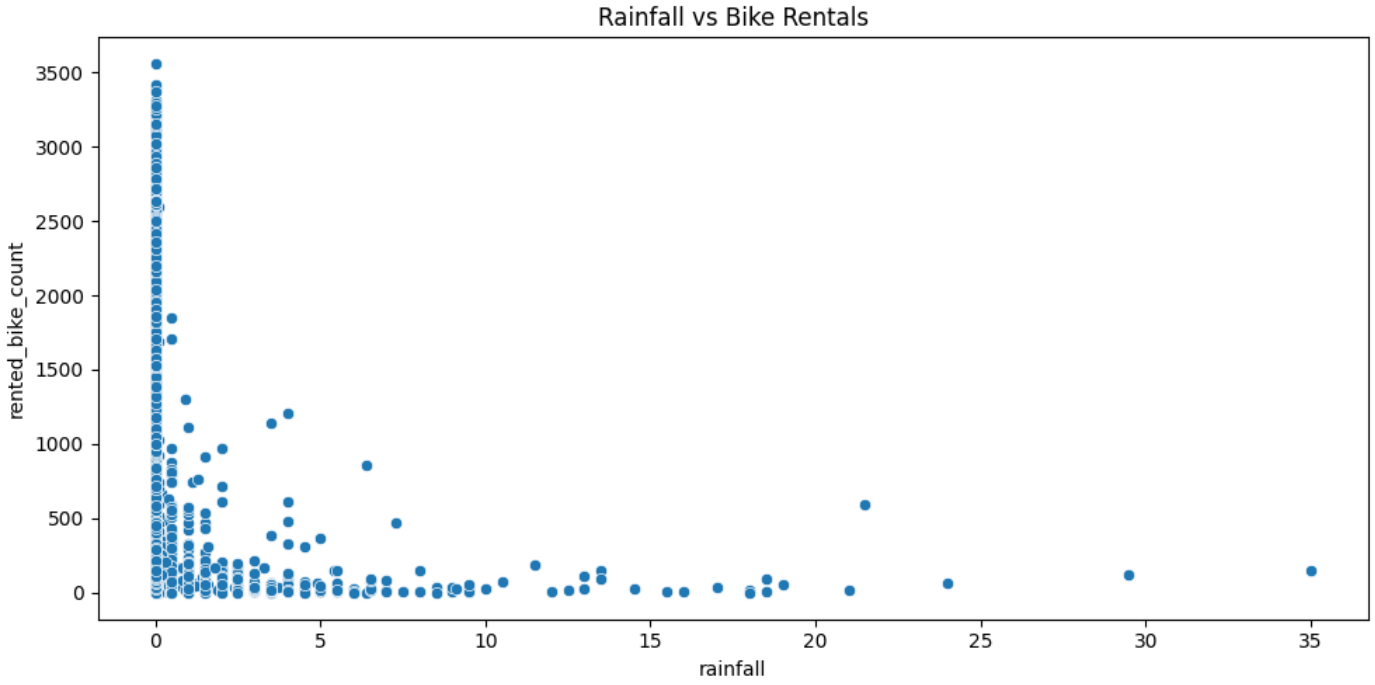
Nhận xét

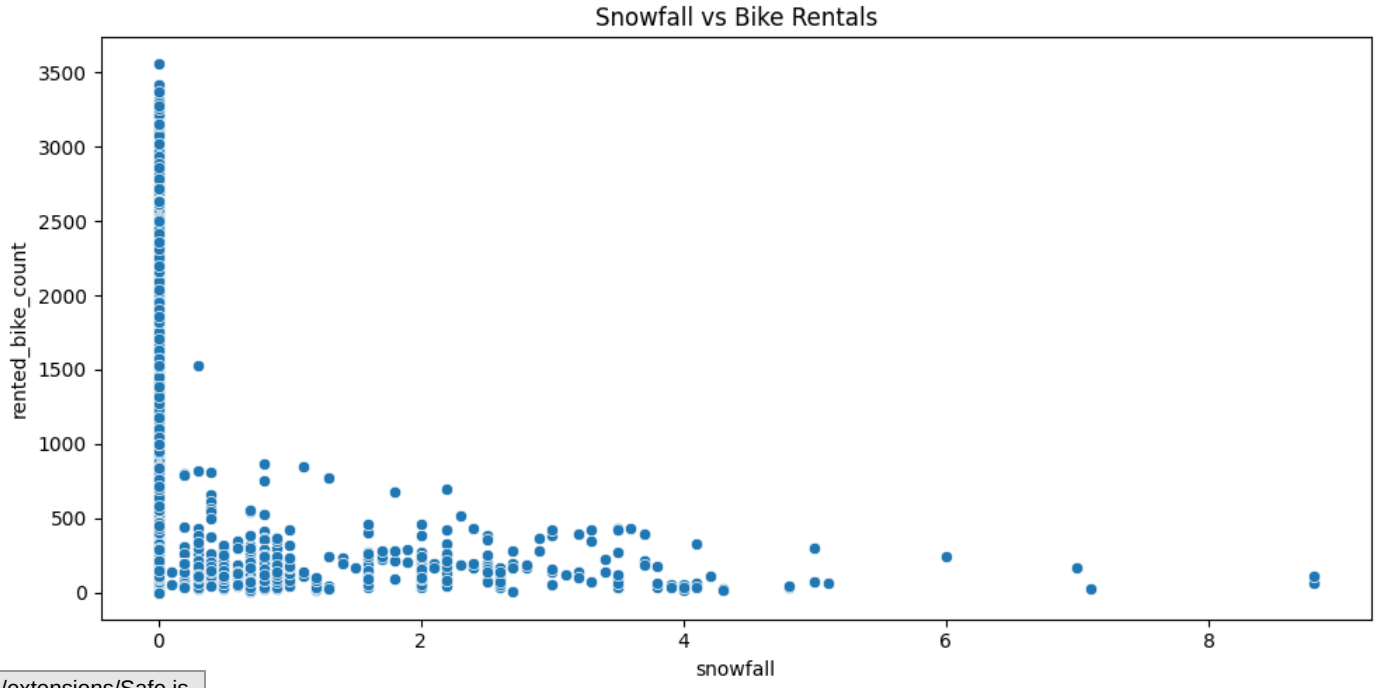
Khung giờ cao điểm nhất trong ngày đối với việc thuê xe đạp đó là vào khoảng chiều tối trong ngày (17 - 19h), ngoài ra thì khoảng thời gian 8 - 9h trong ngày cũng có nhu cầu tăng đột biến đối với những khung giờ gần đó, có thể do đây là khoảng thời gian di chuyển đến nơi làm việc của hầu hết mọi người.

4.3.3. Tác động của những yếu tố thời tiết đến nhu cầu thuê xe đạp như thế nào?

In [30]:
```
# Vẽ các biểu đồ so sánh số lượng thuê xe với các điều kiện thời tiết khác nhau (Plottin
fig, axs = plt.subplots(nrows = 3, ncols = 1, figsize = (10, 15))

# Biểu đồ scatterplot của biến temperature và biến rented_bike_count
sns.scatterplot(x = 'temperature', y = 'rented_bike_count', data = data, ax = axs[0])
axs[0].set_title('Temperature vs Bike Rentals')

# Biểu đồ scatterplot của biến rainfall và biến rented_bike_count
sns.scatterplot(x = 'rainfall', y = 'rented_bike_count', data = data, ax = axs[1])
axs[1].set_title('Rainfall vs Bike Rentals')

# Biểu đồ scatterplot của biến snowfall và biến rented_bike_count
sns.scatterplot(x = 'snowfall', y = 'rented_bike_count', data = data, ax = axs[2])
axs[2].set_title('Snowfall vs Bike Rentals')

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







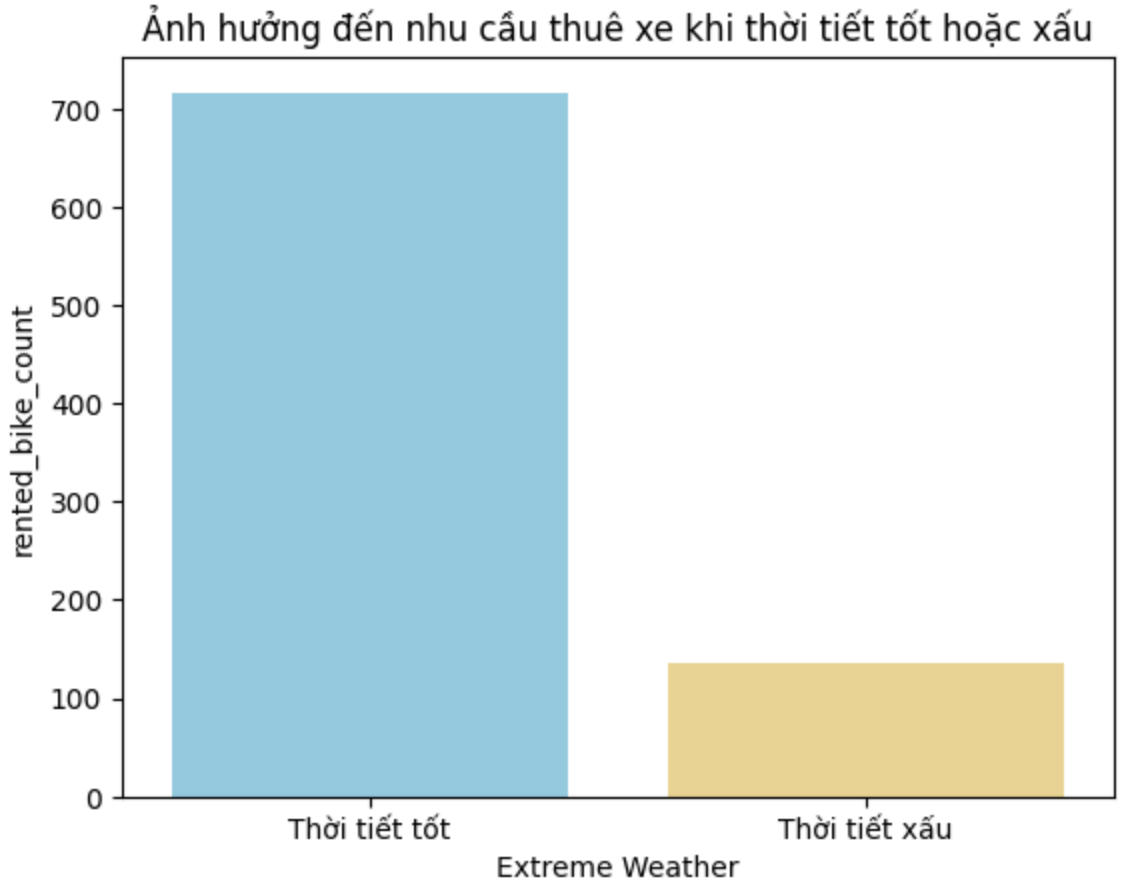
Nhận xét

Có thể thấy ngoài ảnh hưởng của nhiệt độ, rainfall và snowfall cũng ảnh hưởng rất lớn tới nhu cầu thuê xe. Đối với những ngày có mức snowfall và rainfall cao, hầu như có rất ít người thuê xe đạp, và hầu hết nhu cầu thuê xe đạp chỉ tập trung vào những ngày có snowfall = 0 và rainfall = 0.

Coi thời tiết xấu là khi có snowfall > 2 hoặc rainfall > 5, nhóm phân tích sự khác biệt rất lớn về nhu cầu thuê xe đạp khi thời tiết thuận lợi và khi thời tiết xấu.

In [31]:
```
df = data.copy(deep=True)

# Thêm biến extreme weather (thời tiết khắc nghiệt và nó tương đương thời tiết xấu)
df['Extreme Weather'] = (df['rainfall'] > 5) | (df['snowfall'] > 2)
df['Extreme Weather'] = df['Extreme Weather'].map({True : 'Thời tiết xấu', False : 'Thời
extreme_weather_impact = df.groupby('Extreme Weather')["rented_bike_count"].mean().reset

# Vẽ bar graph so sánh nhu câù thuê xe khi thời tiết tốt và xấú
colors = ['#87CEEB', '#F6DA86']
sns.barplot(x = 'Extreme Weather', y = 'rented_bike_count', data = extreme_weather_impac
plt.title('Ảnh hưởng đến nhu cầu thuê xe khi thời tiết tốt hoặc xấu')
plt.show()
```

```
C:\Users\DMX\AppData\Local\Temp\ipykernel_17860\3617429811.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x = 'Extreme Weather', y = 'rented_bike_count', data = extreme_weather_imp
act, palette = colors)
```

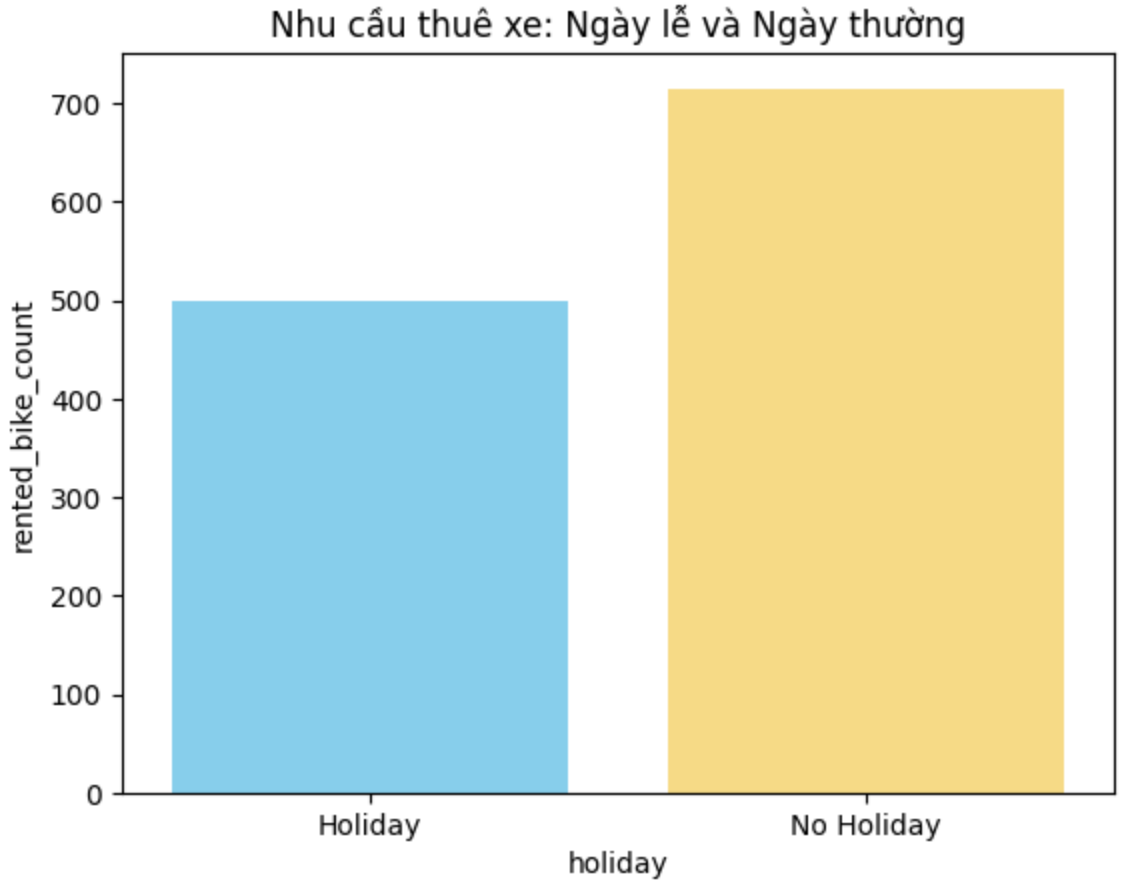


Nhận xét

Có thể thấy thời tiết xấu (Rainfall, Snowfall) ảnh hưởng trực tiếp đến quyết định thuê xe của người dùng, khi mà trung bình số lượng thuê xe khi thời tiết tốt gấp khoảng 7 lần khi gặp thời tiết xấu.

4.3.4. Nhu cầu thuê xe vào ngày lễ có khác gì với ngày thường?

In [32]:
```
# Thống kê số lượng thuê xe theo ngay lễ và ngày thường
holiday_effect = data.groupby('holiday')['rented_bike_count'].mean().reset_index()

# Vẽ bar graph
barplot = sns.barplot(x = 'holiday', y = 'rented_bike_count', data = holiday_effect)

barplot.patches[0].set_facecolor('#87CEEB')
if len(barplot.patches) > 1:
    barplot.patches[1].set_facecolor('#F6DA86')

plt.title('Nhu cầu thuê xe: Ngày lễ và Ngày thường')
plt.show()
```



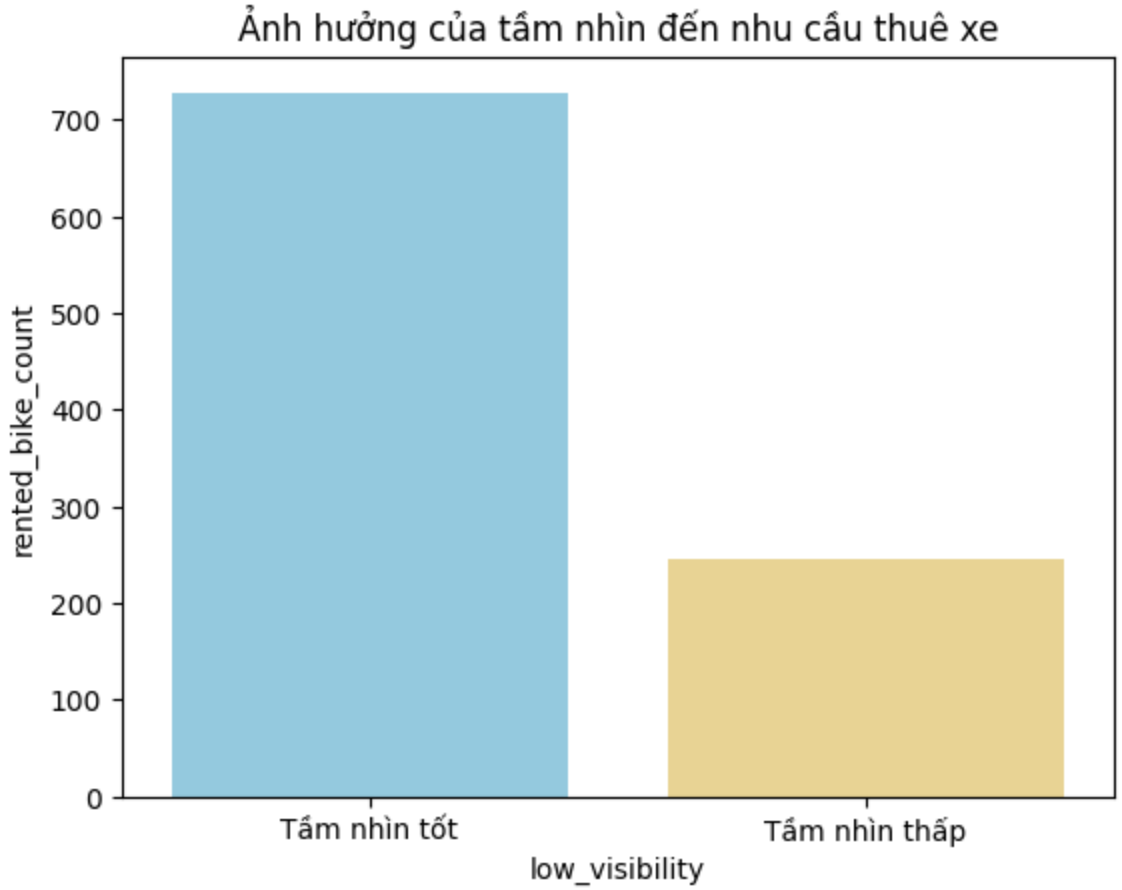
Nhận xét

Trái với dự đoán của nhóm thì nhu cầu thuê xe vào ngày thường lại lớn hơn so với nhu cầu thuê xe vào ngày lễ, kết hợp với việc thống kê ở trên cho thấy nhu cầu thuê xe cao bất thường vào khung giờ đi làm vào buổi sáng (8 - 9h) và sau khi tan làm vào buổi chiều (17 - 19h), ta có thể rút ra nhận xét có một lượng đáng kể người dân sử dụng xe đạp thuê để đi làm.

4.3.5. Ảnh hưởng của Chỉ số Tầm nhìn xa (Sự an toàn đối với người đi xe đạp) đến nhu cầu thuê xe đạp như thế nào?

In [33]:
```
data_copy = data.copy(deep=True)

# Tạo biến low_visibility chỉ tầm nhìn thấp khi visibility < 300 (nhóm nhận định)
data_copy['low_visibility'] = data_copy['visibility'] < 300
visibility_impact = data_copy.groupby('low_visibility')['rented_bike_count'].mean().rese

visibility_impact['low_visibility'] = visibility_impact['low_visibility'].map({False : '

# Vẽ bar graph
barplot = sns.barplot(x = 'low_visibility', y = 'rented_bike_count', data = visibility_i

plt.title('Ảnh hưởng của tầm nhìn đến nhu cầu thuê xe')
plt.show()
```

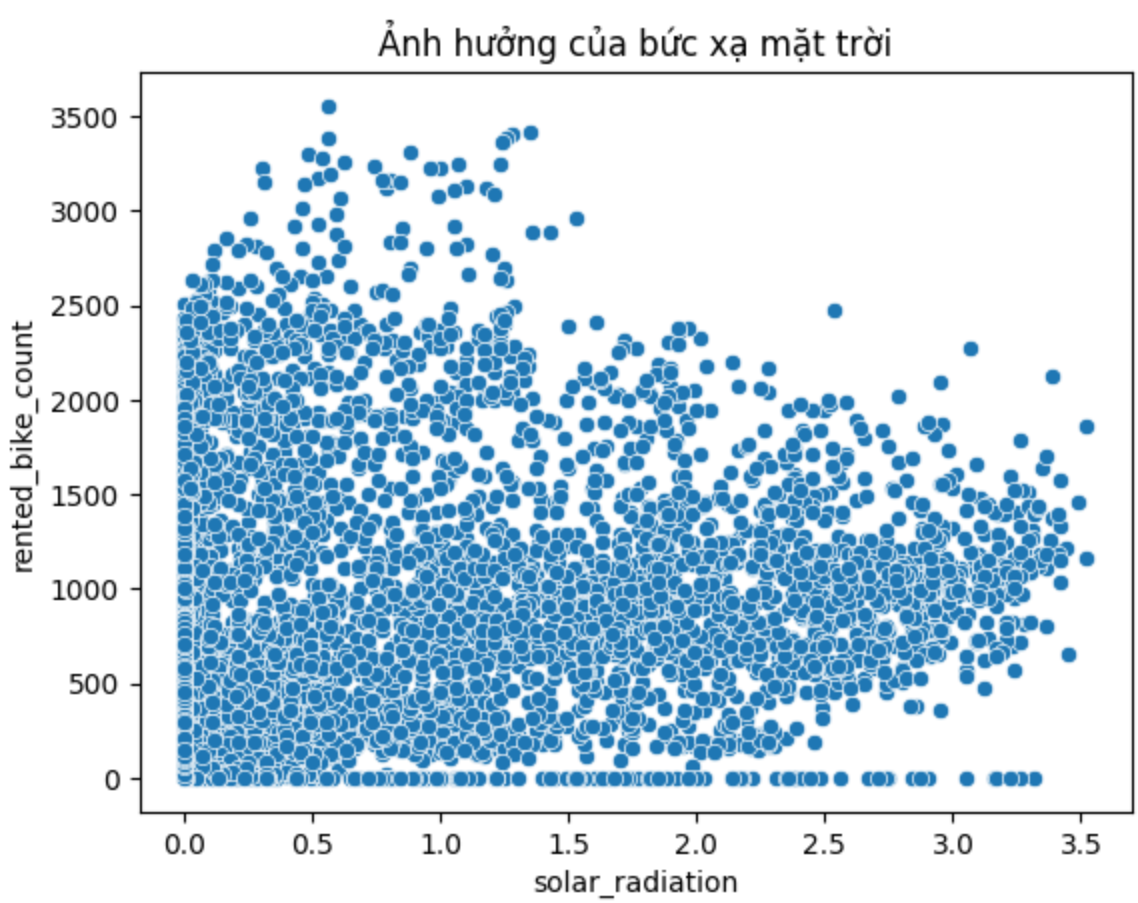
```
C:\Users\DMX\AppData\Local\Temp\ipykernel_17860\2368083855.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  barplot = sns.barplot(x = 'low_visibility', y = 'rented_bike_count', data = visibility
_impact, palette = ['#87CEEB', '#F6DA86'])
```



Nhận xét

Đúng như dự đoán, nhu cầu thuê xe giảm mạnh vào thời điểm mà tầm nhìn kém (nhóm chọn threshold là tầm nhìn dưới 300m là tầm nhìn kém), điều này cho thấy người dùng có lo lắng đến vấn đề an toàn khi sử dụng xe đạp khi quyết định thuê xe đạp.

In [34]:
```
# Vẽ scatterplot của biến solar_radiation và biến rented_bike_count

sns.scatterplot(x = 'solar_radiation', y = 'rented_bike_count', data = data)
plt.title('Ảnh hưởng của bức xạ mặt trời')
```



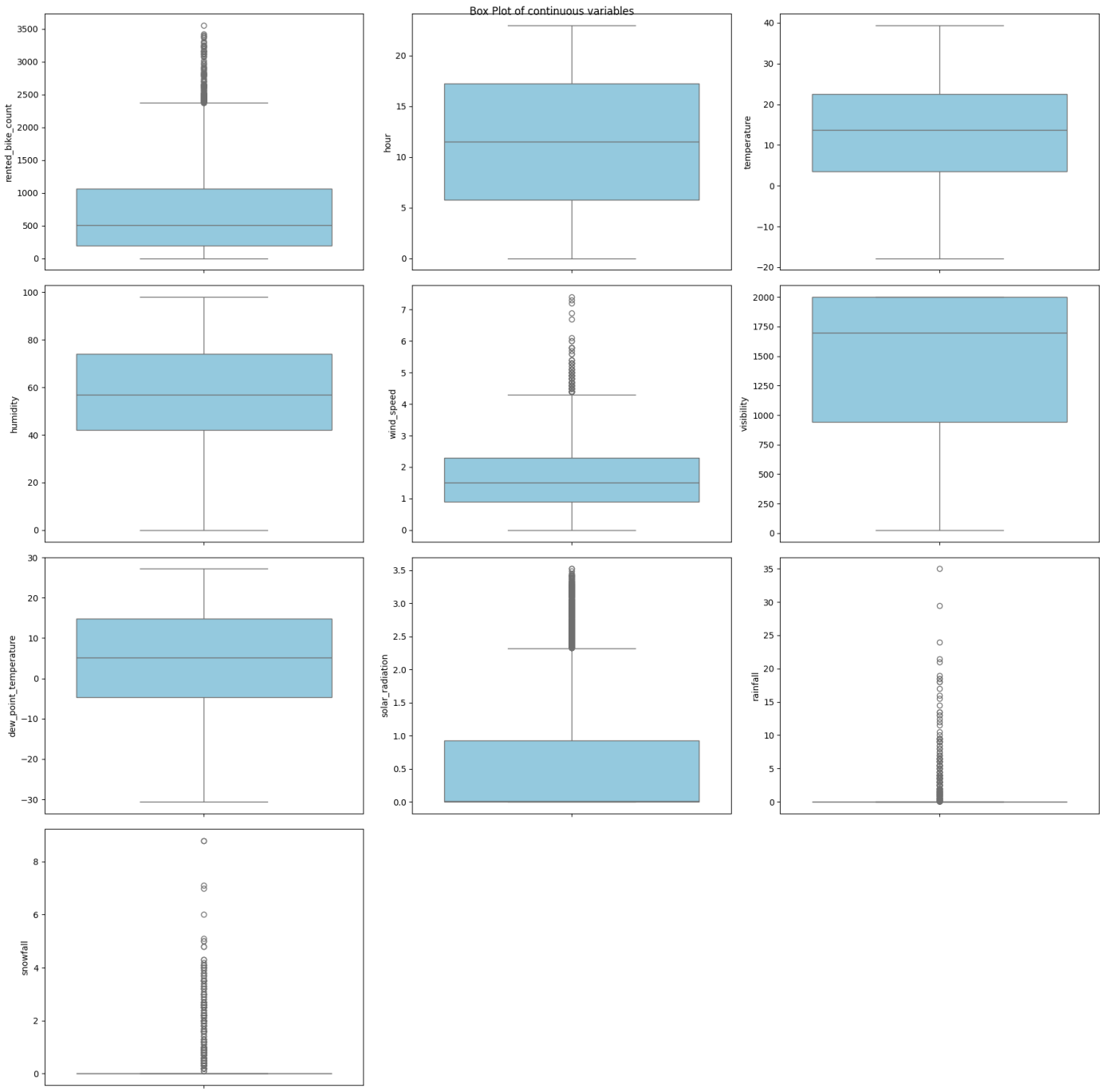
Nhận xét

Có thể thấy nhu cầu thuê xe cao hơn một chút đối với những ngày có bức xạ mặt trời thấp, tuy nhiên sự chênh lệch này là không đáng kể. Điều này cho thấy người dùng không thực sự quan tâm đến ảnh hưởng của bức xạ mặt trời đến quá trình sử dụng xe đạp, khác hẳn đối với lo ngại về yếu tố ảnh hưởng trực tiếp đến sự an toàn hơn như là Tầm nhìn xa nhóm đã phân tích ở trên.

4.4. Loại bỏ giá trị ngoại lai (Reduce outliers)

Ở phần tiếp theo nhóm sử dụng boxplot để biểu thị từng biến và xác định outlier để có cách xử lý phù hợp ở các bước tiếp theo

In [35]:
```
# Box plot của các biến
plt.figure(figsize = (18, 18))
for i, col in enumerate(data.select_dtypes(include = ['float64','int64']).columns):
    plt.rcParams['axes.facecolor'] = 'white'
    ax = plt.subplot(4,3, i+1)
    sns.boxplot(data = data, y = col, ax = ax,color = '#87CEEB')

plt.suptitle('Box Plot of continuous variables')
plt.tight_layout()
plt.show()
```



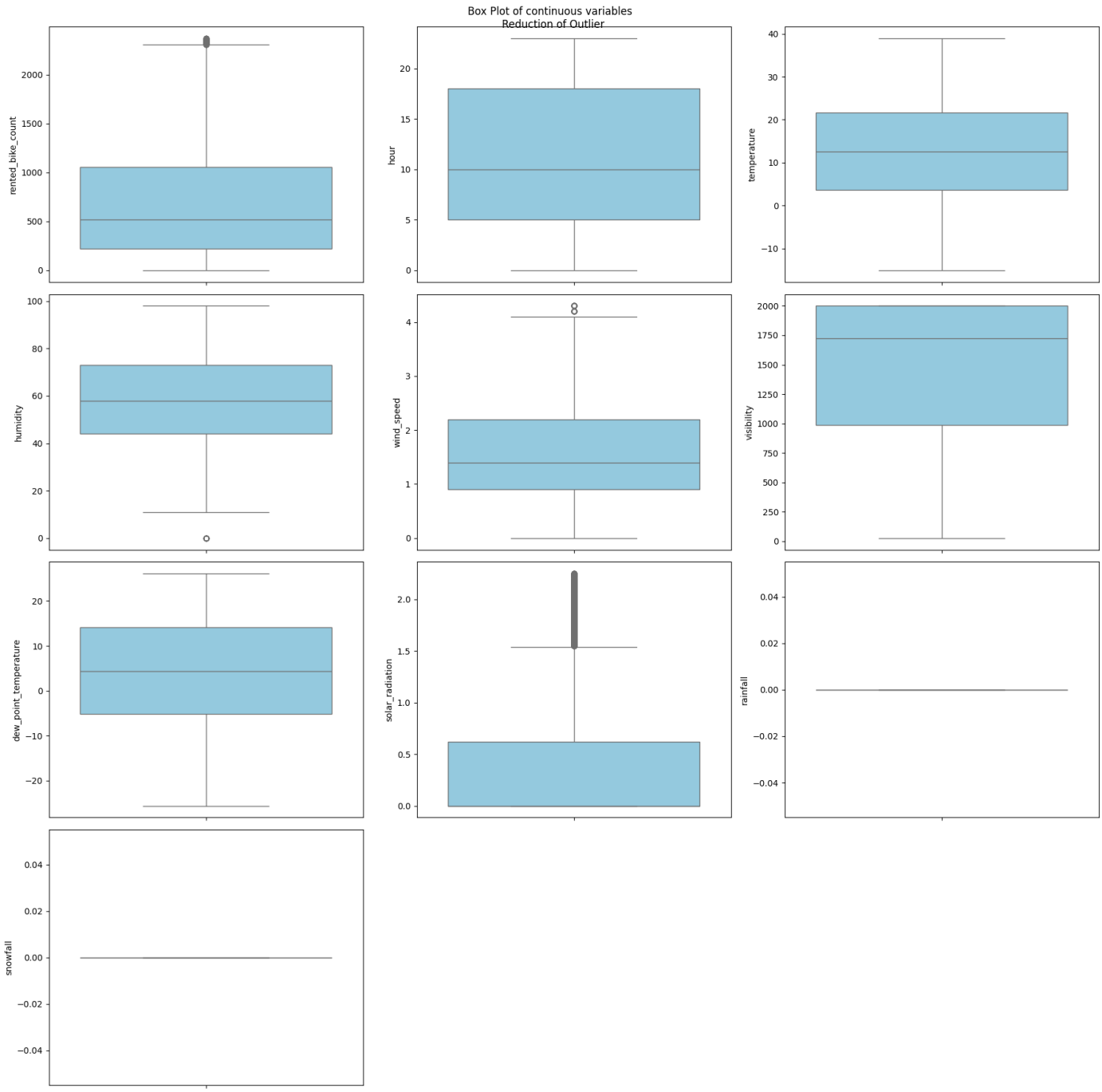
In [36]:
```python
# Định nghĩa hàm remove_outliers để loại bỏ các outliers

def remove_outliers(df: pd.DataFrame, feat: str):
    """
    """
    feat_fraud = df[feat].values
    q25, q75 = np.percentile(feat_fraud, 25), np.percentile(feat_fraud, 75)
    iqr = q75 -q25

    cut_off = iqr * 1.5
    lower, upper = q25 - cut_off, q75 + cut_off

    outliers = [x for x in feat_fraud if x < lower or x > upper]
    print(f'Feature {feat} Outliers for Fraud Cases: {len(outliers)}')

    return df.drop(df[(data[feat] > upper) | (df[feat] < lower)].index)
```

In [37]:
```python
# Sử dụng hàm remove_outliers vừa tạo để loại bỏ outliers

for i in data.select_dtypes(include=['float64','int64']).columns:
    _outliers(data,i)
```

```
Feature rented_bike_count Outliers for Fraud Cases: 158
Feature hour Outliers for Fraud Cases: 0
Feature temperature Outliers for Fraud Cases: 0
Feature humidity Outliers for Fraud Cases: 0
Feature wind_speed Outliers for Fraud Cases: 161
Feature visibility Outliers for Fraud Cases: 0
Feature dew_point_temperature Outliers for Fraud Cases: 0
Feature solar_radiation Outliers for Fraud Cases: 681
Feature rainfall Outliers for Fraud Cases: 512
Feature snowfall Outliers for Fraud Cases: 398
```

In [38]:

```
# Box plot của các biến sau khi đã lọc outliers

plt.figure(figsize=(18, 18))
for i, col in enumerate(data.select_dtypes(include=['float64','int64']).columns):
    plt.rcParams['axes.facecolor'] = 'white'
    ax = plt.subplot(4,3, i+1)
    sns.boxplot(data=data, y=col, ax=ax,color='#87CEEB')
plt.suptitle('Box Plot of continuous variables \n Reduction of Outlier')
plt.tight_layout()
plt.show()
```



4.5. Xác định biến quan trọng (Identify important variable)

Tại phần này nhóm xác định biến quan trọng bằng chỉ số The Pearson Correlation Coefficient và P-value

In [39]:
```
data.shape[1]
```

Out[39]:
```
16
```

Nhận xét

Các biến có chỉ số Correlation không gần 0 và có P-value < 0.05 nên các biến đều quan trọng và nhóm sử dụng tất cả các biến để phát triển model vì không muốn bỏ sót biến khiến mô hình trở nên không chính xác.

PHẦN V. TRIỂN KHAI MÔ HÌNH (MODEL DEVELOPMENT)

5.1. Xử lý dữ liệu (Data processing)

In [40]:
```
process_data = data.copy()
```

In [41]:
```
process_data.head(3)
```

Out[41]:

	rented_bike_count	hour	temperature	humidity	wind_speed	visibility	dew_point_temperature	solar_radiatio
0	254	0	-5.2	37	2.2	2000	-17.6	0.
1	204	1	-5.5	38	0.8	2000	-17.6	0.
2	173	2	-6.0	39	1.0	2000	-17.7	0.

ONE HOT ENCODING

In [42]:
```
process_data.drop('season',axis=1,inplace=True)
```

In [43]:
```
process_data = pd.get_dummies(
    process_data,
    columns = ['holiday','functioning_day','month','year','day_in_week'],
)
```

In [44]:
```
process_data.head(2)
```

Out[44]:

	rented_bike_count	hour	temperature	humidity	wind_speed	visibility	dew_point_temperature	solar_radiatio
0	254	0	-5.2	37	2.2	2000	-17.6	0.
1	204	1	-5.5	38	0.8	2000	-17.6	0.

2 rows × 35 columns

In [45]:
```
len(process_data.columns)
```

Out[45]:
```
35
```

Chia tập dữ liệu train/test

In [46]:
```
from sklearn.model_selection import train_test_split
X, Y = np.array(process_data.drop('rented_bike_count', axis=1)), np.array(process_data['
X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size=0.1,random_state=42)
```

In [47]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[47]:
```
((6165, 34), (685, 34), (6165,), (685,))
```

5.2. Xây dựng mô hình (Model construction)

In [48]:
```
!pip install xgboost -q
!pip install polars -q
from xgboost import XGBRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score, explained
import polars as pl
import math
```

```
[notice] A new release of pip is available: 23.3.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip

[notice] A new release of pip is available: 23.3.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Phương pháp Grid Search: Tìm bộ hệ số tốt nhất cho Model

In [49]:
```
params_gbr = {
    'loss': ['squared_error'],      # hàm loss
    'learning_rate': [0.1,0.01],    # learning rate
    'n_estimators': [100,200,300],  # số lượng cây
    'criterion' : ['friedman_mse'], # hàm đánh giá
    'max_depth': [3,5,7,9],         # độ sâu của cây con
    'validation_fraction': [0.1]    # phần trăm dữ liệu dùng để validation
}

params_xgb = {
    'learning_rate': [0.01, 0.1],  # learning rate
    'max_depth': [3, 5, 7],        # độ sâu của cây con
    'min_child_weight': [1, 3, 5], # độ sâu nhỏ nhất của cây con
    'subsample': [0.5, 0.7],      # tỉ lệ dữ liệu dùng để train
    'colsample_bytree': [0.5, 0.7], # tỉ lệ features dùng để train
    'n_estimators' : [100, 200, 300,500,1000], # số lượng cây
    'objective': ['reg:squarederror'] # hàm loss
}
```

Huấn luyện Model và vẽ biểu đồ các chỉ số quan trọng ảnh hưởng đến dự đoán

In [50]:
```
def train_model(model,params = None, grid_search = True, scaler = True):
    print(model.__class__.__name__)
    if grid_search:
        grid = GridSearchCV(model,params,cv=5,verbose=1,n_jobs=-1)
        grid.fit(X_train,y_train)
        y_pred = grid.predict(X_test)
    else:
        if scaler:
            x_scaler = StandardScaler()
            y_scaler = StandardScaler()
            X_scale = x_scaler.fit_transform(X_train)
            y_scale = y_scaler.fit_transform(y_train.reshape(-1,1))
            x_test_scale = x_scaler.transform(X_test)
            model.fit(X_scale,y_scale)
            y_pred_scale = model.predict(x_test_scale)
```

```
            y_pred = y_scaler.inverse_transform(y_pred_scale)
        else:
            model.fit(X_train,y_train)
            y_pred = model.predict(X_test)
            # plot feature importance
    try:
        try:
            feature_importances = grid.best_estimator_.feature_importances_
        except:
            feature_importances = model.feature_importances_
        indices = np.argsort(feature_importances)[::-1]

        # plot feature importance and transpose to horizontal
        plt.figure(figsize=(10,5))
        plt.title("Feature Importance")
        plt.bar(range(X_train.shape[1]), feature_importances[indices], align = 'center')
        plt.xticks(range(X_train.shape[1]), process_data.drop('rented_bike_count',axis =
        plt.tight_layout()
        plt.show()
    except:
        pass

    metrics = {
        "Model": model.__class__.__name__,
        'R2 Score': round(r2_score(y_test,y_pred),4),
        'MAE': round(mean_absolute_error(y_test,y_pred),4),
        'MSE': round(mean_squared_error(y_test,y_pred),4),
        'Explained Variance Score': round(explained_variance_score(y_test,y_pred),4)
    }
    return pl.DataFrame(metrics)
```

In [51]:

```
# Linear Regression (Hồi quy tuyến tính)
df_lin = train_model(model = LinearRegression(),grid_search = False)

print(df_lin)
```

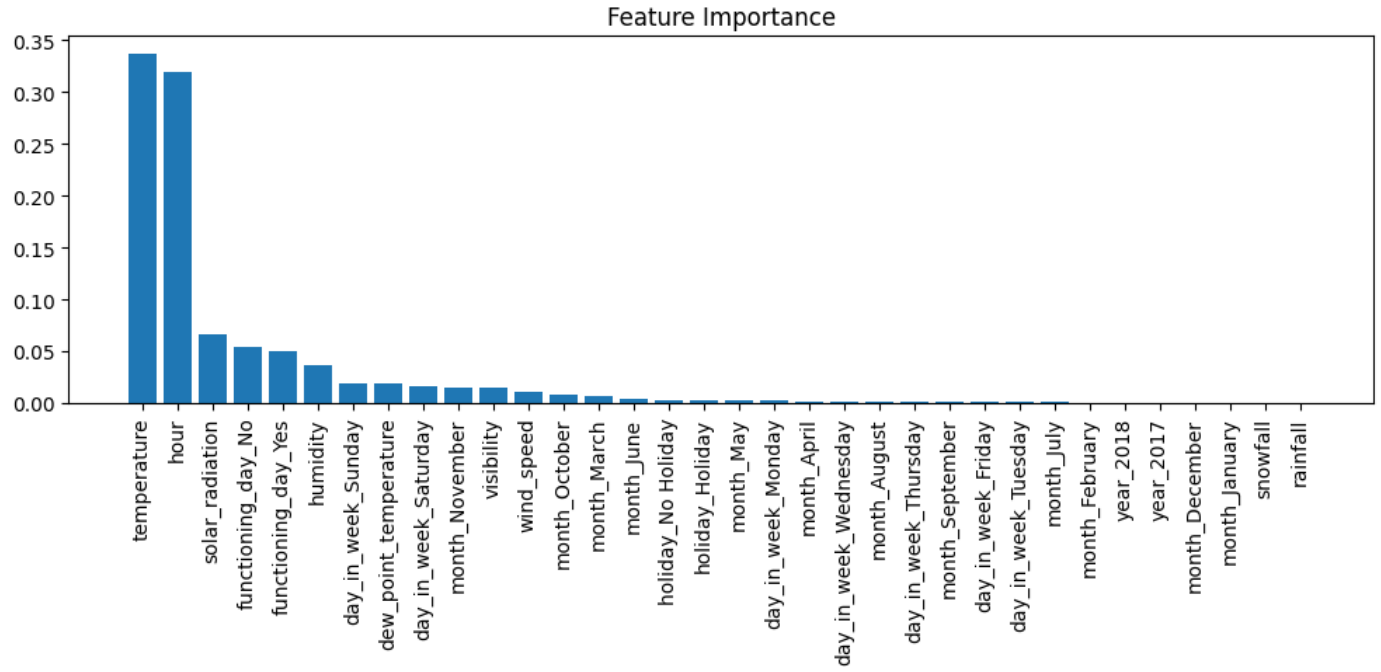
```
LinearRegression
shape: (1, 5)
```

Model	R2 Score	MAE	MSE	Explained Variance Score
str	f64	f64	f64	f64
LinearRegression	0.5925	293.666	147984.4289	0.5925

In [52]:

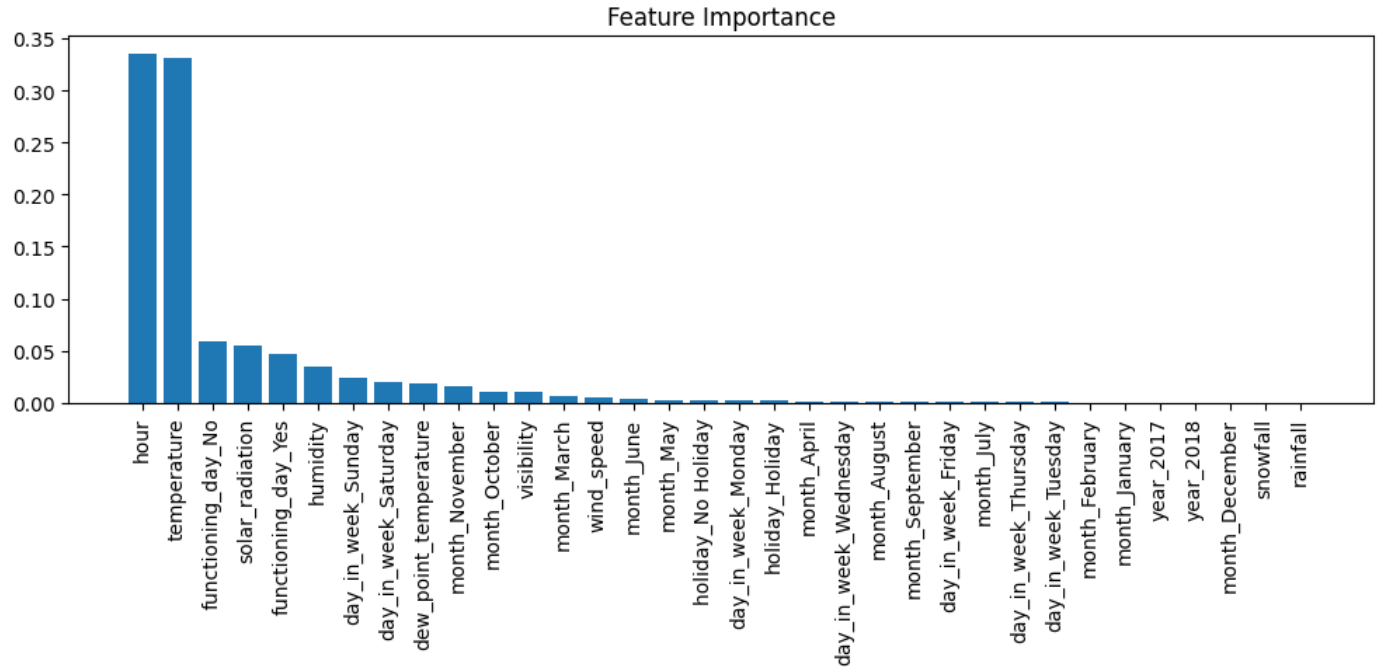
```
df_rfr = train_model(
    model = RandomForestRegressor(n_estimators = 300,criterion = 'friedman_mse'),
    grid_search = False,
    scaler = False)
```

```
RandomForestRegressor
```



In [53]:
```
# Gradient Boosting Regressor (Mô Hình Gradient Boosting)

df_gbr = train_model(model = GradientBoostingRegressor(),params = params_gbr)

print(df_gbr)
```

```
GradientBoostingRegressor
Fitting 5 folds for each of 24 candidates, totalling 120 fits
```

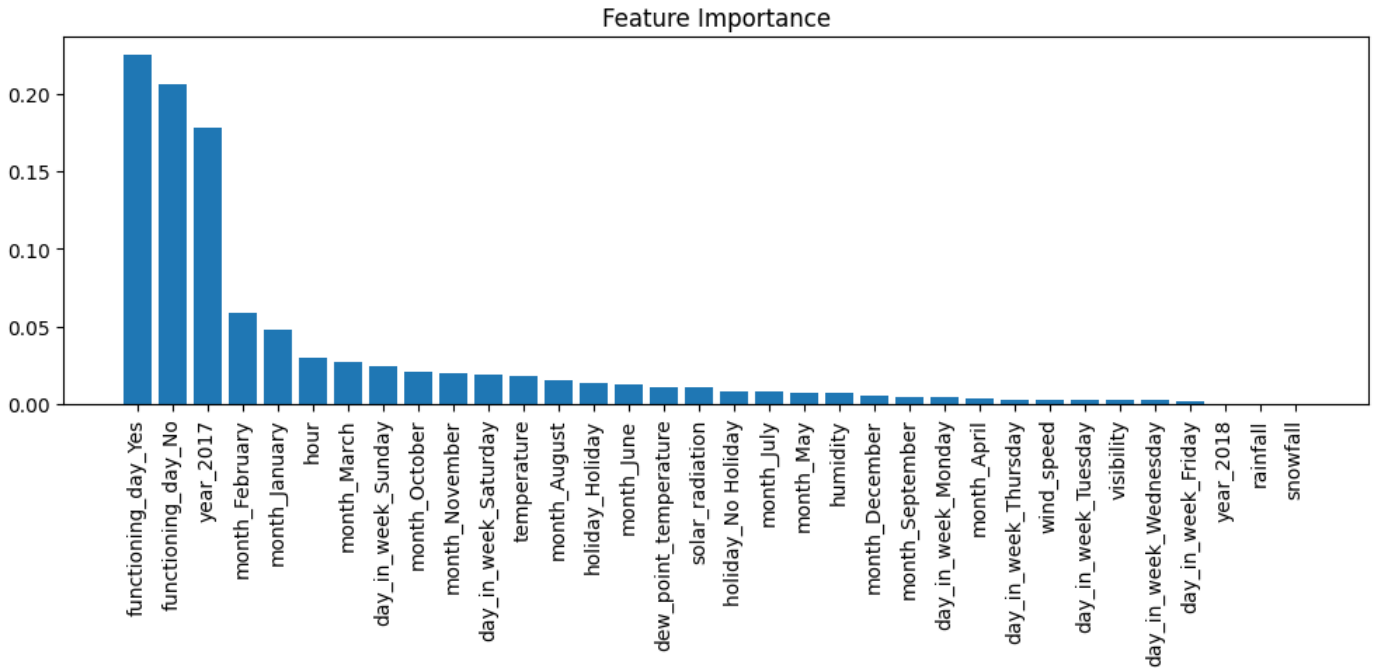


shape: (1, 5)

Model	R2 Score	MAE	MSE	Explained Variance Score
---	---	---	---	---
str	f64	f64	f64	f64
GradientBoostingRegressor	0.9492	82.7082	18461.2822	0.9492

In [54]:
```
# XGBoost Regressor (Mô Hình XGBoost)

df_xgb = train_model(model = XGBRegressor(),params = params_xgb)
print(df_xgb)
```

```
XGBRegressor
Fitting 5 folds for each of 360 candidates, totalling 1800 fits
```



shape: (1, 5)

Model	R2 Score	MAE	MSE	Explained Variance Score
---	---	---	---	---
str	f64	f64	f64	f64
XGBRegressor	0.9518	81.8859	17507.3047	0.9518

5.2. Tinh chỉnh mô hình (Fine-tuning model)

5.3. Đánh giá mô hình (Model evaluation)

In [55]:
```
df = pl.concat([df_lin,df_rfr,df_gbr,df_xgb])
print(df)
```

shape: (4, 5)

Model	R2 Score	MAE	MSE	Explained Variance Score
str	f64	f64	f64	f64
LinearRegression	0.5925	293.666	147984.4289	0.5925
RandomForestRegressor	0.9216	99.4059	28475.4104	0.9216
GradientBoostingRegressor	0.9492	82.7082	18461.2822	0.9492
XGBRegressor	0.9518	81.8859	17507.3047	0.9518

PHẦN VI. KẾT LUẬN VÀ ĐỀ XUẤT (CONCLUSION AND SUGGESTIONS)

1. Kết luận về các nghiên cứu (Conclusion)

Qua việc thống kê mô tả tập dữ liệu “Seoul Bike-sharing Demand” và xây dựng, tinh chỉnh các mô hình, có rất nhiều yếu tố ảnh hưởng đến lượng thuê xe đạp như đã trình bày trong tổng quan nghiên cứu. Cụ thể như sau:

- **Các yếu tố thời tiết**: Nhiệt độ, lượng mưa và lượng tuyết rơi ảnh hưởng trực tiếp đến lượng thuê xe đạp. Cụ thể, số xe đạp thuê nhiều nhất nếu nhiệt độ trung bình ở mức 25 độ C. Trong đó, ảnh hưởng của nhiệt độ là lớn nhất ở 2 trong 4 mô hình. Ngoài ra, đúng với giả thuyết đặt ra, nhóm còn phát hiện thêm yếu tố mới là lượng tuyết rơi tương quan đồng biến với biến phụ thuộc. Mùa trong năm cũng tác động đến số lượng xe đạp, theo đó mọi người có xu hướng thuê xe vào các mùa ấm hơn như mùa hè, giảm dần về các tháng đầu và cuối năm.

- **Ngày nghỉ lễ**: Ngược với nhiều giả thuyết, kết quả đưa ra về tập dữ liệu cho thấy mối tương quan nghịch biến giữa các ngày nghỉ lễ và số xe đạp được thuê. Đây có thể là phát hiện mới của nhóm, nhưng có rất nhiều yếu tố tác động đến lượng thuê xe vào các ngày nghỉ lễ, như quy mô dân số khu vực trạm xe hay mục đích sử dụng *(RB Noland và cộng sự, 2016)*. Do hạn chế về phạm vi, các bài nghiên cứu sau có thể làm rõ hơn về kết quả này. Điều này cũng sẽ giúp nhà hoạch định đưa ra được chính sách tốt hơn về việc phân bổ xe vào các ngày nghỉ lễ tại các trạm tại Seoul.

- **Thời gian thuê xe**: Lượng xe đạp thuê tăng đột biến vào các khung giờ đi làm (8 - 9h) và tan làm (17 - 19h). Có thể rút ra kết luận số lượng người dân tại khu vực thuê xe để phục vụ di chuyển cho công việc. Hơn nữa, người ta nhận thấy rằng trong giờ cao điểm, việc thuê xe đạp có tính cạnh tranh cao hơn so với ô tô về thời gian di chuyển do tắc nghẽn, khiến việc chuyển đổi phương tiện trở nên hấp dẫn hơn *(Florian và cộng sự, 2023)*.

- **Sự an toàn**: Bên cạnh các biến dựa theo các nghiên cứu trước, nhóm tác giả còn phát hiện người đi xe đạp quan tâm đến sự an toàn khi sử dụng. Điều này được chỉ ra bởi chỉ số "Tầm nhìn xa" và "Bức xạ mặt trời". Nếu tầm nhìn có ảnh hưởng sâu sắc đến số lượng xe đạp được thuê thì Bức xạ mặt trời không tương quan quá nhiều.

Ngoài các phát hiện về yếu tố ảnh hưởng đến số lượng thuê xe đạp tại trạm xe ở Seoul, nghiên cứu còn nhiều hạn chế, như việc dữ liệu không bao gồm chi tiết về mục đích sử dụng của từng khách hàng hoặc hoạt động của từng trạm nối. Ngoài dữ liệu lịch sử của "Seoul Bike-Sharing Demand", chỉ bao gồm dữ liệu thời tiết và thời gian hay ngày lễ, thì các yếu tố khác (như giao thông, chính sách của chính phủ,...) không được đưa vào. Các bài nghiên cứu tiếp theo có thể đào sâu hơn vào các chỉ số khác và phân tích sự tác động đến số xe đạp được thuê.

2. Đề xuất giải pháp làm tăng số lượng khách thuê xe (Suggestions)

- **Tăng cường kiểm tra, bảo dưỡng xe đạp định kỳ**: Đặc biệt là vào những ngày thời tiết nắng nóng, cần tăng cường bảo dưỡng để giúp xe đạp hoạt động tốt hơn trong điều kiện thời tiết nắng nóng, tránh xảy ra các sự cố như xe bị hỏng, lốp xe bị xịt,... Hơn nữa, vào các tháng đầu và cuối năm cần tránh để xe đạp lâu ngày ở nơi có ánh nắng trực tiếp, mưa gió, hoặc ẩm ướt.
- **Giảm bớt lượng xe đạp ở trạm vào các tháng đầu và cuối năm**: Bên cạnh việc cân nhắc các yếu tố như lưu lượng xe, nhu cầu dùng, cần giảm bớt lượng xe đạp ở mỗi trạm để giúp tiết kiệm chi phí dịch vụ xe đạp công cộng, đồng thời vẫn đáp ứng được nhu cầu sử dụng của người dân.
- **Tiến hành quảng cáo và giảm giá cước trong ngày lễ**: Nhằm thu hút khách hàng cân nhắc tới dịch vụ xe đạp như là một hoạt động trong ngày lễ. Việc này sẽ giúp thu hút thêm nhiều người sử dụng, đặc biệt là đối với học sinh, sinh viên hay những đối tượng quan tâm tới việc nâng cao sức khỏe.
- **Tăng số lượng xe đạp ở các khu vực cư dân đông đúc**: Điều này sẽ giúp người dùng dễ dàng tìm thấy xe đạp để thuê, đặc biệt là vào những giờ cao điểm, nhiều người sử dụng xe đạp công cộng.
- **Tổ chức các đội ngũ nhân viên hỗ trợ tại các trạm thu/trả xe**: Giải pháp này sẽ thêm hiệu quả khi kết hợp với cung cấp dịch vụ tốt nhằm giúp người dùng thuê xe, trả xe nhanh chóng, thuận tiện.
- **Nâng cao chất lượng dịch vụ**: Chất lượng dịch vụ xe đạp công cộng cần được nâng cao, bao gồm cả chất lượng của xe đạp, chất lượng của hệ thống trạm thu/trả xe, chất lượng của đội ngũ nhân viên vận hành,... Việc này sẽ giúp đảm bảo sự hài lòng của người sử dụng, từ đó khuyến khích họ sử dụng dịch vụ thường xuyên hơn.
- **Trang bị đèn chiếu sáng, áo phản quang cho người dùng**: Việc trang bị các phương tiện này sẽ đảm bảo an toàn khi tham gia giao thông. Các vật dụng này có thể thêm vào các chương trình ưu đãi hoặc tính phí tùy theo chiến lược của nhà cung cấp.
- **Tuyên truyền, giáo dục người dân về ý thức sử dụng và bảo quản xe đạp công cộng**: Việc tuyên truyền, giáo dục người dân sẽ giúp nâng cao ý thức sử dụng và bảo quản xe đạp công cộng, từ đó góp phần kéo dài tuổi thọ của xe.

TÀI LIỆU THAM KHẢO (REFERENCES)

[1] Lin L., He Z., Peeta S. (2018). *Predicting station-level hourly demand in a large-scale bike-sharing n convolutional neural network approach*. https://doi.org/10.1016/j.trc.2018.10.011.

[2] Mateo-Babiano I., Bean R., Corcoran J. and Pojani D. (2016).*How does our natural and built environment affect the use of bicycle sharing?*, Transportation Research Part A: Policy and Practice, 94, pp.295-307.

[3] Wang W. (2016). *Forecasting Bike Rental Demand Using New York Citi Bike Data*. https://arrow.tudublin.ie/cgi/viewcontent.cgi?article=1083&context=scschcomdis.

[4] Wilkesmann F., Ton D., Schakenbos R. and Cats O. (2023). *Determinants of station-based round-trip bikesharing demand*, Journal of Public Transportation, 25,p.100048. https://www.sciencedirect.com/science/article/pii/S1077291X23000097.

[5] Wu X., Lyu C., Wang Z., Liu Z. (2019). *Station-Level Hourly Bike Demand Prediction for Dynamic Repositioning in Bike Sharing Systems*. https://doi.org/10.1007/978-981-13-8683-1_3.

[6] Xu X., Ye Z., Li J., Xu M. (2018). *Understanding the Usage Patterns of Bicycle-Sharing Systems to Predict Users' Demand: A Case Study in Wenzhou, China*. https://doi.org/10.1155/2018/9892134.

[7] Yang Z. (2016). *Mobility Modeling and Prediction in Bike-Sharing Systems*. https://doi.org/10.1145/2906388.2906408.

[8] Ye X.V. and Bai J.J. (2021). *Predicting Daily Rental Counts for Bike-Sharing Programs*. https://causeweb.org/usproc/sites/default/files/usclap/2021-2/Predicting%20Daily%20Rental%20Counts%20For%20Bike-Sharing%20Programs.pdf.