Term Project

김세중 노현지 양다인 오진영

< End-to-End Process >

1.Business objective :

To measure the risk of traffic accidents and enable people to be aware of the accident probability before going outside. This contributes to public safety. By avoiding areas and times with a high probability of traffic accidents, or being aware in advance that there may be a risk of accidents, people can take preventive measures and be cautious.

2. Data exploration (including dataset description):

* Accident

114442 records x 16 columns

Accident.csv: A dataset that includes the date, location, and the number of casualties and injured individuals in Seoul.

Sites are identified by 'Gu' and 'Dong' names such as Songpa-gu and Garak-dong.

텍스트, 폰트, 스크린샷, 시계이(가) 표시된 사진

자동 생성된 설명

스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명

* Population

450 records x 14 columns

Population.csv: Contains information about the population residing in each 'Gu' and 'Dong' during each quarter of the years 2017, 2018 and2019.

By utilizing accident.csv and population.csv, our team analyzes the accident risk in each 'Dong' and 'Gu' (district).

텍스트, 폰트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명

'region\_mapping' and 'Population\_region\_name\_change' are used to establish a connection between the accident and population tables.

The accident table uses the legal names of 'Dong', while the population table uses the administrative names of 'Dong'. The administrative names have a different format from the legal names and provide more specific classification of the 'Dong' areas. For example, the legal name 'Garak-dong' is further classified into 'Garak 1-dong', 'Garak 2-dong', and 'Garakbon-dong' in the administrative naming system.

* Region\_mapping

765 records x 7 columns

Region\_mapping includes both the legal and administrative names of 'Dong', serving as a link between the two tables.

텍스트, 폰트, 스크린샷, 시계이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명

However, there is a small issue. Region\_mapping uses the administrative names of 'Dong', but its format differs from the administrative naming used in the population table. For instance, the population table uses 'Changsin1-dong', whereas region\_mapping uses 'Changsin Je 1-dong'.

* Population\_region\_name\_change

180 records x 2 columns

The Population\_region\_name\_change handles this difference in administrative naming formats to establish a connection between the region\_mapping and population tables.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**3. Data preprocessing source code**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

region\_population\_mapping = dict()

region\_do\_population\_mapping = dict()

# def

# Read csv files

# cat accident data

accident\_df = pd.read\_csv("accident.csv", sep=",", encoding="cp949")

# population accident data

population\_df = pd.read\_csv("population.csv", sep=",", encoding\_errors="ignore")

# car accident data

population\_region\_name\_change\_df = pd.read\_csv("population\_region\_name\_change.csv", sep=",", encoding="cp949")

# Region mapping data

region\_mapping\_df = pd.read\_csv("region\_mapping.csv", sep=",", encoding="cp949")

# Make ['발생일'] type to datetime

accident\_df['발생일'] = pd.to\_datetime(accident\_df['발생일'])

accident\_df['사고건수'] = pd.to\_numeric(accident\_df['사고건수'])

accident\_df['사망자수'] = pd.to\_numeric(accident\_df['사망자수'])

accident\_df['중상자수'] = pd.to\_numeric(accident\_df['중상자수'])

accident\_df['경상자수'] = pd.to\_numeric(accident\_df['경상자수'])

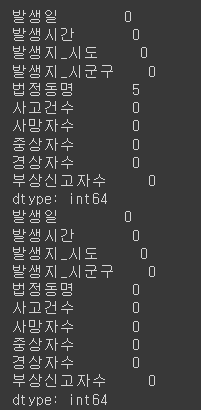
accident\_df['부상신고자수'] = pd.to\_numeric(accident\_df['부상신고자수'])

population\_df.head(10)

스크린샷이(가) 표시된 사진

자동 생성된 설명

# Check dirtydata



# Check dirty data

print(accident\_df.isnull().sum())

# Cleaning dirty data

accident\_df.dropna(axis=0, how='any', inplace=True)

accident\_df.isnull().sum()

As a result of checking the dirty data of accidents\_df, five null data were found in the columns of [“법정 동명”]. We decided to drop the data because it was a small portion of the total data of about 120,000.

# Set each column type of accident\_df

# Feature creation (derive New Features from existing ones)

accident\_df['발생\_연도'] = accident\_df['발생일'].dt.year

accident\_df['발생\_월'] = accident\_df['발생일'].dt.month

accident\_df['발생\_일'] = accident\_df['발생일'].dt.day

accident\_df['발생\_요일'] = accident\_df['발생일'].dt.day\_name()

accident\_df

스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명

The ['발생일'] consists of time series data. We created new columns of accident\_df [‘발생\_연도’, ‘발생\_월’, ‘발생\_일’, ‘발생\_요일’] to encode data of [‘발생일’].

def population\_region\_name\_fit():

  for i in range(len(population\_df)):

      change\_population\_region\_name = population\_region\_name\_change\_df.loc[population\_region\_name\_change\_df['population\_region'] == population\_df.iloc[i]['행정동명']]['mapping\_region'].values

      if len(change\_population\_region\_name) > 0:

          population\_df.iloc[i]['행정동명'] = change\_population\_region\_name[0]

population\_region\_name\_fit()

population\_df

The purpose of the provided function population\_region\_name\_fit() is to align the administrative district names in a population dataset with a CSV file containing mappings between legal district names and administrative district names.

# The part that adds the number of people by accident

# Function that returns the population of the date in the area when you insert the accident area and the date

def get\_region\_population(accident\_region\_name, accident\_year, accident\_month):

    quarter = int((accident\_month - 1) / 3 + 1)

    accident\_quarter = str(accident\_year)+ ' ' + str(quarter) + '/4'

    # population capacity variable

    popluation = 0

    population\_regions = []

    accident\_popul\_key = accident\_region\_name + accident\_quarter

    # accident region

    if accident\_popul\_key not in region\_population\_mapping:

        # Select population dong (administrative dong)from the accident dong (legal name)

        population\_regions = region\_mapping\_df.loc[region\_mapping\_df['동리명'] == accident\_region\_name]['읍면동명'].values

        for region in population\_regions:

            # if len(population\_df.loc[population\_df['행정동명'] == region][accident\_quarter]) > 0:

            if len(population\_df.loc[population\_df['행정동명'] == region][accident\_quarter]) > 0:

                popluation += int(population\_df.loc[population\_df['행정동명'] == region][accident\_quarter].values[0])

        region\_population\_mapping[accident\_popul\_key] = popluation

    else:

        popluation = region\_population\_mapping[accident\_popul\_key]

    # region\_mapping\_df +=

    return popluation

# A function that returns the population of the date in the accident area (degree) and date

def get\_gu\_region\_population(accident\_do\_name, accident\_year, accident\_month):

    quarter = int((accident\_month - 1) / 3 + 1)

    accident\_quarter = str(accident\_year)+ ' ' + str(quarter) + '/4'

    accident\_popul\_key = accident\_do\_name + accident\_quarter

    # accident region

    if accident\_popul\_key not in region\_do\_population\_mapping:

        popluation = int(population\_df.loc[(population\_df['구'] == accident\_do\_name) & (population\_df['행정동명'] == '소계')][accident\_quarter].values[0])

        region\_do\_population\_mapping[accident\_popul\_key] = popluation

    else:

        popluation = region\_do\_population\_mapping[accident\_popul\_key]

    return popluation

# Add population people by accident ‘법정동명’

for i in range(len(accident\_df)):

    accident\_df.iat[i, 14] = get\_region\_population(accident\_df.iat[i, 4], accident\_df.iat[i, 10], accident\_df.iat[i, 11])

    accident\_df.iat[i, 15] = get\_gu\_region\_population(accident\_df.iat[i,3], accident\_df.iat[i, 10], accident\_df.iat[i, 11])

accident\_df

스크린샷, 키보드이(가) 표시된 사진

자동 생성된 설명

`get\_region\_population(accident\_region\_name, accident\_year, accident\_month)` Function that returns the population of the date in the area when you insert the accident area and the date..

1. This function takes an accident region name, accident year, and accident month as input.
2. It calculates the quarter based on the provided accident month.

`get\_gu\_region\_population(accident\_do\_name, accident\_year, accident\_month)` Function that returns the population of the date in the accident area (degree) and date

1. This function takes an accident region (province) name, accident year, and accident month as input.
2. It calculates the quarter based on the provided accident month.

# Add [‘위험도’] columns each accident

accident\_df['위험도'] = np.nan

# Create deatset to run modeling

for i in range(len(accident\_df)):

    accident\_score = ((accident\_df.iat[i, 6] \* 10 + accident\_df.iat[i, 7] \* 5 + accident\_df.iat[i, 8] \* 3 + accident\_df.iat[i, 9] \* 1) \* 10000) / accident\_df.iat[i, 14]

    accident\_df.iat[i, 16] = accident\_score

accident\_df

스크린샷이(가) 표시된 사진

자동 생성된 설명

A new [‘위험도’] column was created by applying different weights of [‘사고건수’, ‘사망자수’, ‘중상자수’, ‘경상자수’] because target barrier is required to proceed with decision tree modeling. This will later be grouped using clustering and then changed to categorical data.

**4. Clustering Algorithm**

# Clustering Without Outlier

# Change 2D-data

X = np.array(accident\_df['위험도'].values).reshape(-1, 1)

cluster\_num = 4

from sklearn.cluster import KMeans

# Create KMeans objects and run clustering

means = KMeans(n\_clusters=cluster\_num, max\_iter = 150)

# Set Clustering num(K)

kmeans.fit(X)

# Checking clustering results

labels = kmeans.labels\_

centroids = kmeans.cluster\_centers\_

# Prepare colors for different clusters

colors = ['blue', 'indigo', 'lawngreen', 'darkorange', 'm', 'orange']

unique, count = np.unique(labels, return\_counts = True)

uniq\_cnt\_zip = dict(zip(unique + 1, count))

centroids =  centroids.flatten().tolist()

centroids\_sorted = sorted(centroids)

risks = ['D', 'C', 'B', 'A']

risk\_index=[]

for centroid in centroids:

    risk\_index.append(risks[(centroids\_sorted.index(centroid))])

risk\_cent\_zip = dict(zip(risk\_index, count))

risk\_list = list(risk\_cent\_zip.keys())

print(risk\_list)

cluster\_bound = [(centroids\_sorted[0] + centroids\_sorted[1])/ 2, (centroids\_sorted[1] + centroids\_sorted[2])/ 2, (centroids\_sorted[2] + centroids\_sorted[3])/ 2]

for cluster in range(cluster\_num):

    # find the distance from the center of the cluster to each point

    cluster\_points = X[labels == cluster]

    print('Cluster {} : {}'.format(str(cluster +1), cluster\_points.size))

    plt.scatter(cluster\_points, np.zeros\_like(cluster\_points), c=colors[cluster], s=20, label='Risk : ' + str(risk\_index[cluster]))

# plt.scatter(centroids, np.zeros\_like(centroids),c='red',s=20)    # display centroid data in red

plt.title('K-means clustering Result')

plt.xlabel('accident risk')

print("클러스터 레이블:")

print(labels)

print("클러스터 중심:")

print(centroids)

print(risk\_cent\_zip)

plt.legend()

plt.show()

# Show the number of grades of A,B,C,D as a graph

# Sort the value of risk\_cent\_zip by key

sorted\_risk\_cent\_zip = sorted(risk\_cent\_zip.items(), key=lambda x: x[0], reverse=True)

# Split into a list of keys and values

labels, values = zip(\*sorted\_risk\_cent\_zip)

plt.bar(labels, values, color = colors)

for i, v in enumerate(values):

    plt.text(i, v, str(v), ha='center')

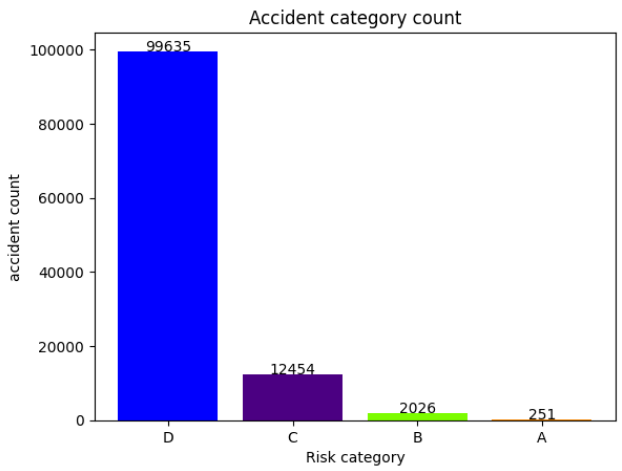
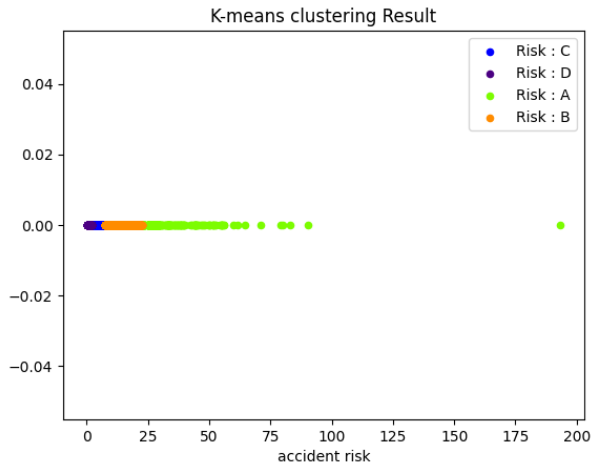
plt.title('Accident category count')

plt.xlabel('Risk category')

plt.ylabel('accident count')

# Plot gragh

plt.show()

****

Clustering without removing the outliner results in a smaller range of clustering and no clear boundary points for the population. If you look at the number of each hazard in the bar graph, you can see that it consists of an unbalanced dataset. It is necessary to process the outliner in order to configure the clustering model.

# Outliered clustering

# Noise data

quantile\_25 = np.quantile(accident\_df['위험도'].values, 0.25)

quantile\_75 = np.quantile(accident\_df['위험도'].values, 0.75)

IQR = quantile\_75 - quantile\_25

maximum = quantile\_75 + 2.0 \* IQR

noise\_data = [] # store noise\_data in this list

noise\_del\_data = [] # store data without noise(outlier)

# check noise using IQR maximum

for risk in accident\_df['위험도'].values:

    if risk > maximum : # if outlier cotain that point in noise\_data

        noise\_data.append(risk)

    else:   # else cotain that point in noise\_del\_data

        noise\_del\_data.append(risk)

# Change 2D-dataset

X = np.array(noise\_del\_data).reshape(-1, 1)

cluster\_num = 4

from sklearn.cluster import KMeans

# Create KMeans objects and run clustering

kmeans = KMeans(n\_clusters=cluster\_num, max\_iter = 150)  # Set Clustering num(K)

kmeans.fit(X)

# Checking clustering results

labels = kmeans.labels\_

centroids = kmeans.cluster\_centers\_

# display cluster data with out noise

# Prepare colors for different clusters

colors = ['blue', 'indigo', 'lawngreen', 'darkorange', 'm', 'orange']

unique, count = np.unique(labels, return\_counts = True)

uniq\_cnt\_zip = dict(zip(unique + 1, count))

centroids =  centroids.flatten().tolist()

centroids\_sorted = sorted(centroids)

risks = ['D', 'C', 'B', 'A']

risk\_index=[]

for centroid in centroids:

    risk\_index.append(risks[(centroids\_sorted.index(centroid))])

risk\_cent\_zip = dict(zip(risk\_index, count))

# Set larger outlier data to ‘A’

risk\_cent\_zip['A'] += len(noise\_data)

cluster\_bound = [(centroids\_sorted[0] + centroids\_sorted[1])/ 2, (centroids\_sorted[1] + centroids\_sorted[2])/ 2, (centroids\_sorted[2] + centroids\_sorted[3])/ 2]

for cluster in range(cluster\_num):

    # find the distance from the center of the cluster to each point

    cluster\_points = X[labels == cluster]

    print('Cluster {} : {}'.format(str(cluster +1), cluster\_points.size))

    plt.scatter(cluster\_points, np.zeros\_like(cluster\_points), c=colors[cluster], s=20, label='Risk : ' + str(risk\_index[cluster]))

plt.scatter(centroids, np.zeros\_like(centroids),c='red',s=20)    # display centroid data in red

plt.title('K-means clustering Result')

plt.xlabel('accident risk')

print("클러스터 레이블:")

print(labels)

print("클러스터 중심:")

print(centroids)

print(risk\_cent\_zip)

plt.legend()

plt.show()

# Show the number of grades of A,B,C,D as a graph

# Sort the value of risk\_cent\_zip by key

sorted\_risk\_cent\_zip = sorted(risk\_cent\_zip.items(), key=lambda x: x[0], reverse=True)

# Split into a list of keys and values

labels, values = zip(\*sorted\_risk\_cent\_zip)

plt.bar(labels, values, color = colors)

for i, v in enumerate(values):

    plt.text(i, v, str(v), ha='center')

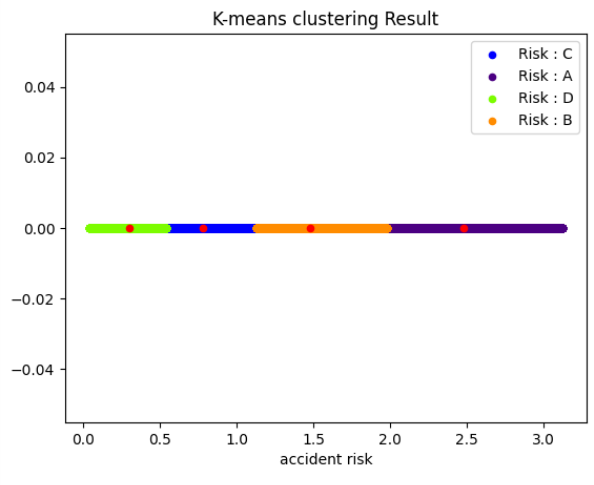
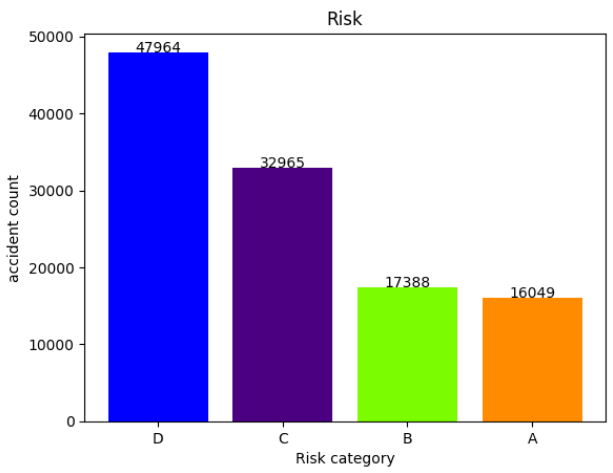
plt.title('Accident category count')

plt.xlabel('Risk category')

plt.ylabel('accident count')

# Plot gragh

plt.show()



The risk of an outliner that is too large was designated as 'A' and the outliner was removed. If you look at the clustering results, you can see that the distribution of the population is evenly spread compared to the previous model. In addition, it was confirmed that the number of A, B, C, and D grades of the bar graph was also balanced. The handling of the outliers in clustering is essential, and failure to do so can have the worst impact on the modeling results.

**5. DecisionTreeClassifier**

# Convert target barrier to categorical data based on clustering results

for i in range(len(accident\_df)):

    if  accident\_df.iat[i, 16] < cluster\_bound[0]:

        accident\_df.iat[i, 16] = 'D'

    elif accident\_df.iat[i, 16] < cluster\_bound[1]:

        accident\_df.iat[i, 16] = 'C'

    elif accident\_df.iat[i, 16] < cluster\_bound[2]:

        accident\_df.iat[i, 16] = 'B'

    else :

        accident\_df.iat[i, 16] = 'A'

# Create decisiontreeClassifier dataset

# Create decision\_dataset

decision\_df = accident\_df.loc[:, ['법정동명', '발생시간', '발생\_월', '발생\_요일', '위험도']]

decision\_df.head()

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

The decision tree is classification and must have a categorical target barrier. To this end, the risk was converted to A, B, C, and D based on the clutter group that was conducted earlier. A has the highest risk rating, and the lower the risk goes.

from sklearn.preprocessing import LabelEncoder

# Convert numberic data to Categorical data

region\_label = LabelEncoder()

day\_label = LabelEncoder()

region\_label.fit(decision\_df['법정동명'])

day\_label.fit(decision\_df['발생\_요일'])

decision\_df['법정동명'] = region\_label.transform(decision\_df['법정동명'])

decision\_df['발생\_요일'] = day\_label.transform(decision\_df['발생\_요일'])

decision\_df = decision\_df.astype('category')

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

For modeling learning, we conducted encoder labeling that transforms categorical data into continuous data. Since there are two columns to encode, we separated label\_fit into two to apply the predicted data set in the future and stored the learned data separately.

# Find best param by RandomForest

from sklearn.model\_selection import GridSearchCV

from sklearn.ensemble import RandomForestClassifier

X = decision\_df.iloc[:,:4]  # feature

y = decision\_df.iloc[:,-1]  # Target variable

params = {

    'n\_estimators': [10],

    'criterion': ['gini', 'entropy'],

    'bootstrap': [True, False]

}

# search best param

rf\_clf = RandomForestClassifier(random\_state = 0, n\_jobs = -1)

grid\_cv = GridSearchCV(rf\_clf, param\_grid = params, cv = 5, n\_jobs = -1)

grid\_cv.fit(X, y)

print('best hyper parameter: ', grid\_cv.best\_params\_)

print('best score: {:.4f}'.format(grid\_cv.best\_score\_))



from sklearn.model\_selection import KFold

from sklearn.model\_selection import cross\_val\_score

random\_model = RandomForestClassifier(n\_estimators=10, bootstrap = True, criterion = 'gini', random\_state=0)

random\_model.fit(X, y)

kfold = KFold(n\_splits=10, shuffle = True, random\_state=0)

# Evaluate base model using cross-validation

base\_scores = cross\_val\_score(random\_model, X, y, cv=kfold)

base\_avg\_score = np.mean(base\_scores)

print("\n========== Base Model Cross-Validation Scores ==========\n", base\_scores)

print("\n========== Base Model Average Score ==========\n", base\_avg\_score)

print("\n========== Base Model Cross-Validation best Score ==========\n", base\_scores.max())

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import cross\_val\_score

X = decision\_df.iloc[:,:4]  # feature

y = decision\_df.iloc[:,-1]  # Target variable

# Base DesicionTree model(depth = 14)

base\_model = DecisionTreeClassifier(max\_depth = 14, min\_samples\_split = 7)

base\_model.fit(X, y)

kfold = KFold(n\_splits=10, shuffle = True, random\_state=0)

# Evaluate base model using cross-validation

base\_scores = cross\_val\_score(base\_model, X, y, cv=kfold)

base\_avg\_score = np.mean(base\_scores)

print("\n========== Base Model Cross-Validation Scores ==========\n", base\_scores)

print("\n========== Base Model Average Score ==========\n", base\_avg\_score)

print("\n========== Base Model Cross-Validation best Score ==========\n", base\_scores.max())

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Found base param of Random Forest Classifier using GridSearchCV. Best para is {'bootstrap': True, 'criterion': 'entropy', 'n\_estimators': 10}. However, the best score reached only 0.56. So we conducted additional testing through K-fold cross validation. The result is a 0.1 improvement, but I'm still not happy with the results.

We turned the decision tree around, putting aside random forest. After conducting the k-fold cv with the decision tree max\_depth as 14, I confirmed the best score of 0.65. This shows a significantly higher score than the random forest classifier.

We judged that it would be better to use the max\_depth=14 modeling of the deciontreeclassifier.

def make\_test\_df(region, month, time, day) :

    # test\_df = pd.Dataframe(columns = ['법정동명', '발생시간', '발생\_월', '발생\_요일'])

    test\_df = {'법정동명': region,

                '발생시간': time,

                '발생\_월': month,

                '발생\_요일': day}

    data = pd.DataFrame(test\_df, index = [0])

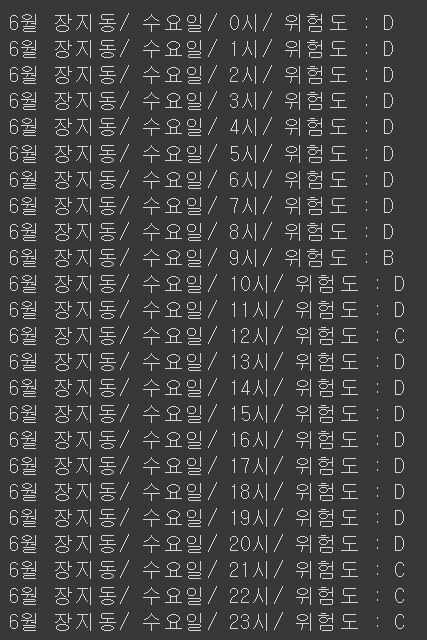
    data['법정동명'] = region\_label.transform(data['법정동명'])

    data['발생\_요일'] = day\_label.transform(data['발생\_요일'])

    return data

for i in range(24):

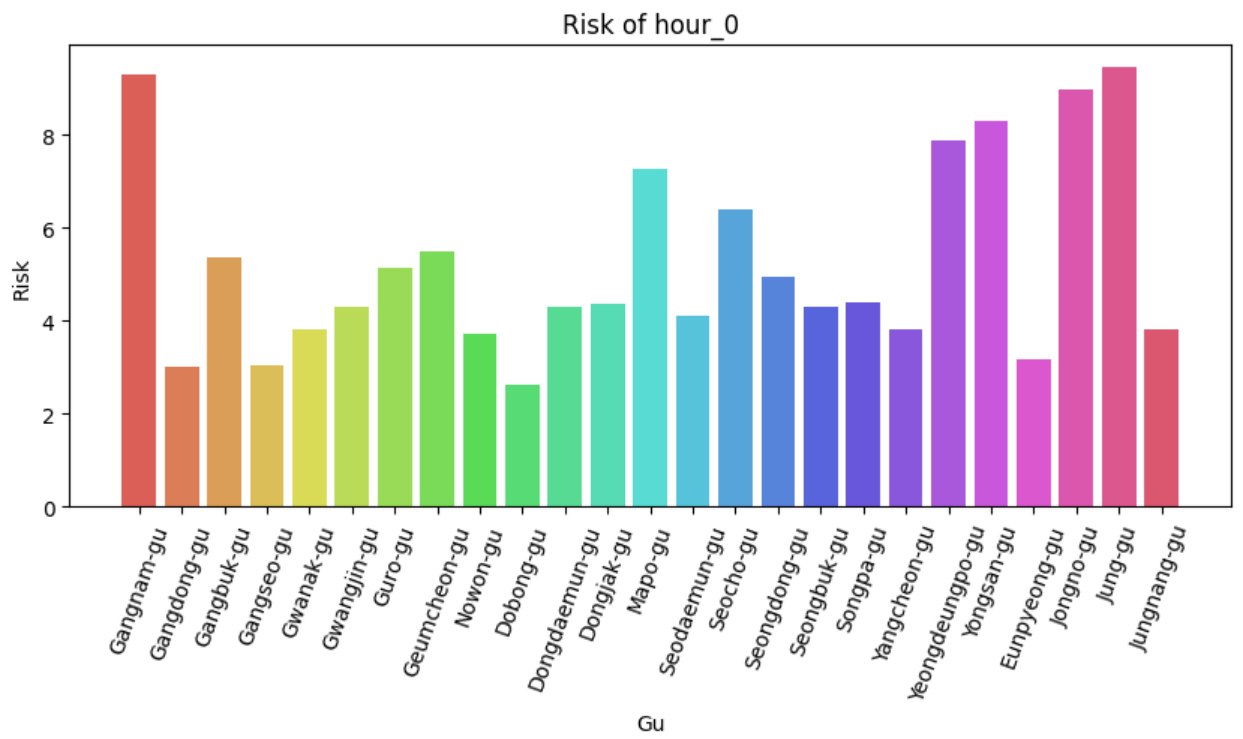
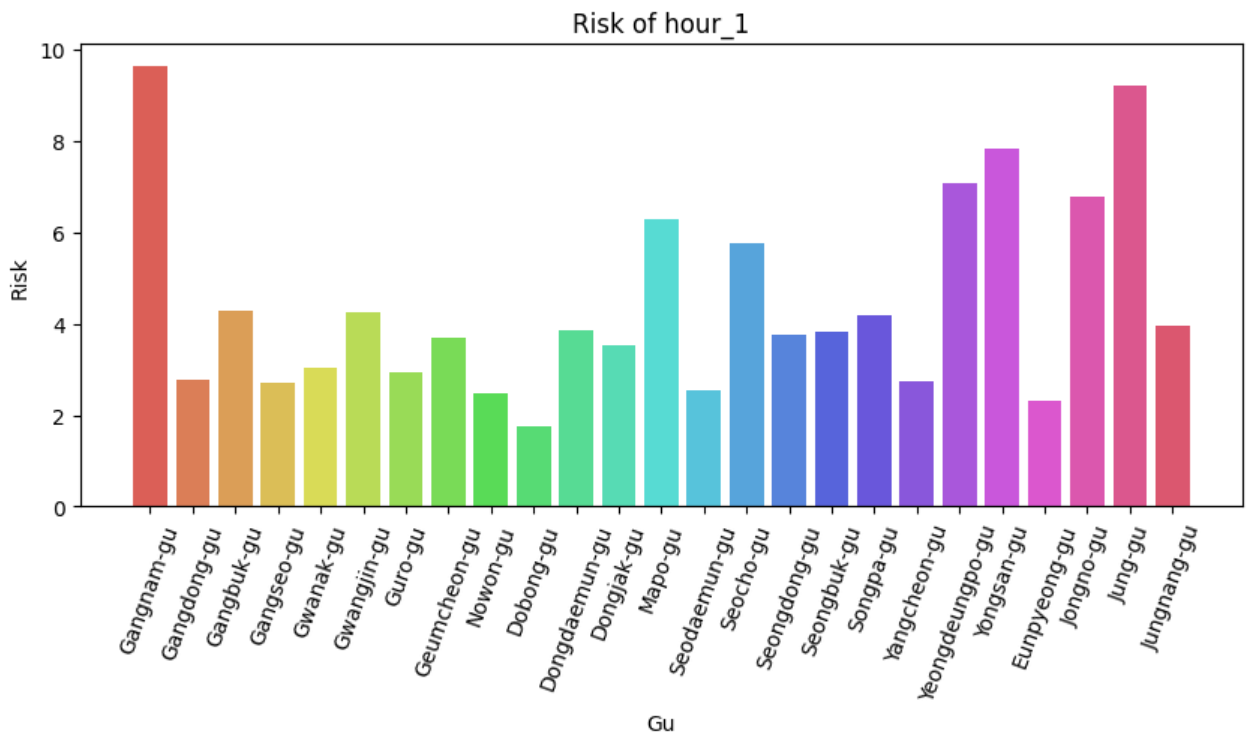
    print('6월 장지동/ 수요일/ {}시/ 위험도 : {}'.format(i ,base\_model.predict(make\_test\_df('장지동', 6, i, 'Wednesday'))[0]))

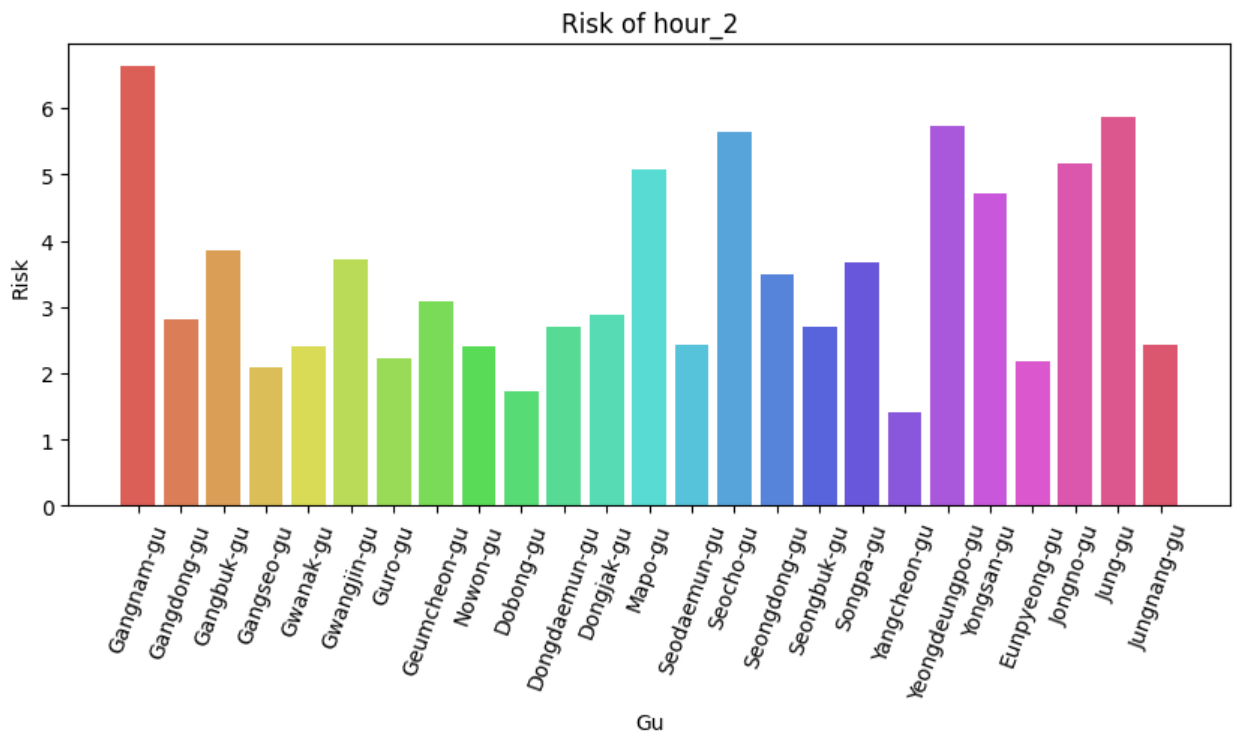
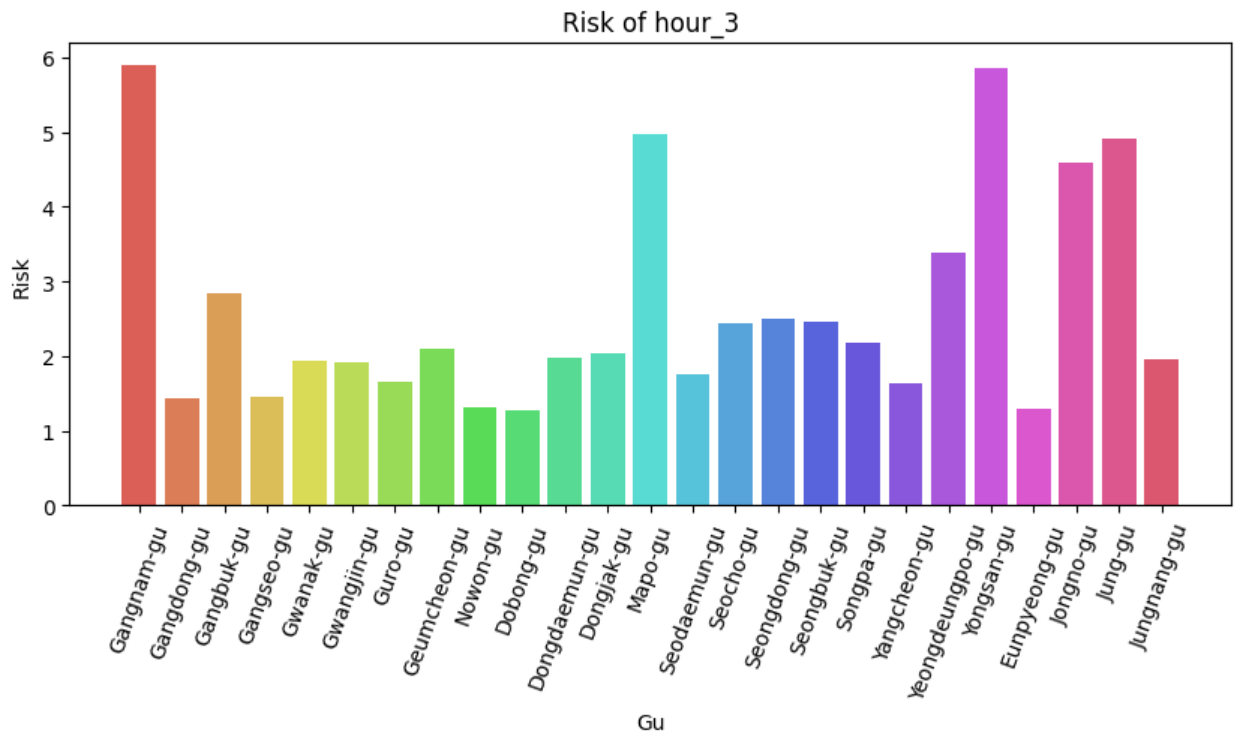
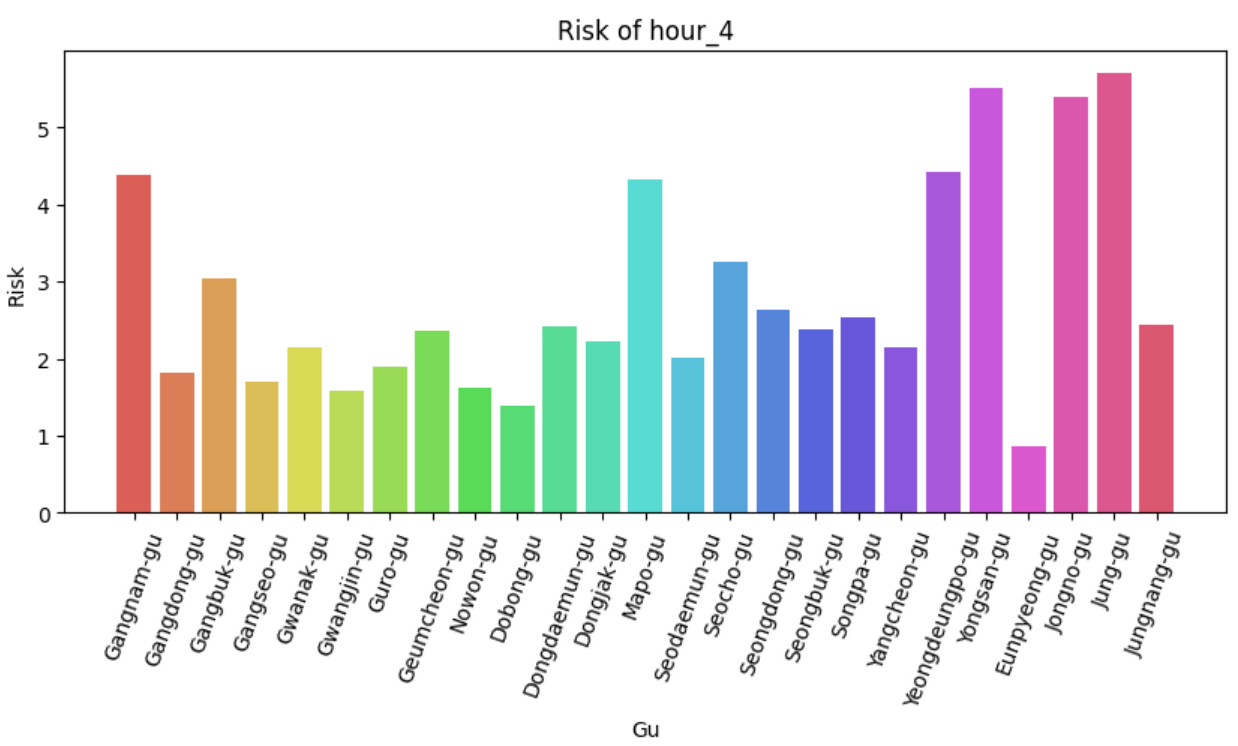
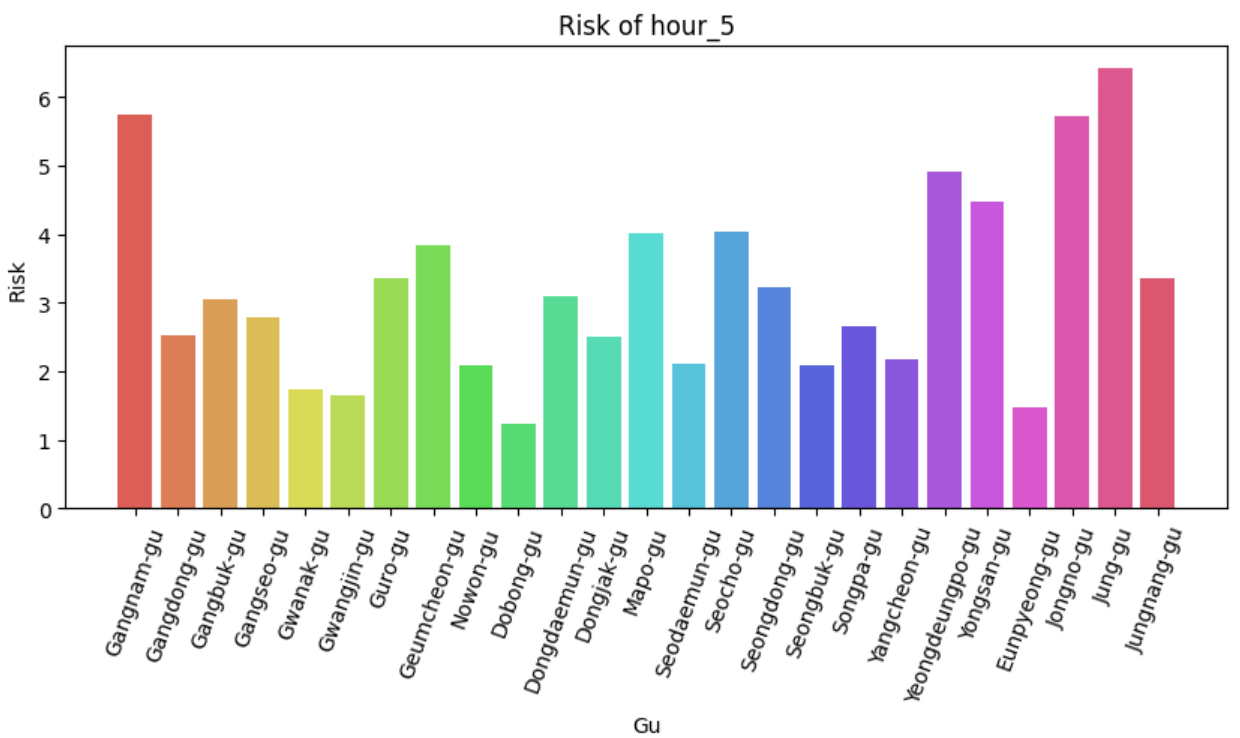
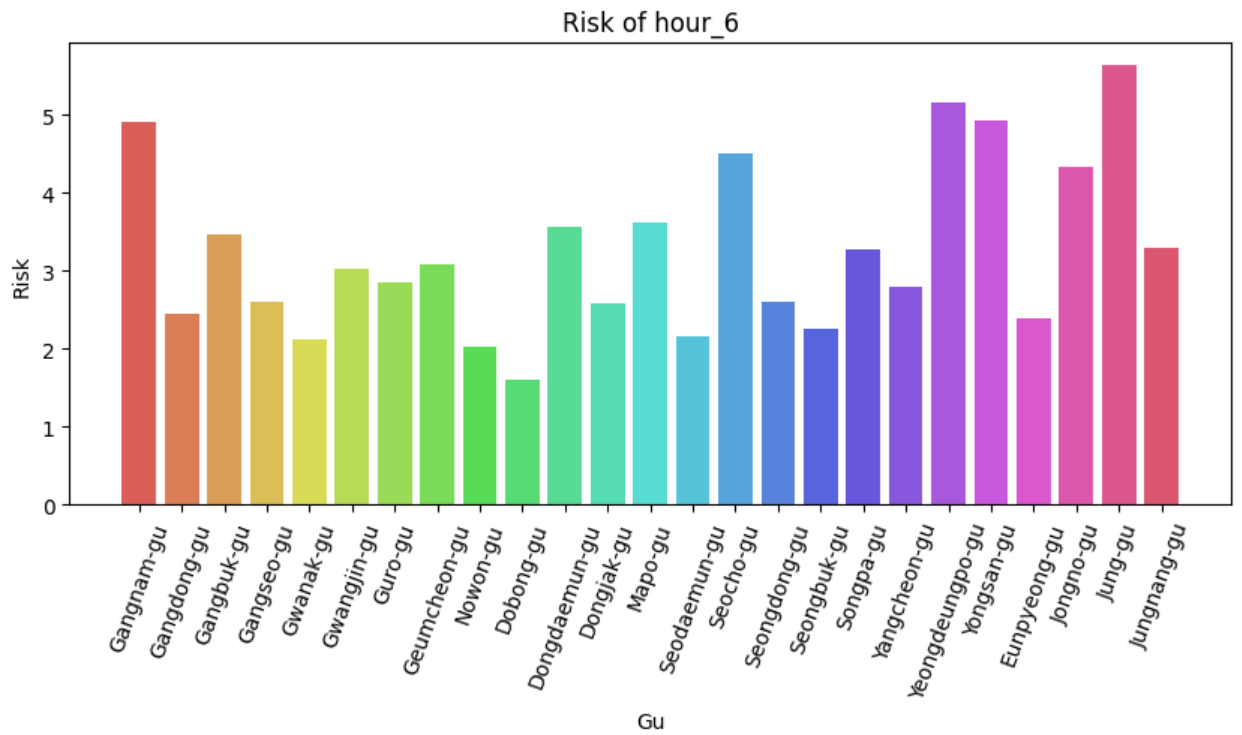
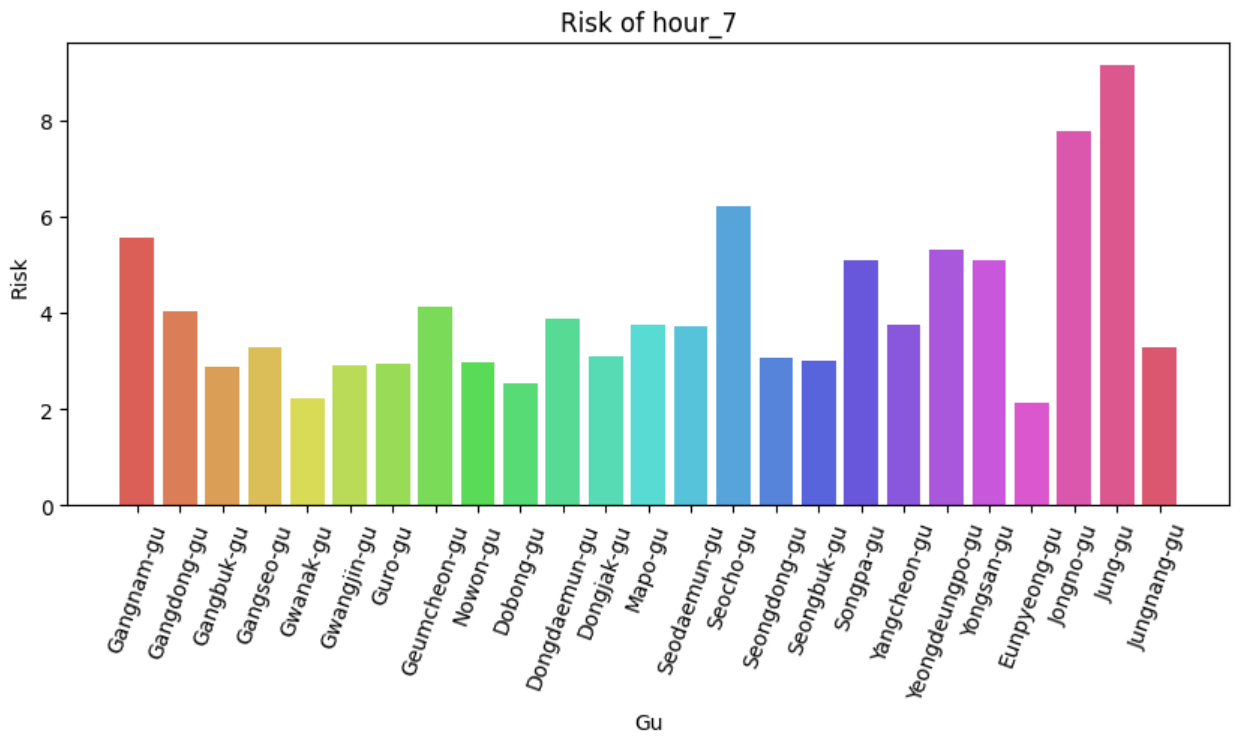
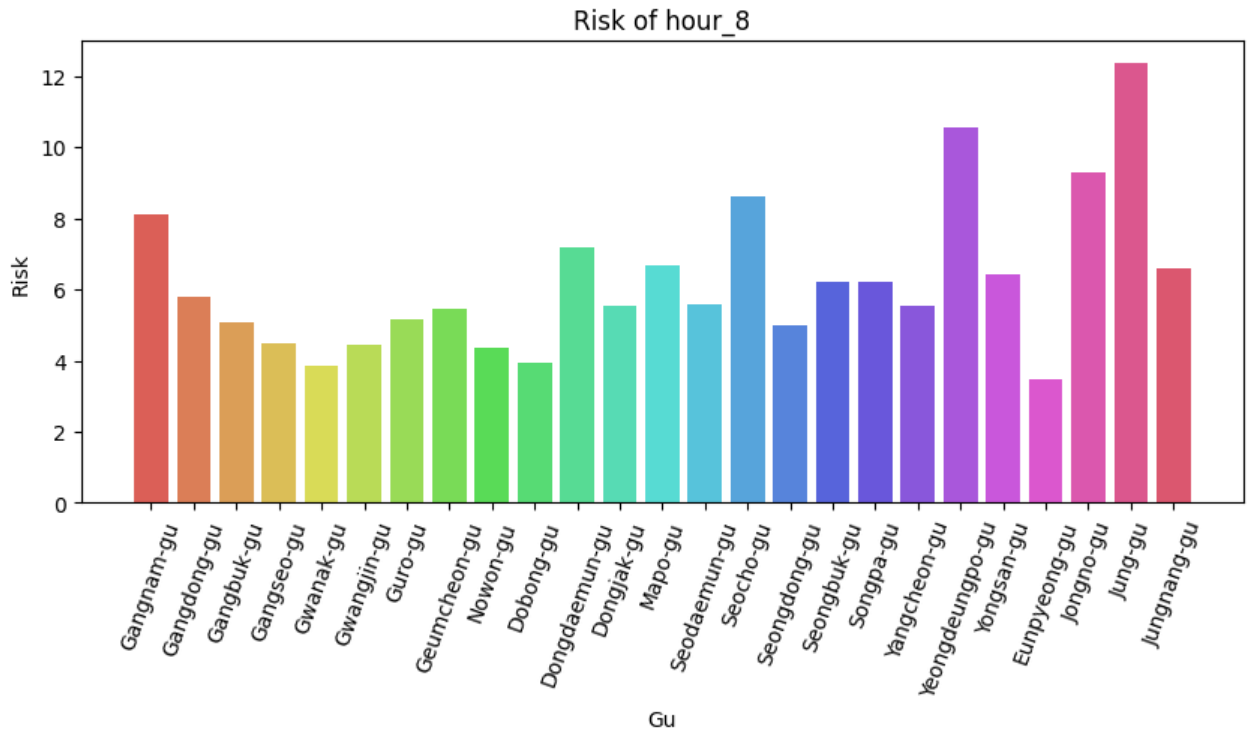
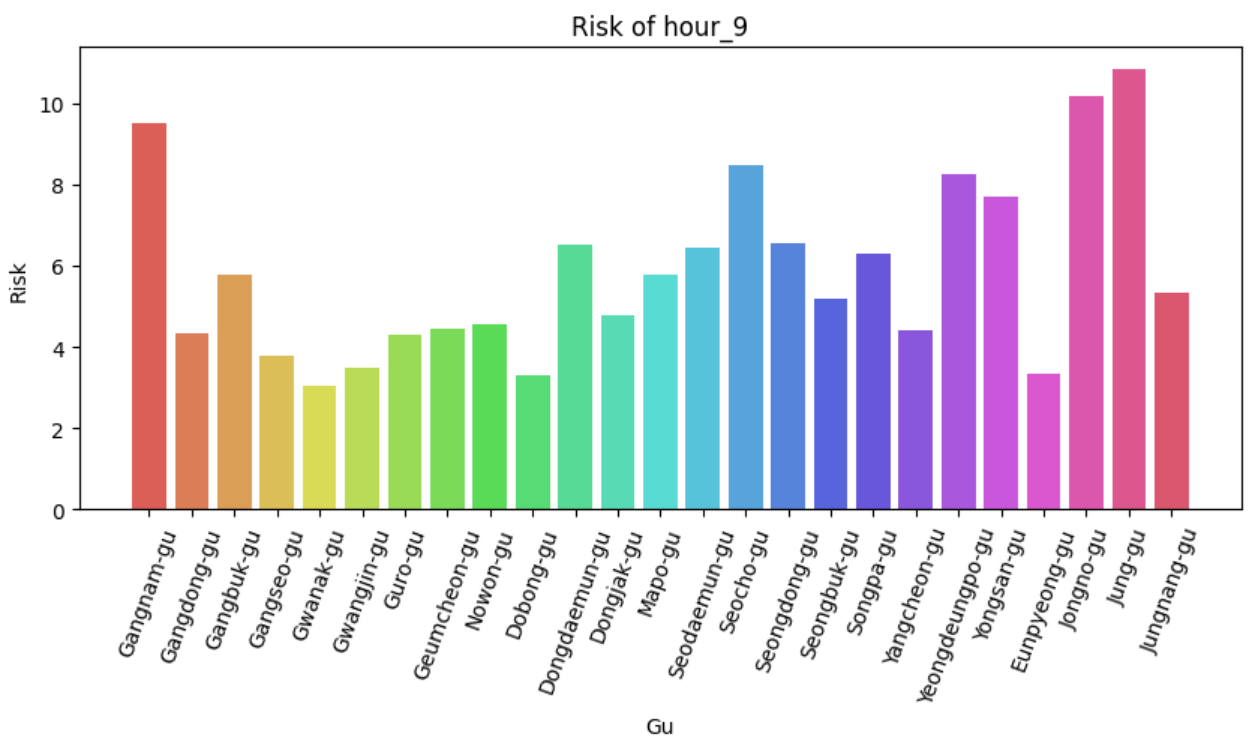
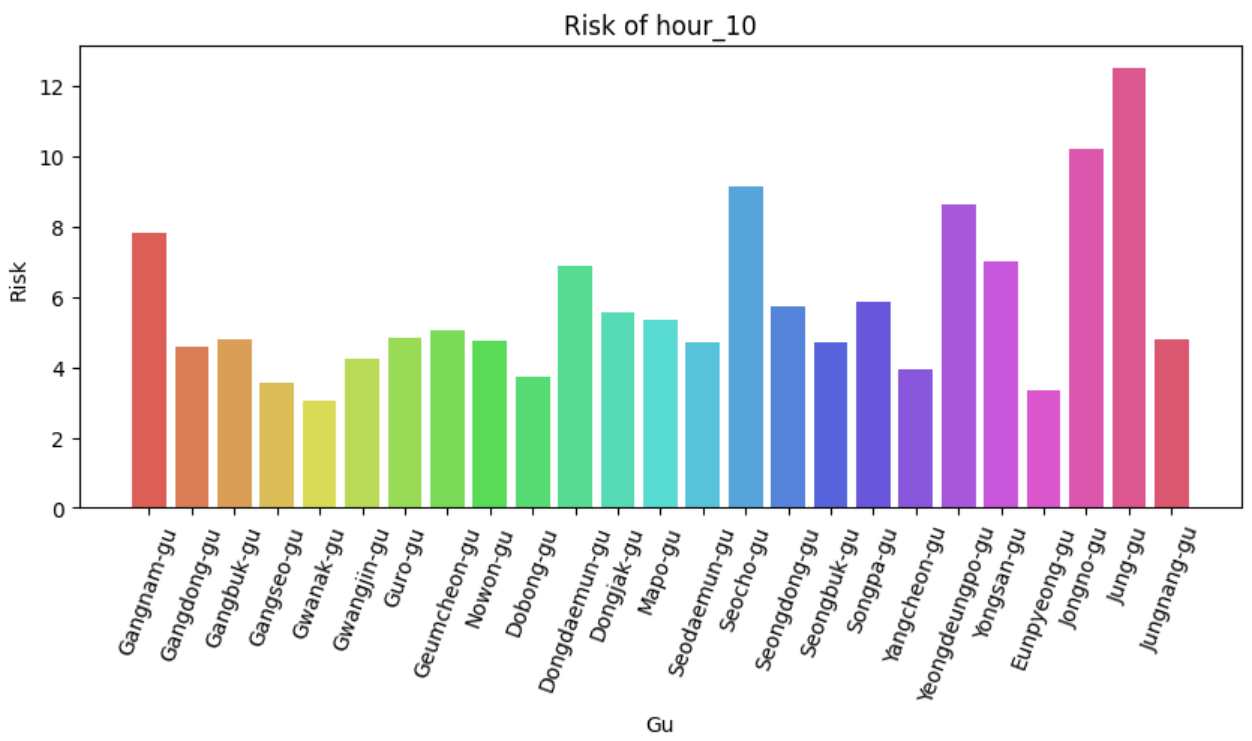
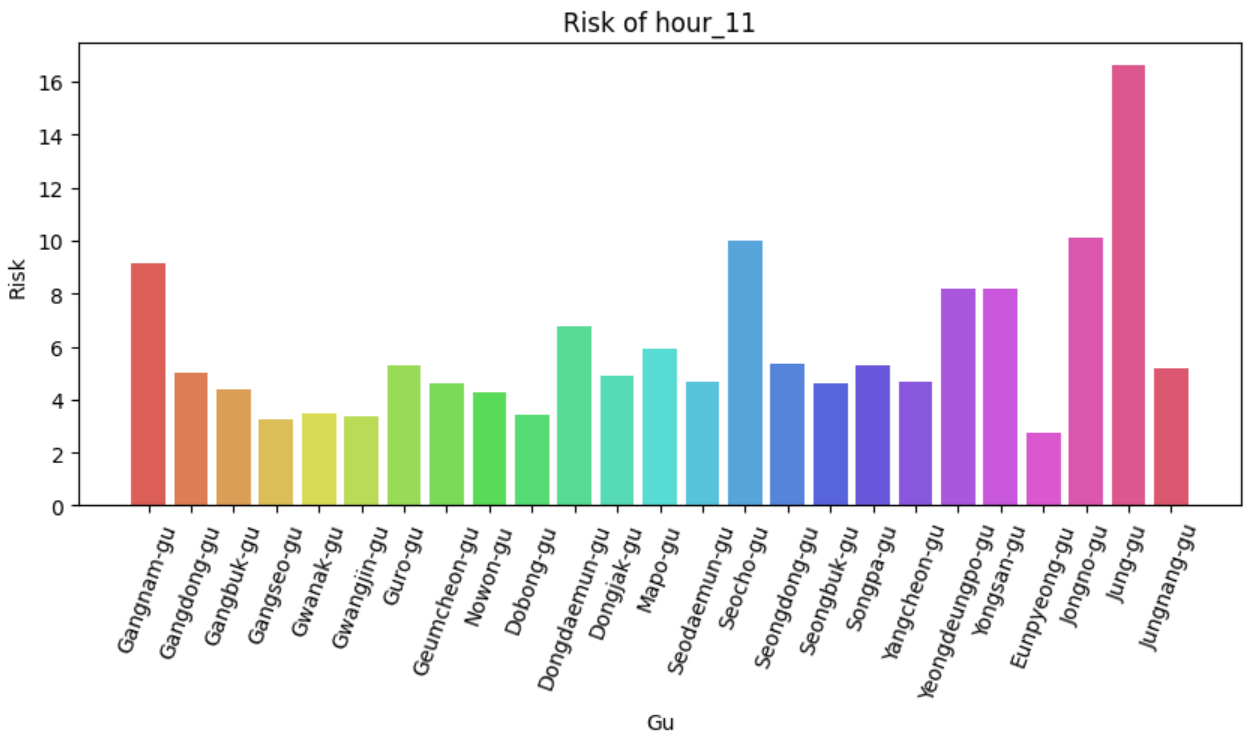
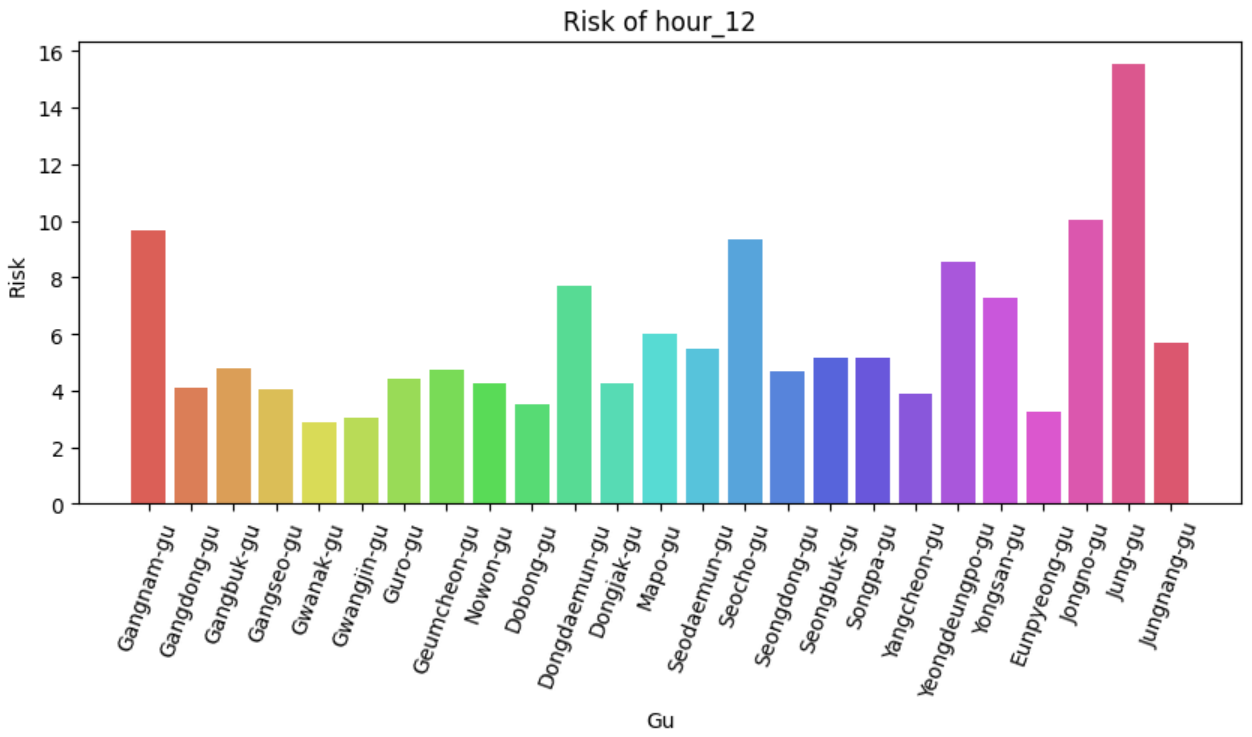
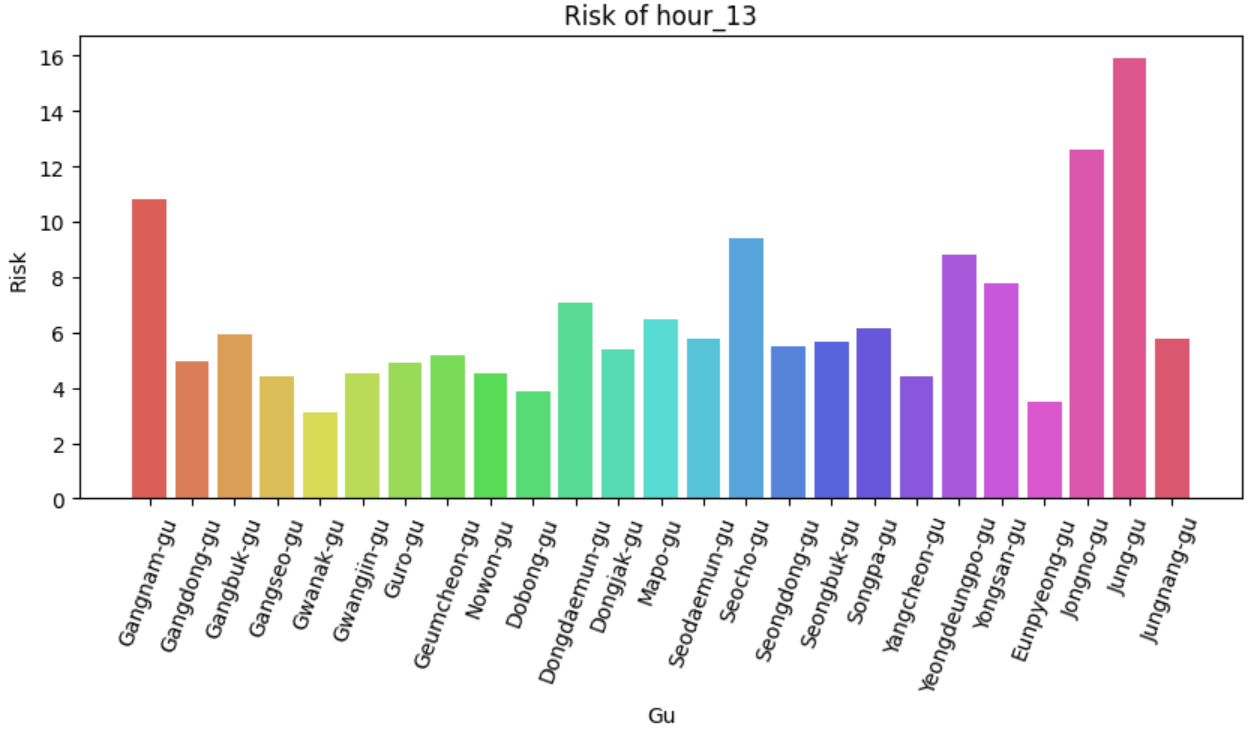
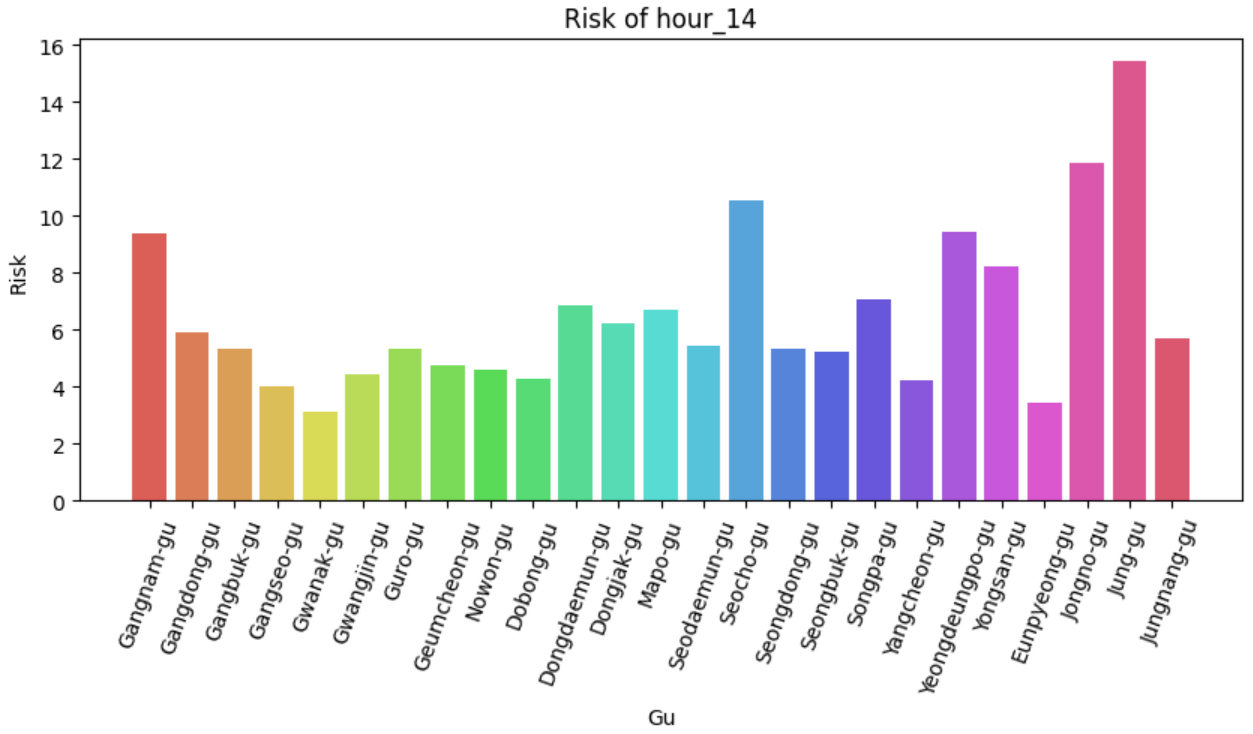
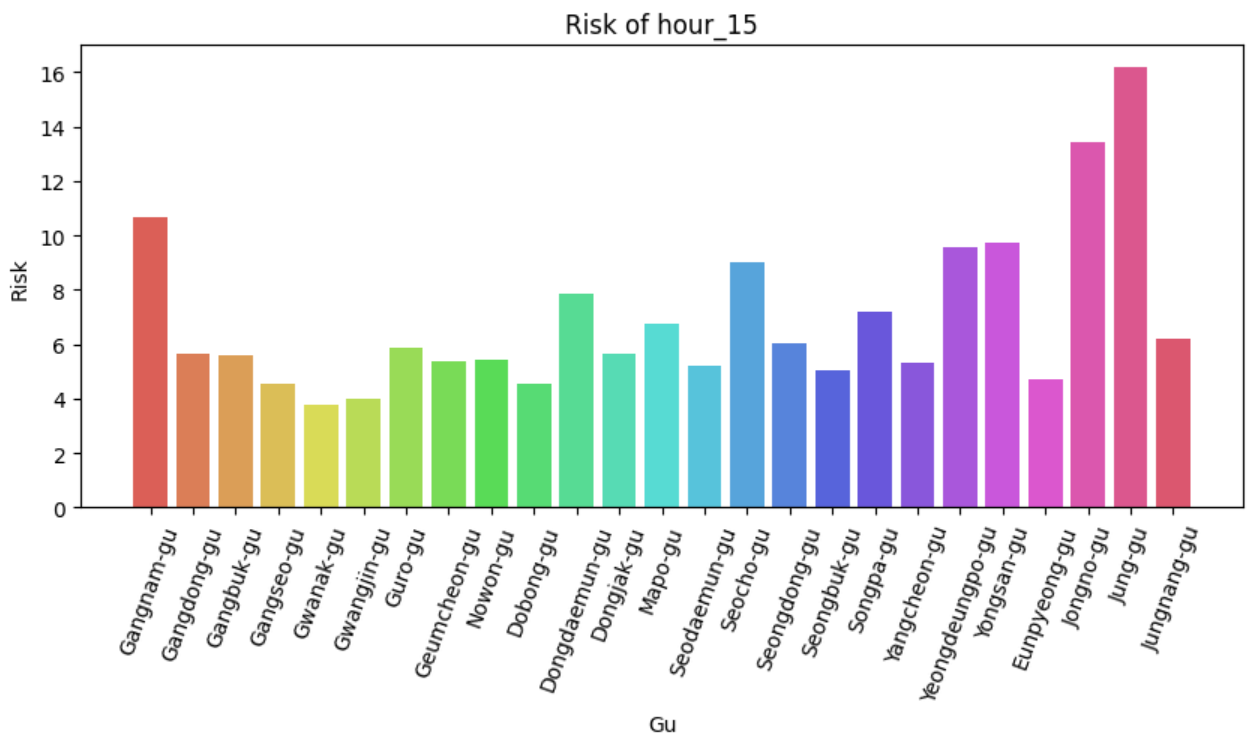
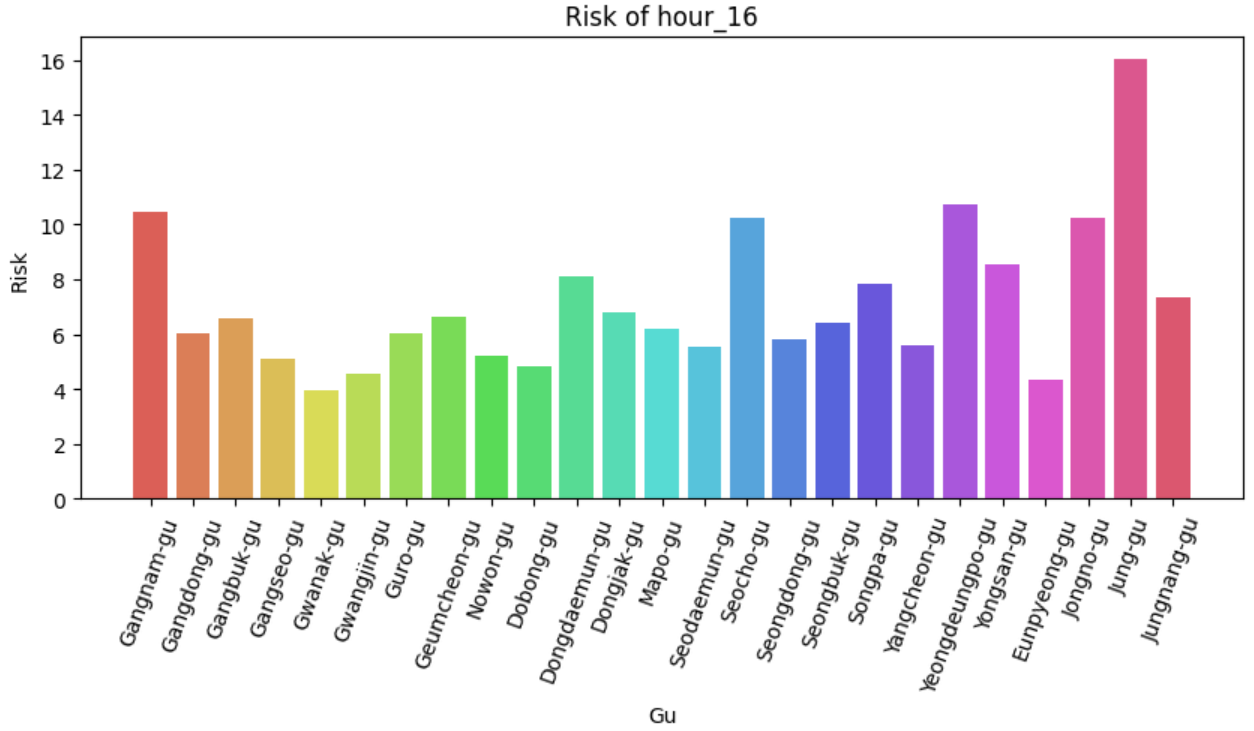
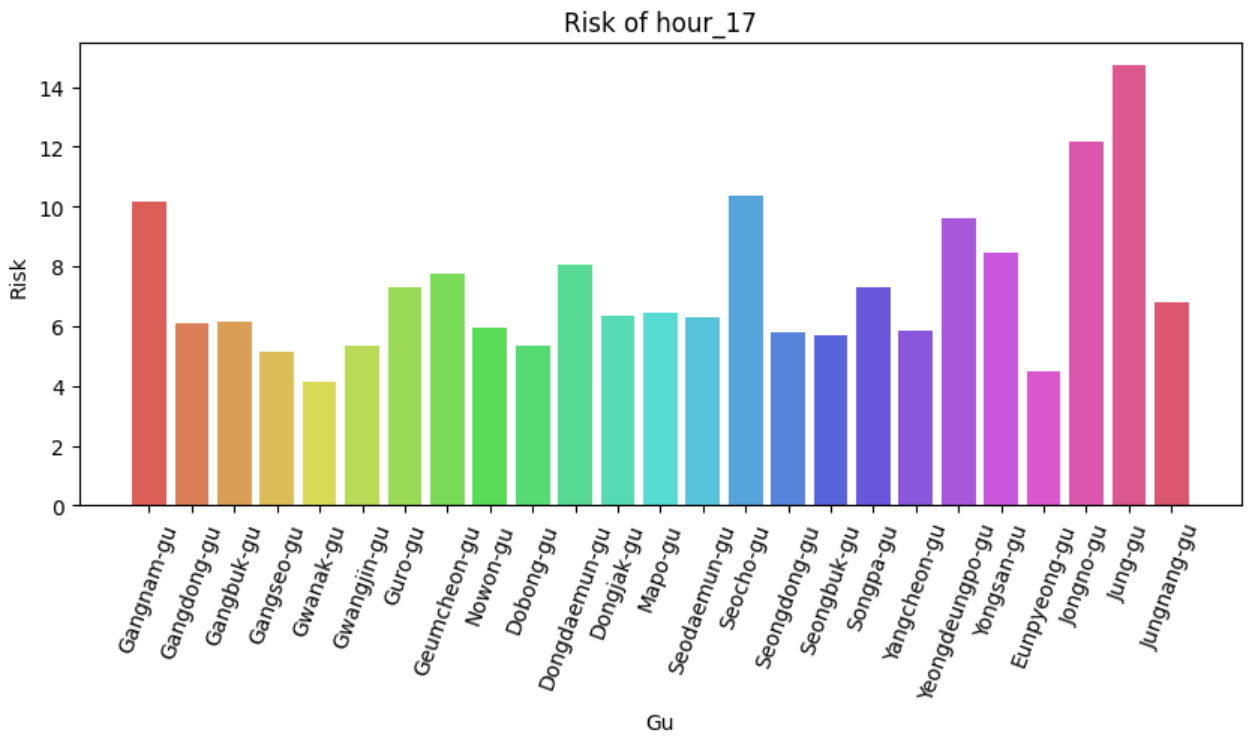
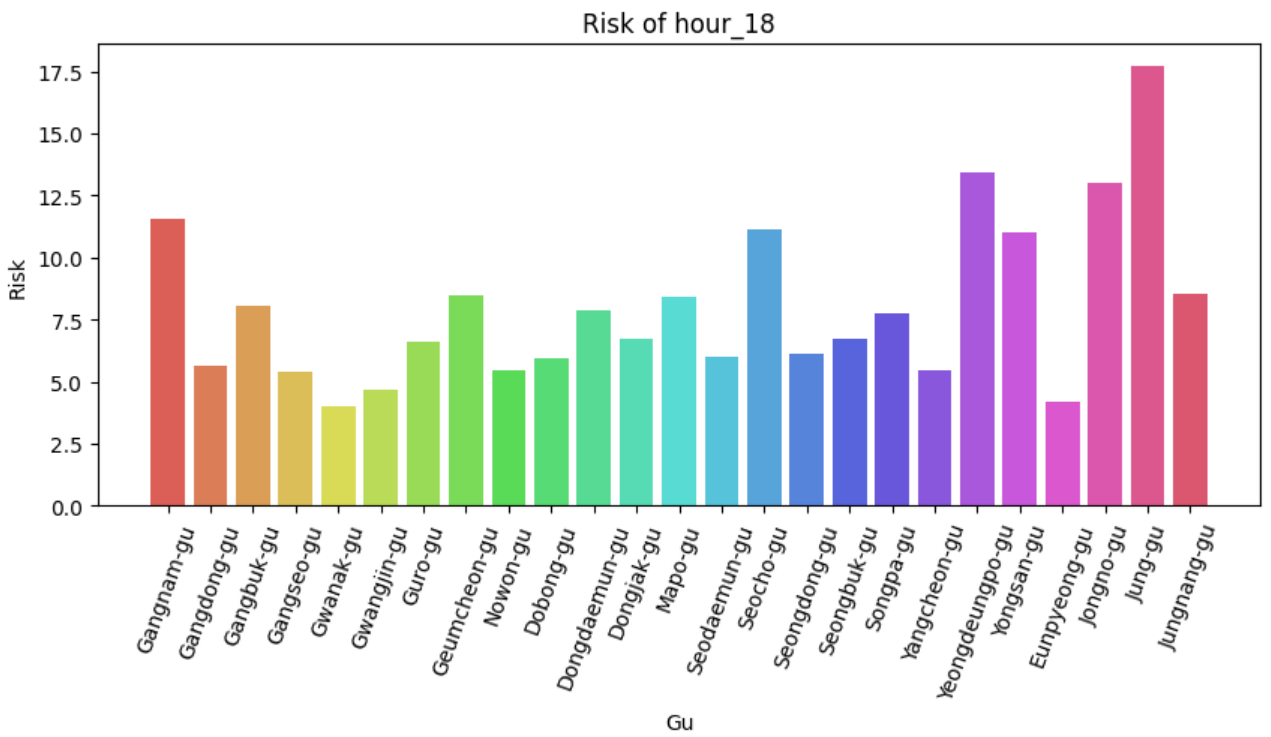
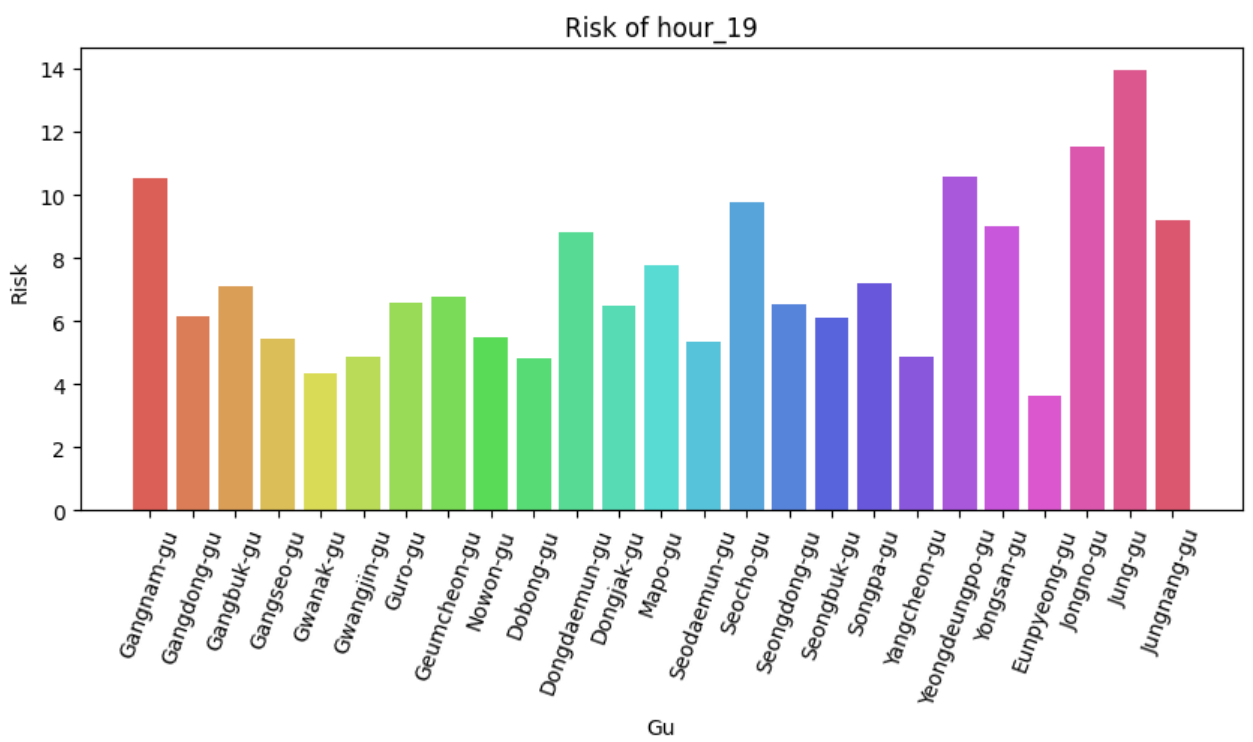
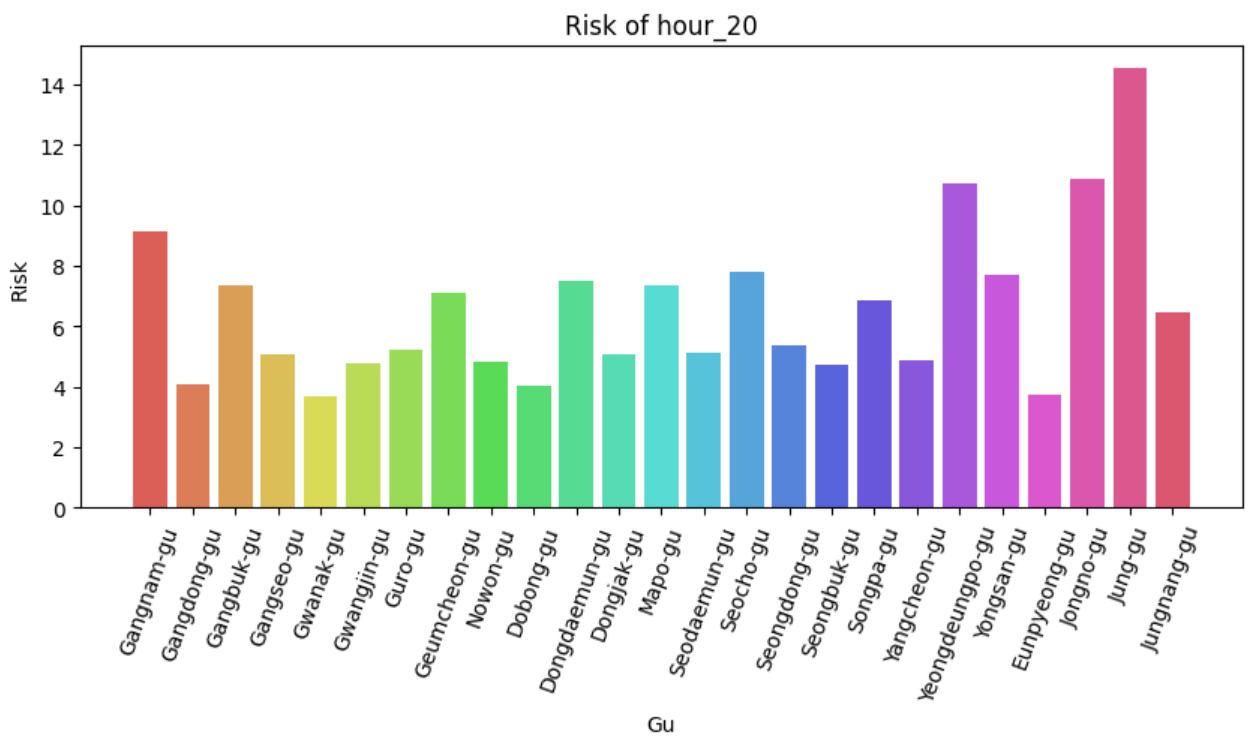
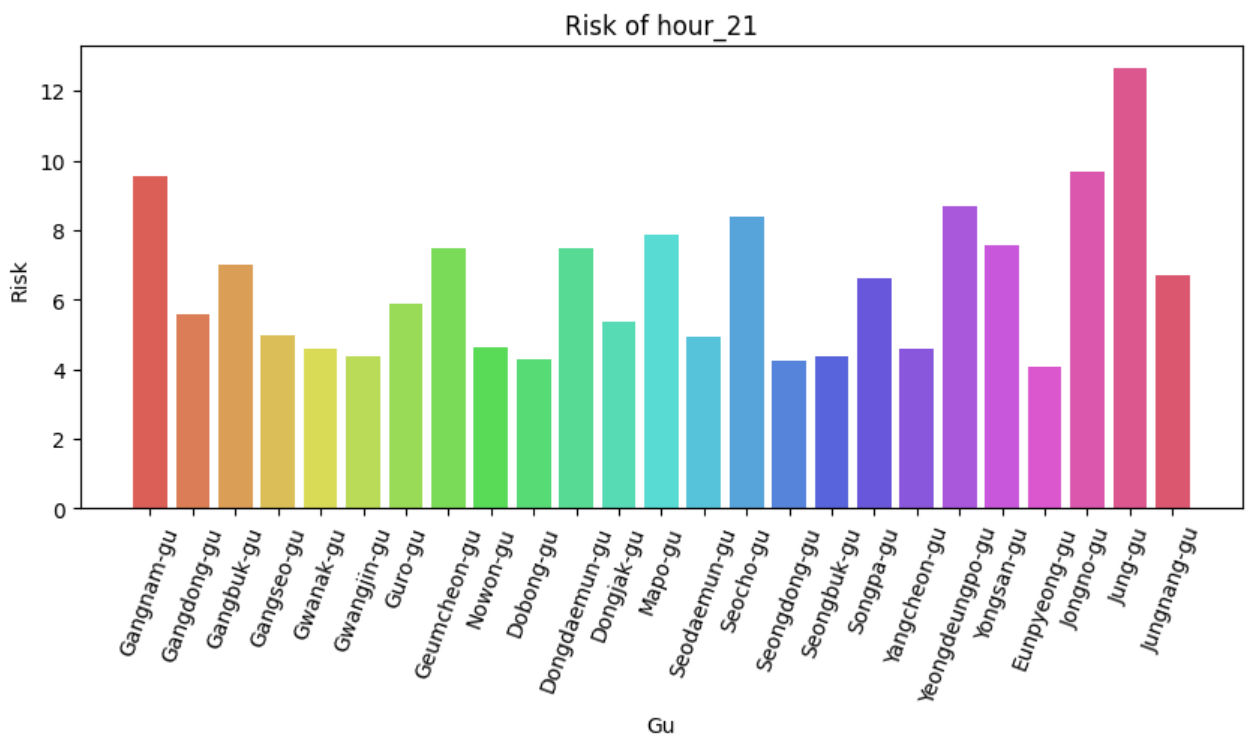
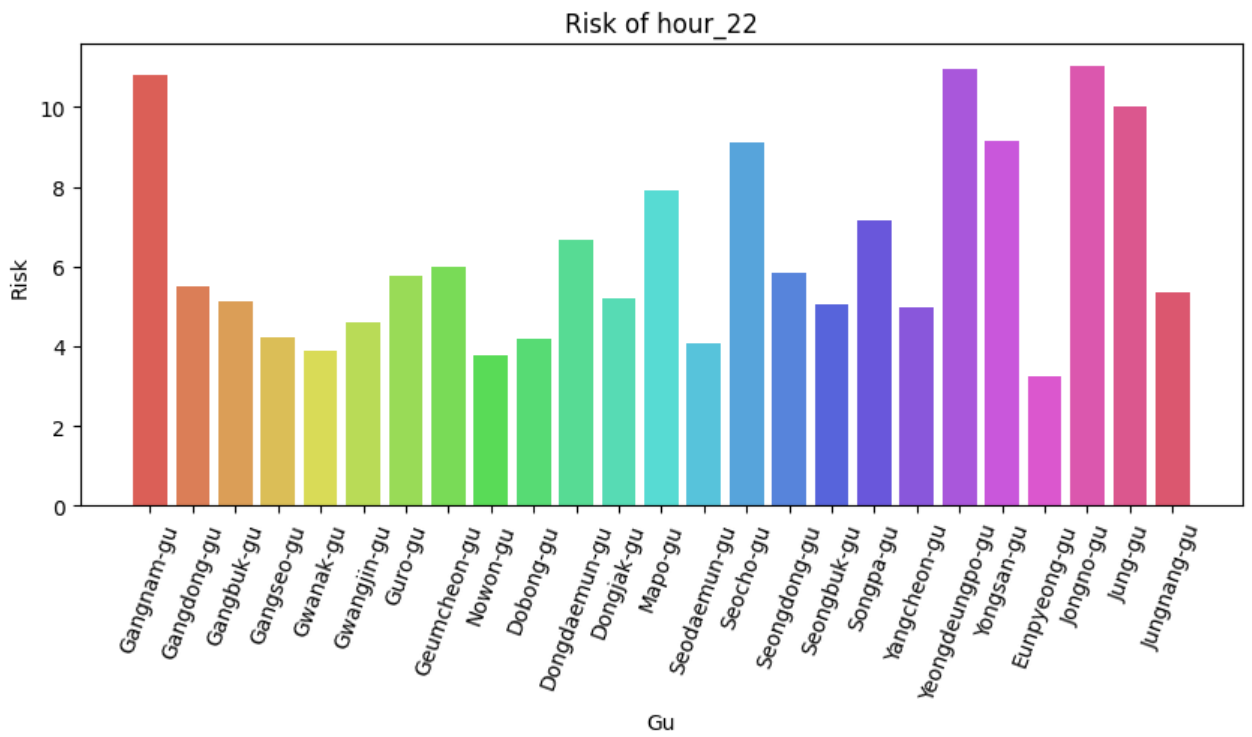
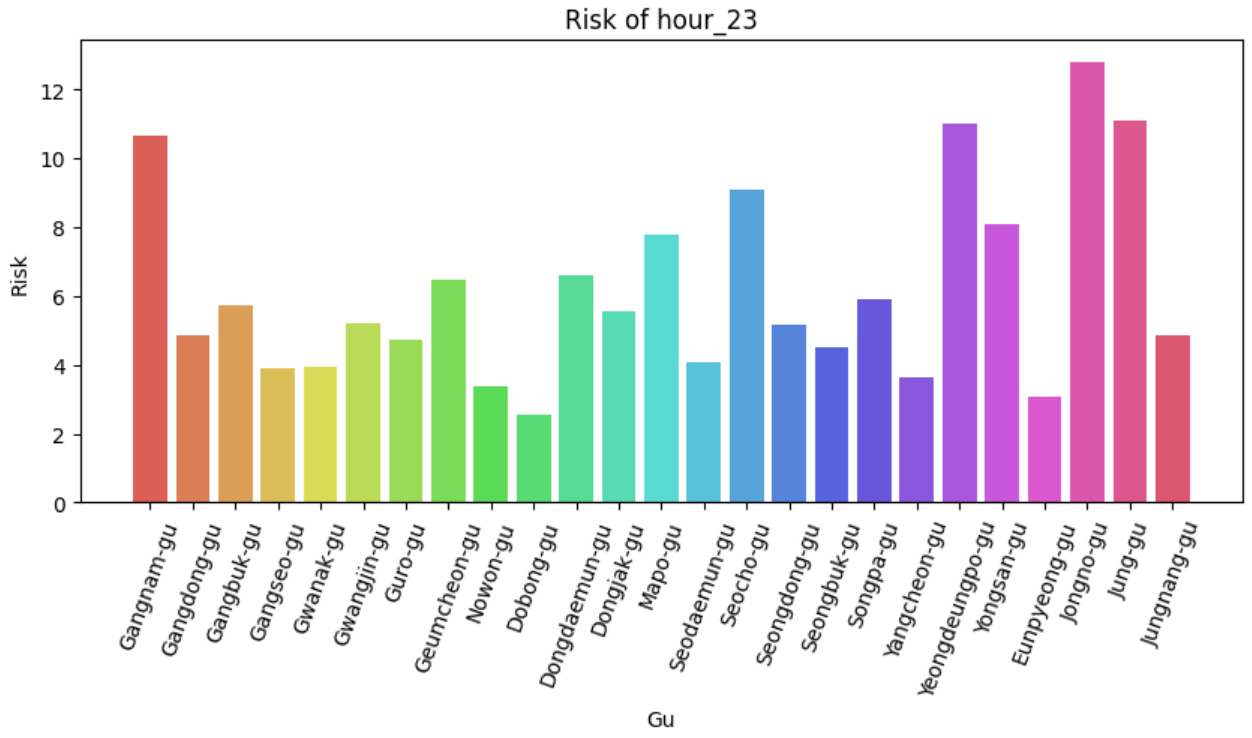


make\_test\_df(region, month, time, day) function takes four inputs: region (동/지역), month (월), time (시간), and day (요일). the code generates input data for the base model by combining a fixed region, month, and day with varying time values. It then uses the generated data to make predictions using the base model, printing the predicted risk score for each hour of the day.

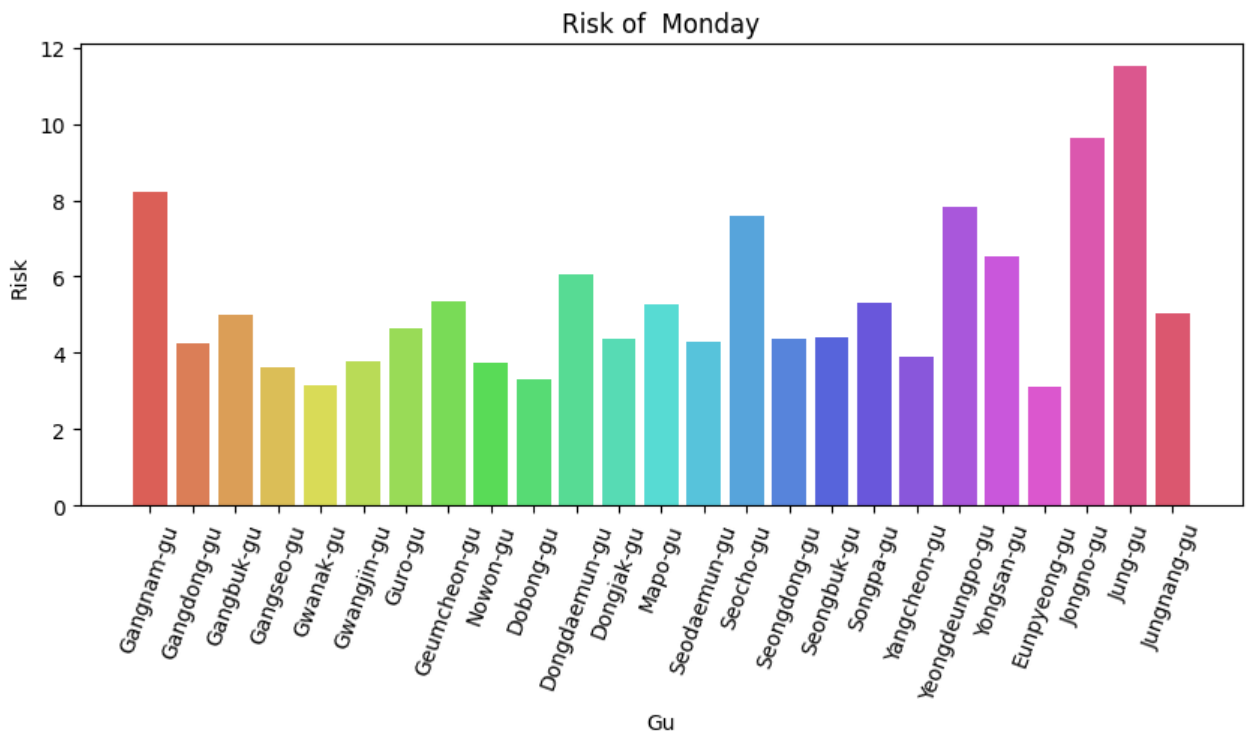
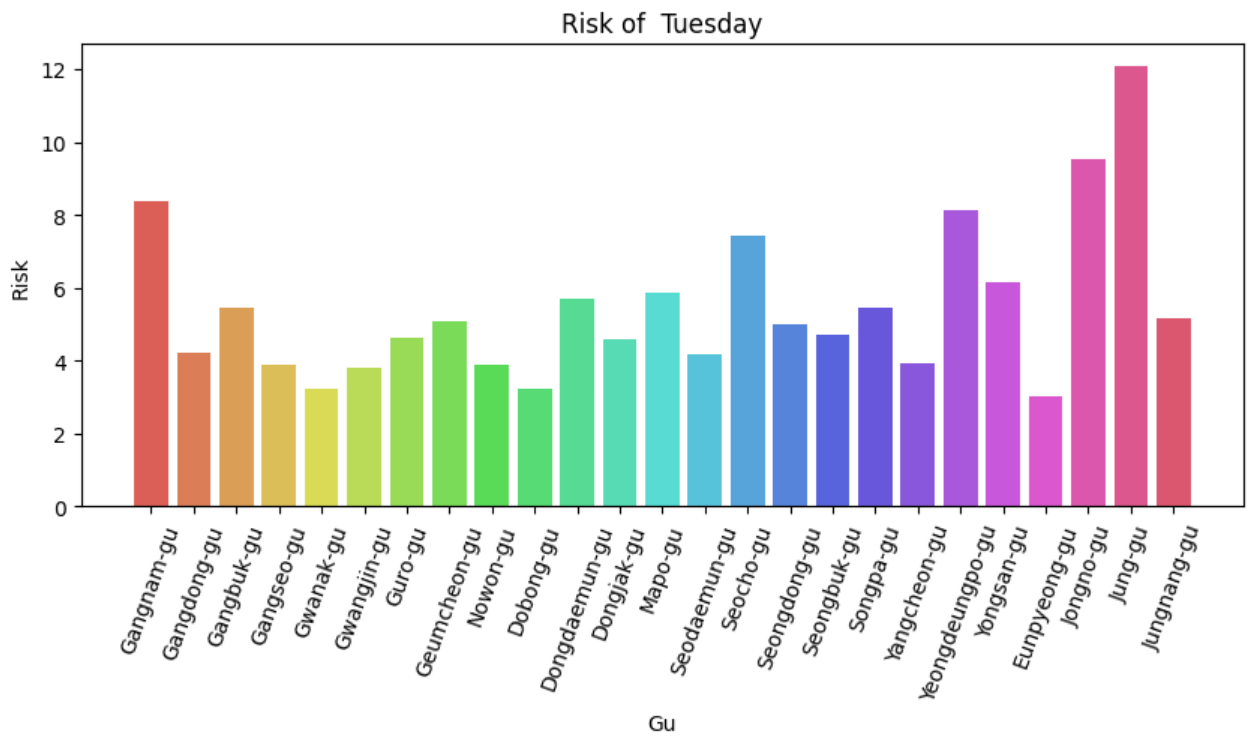
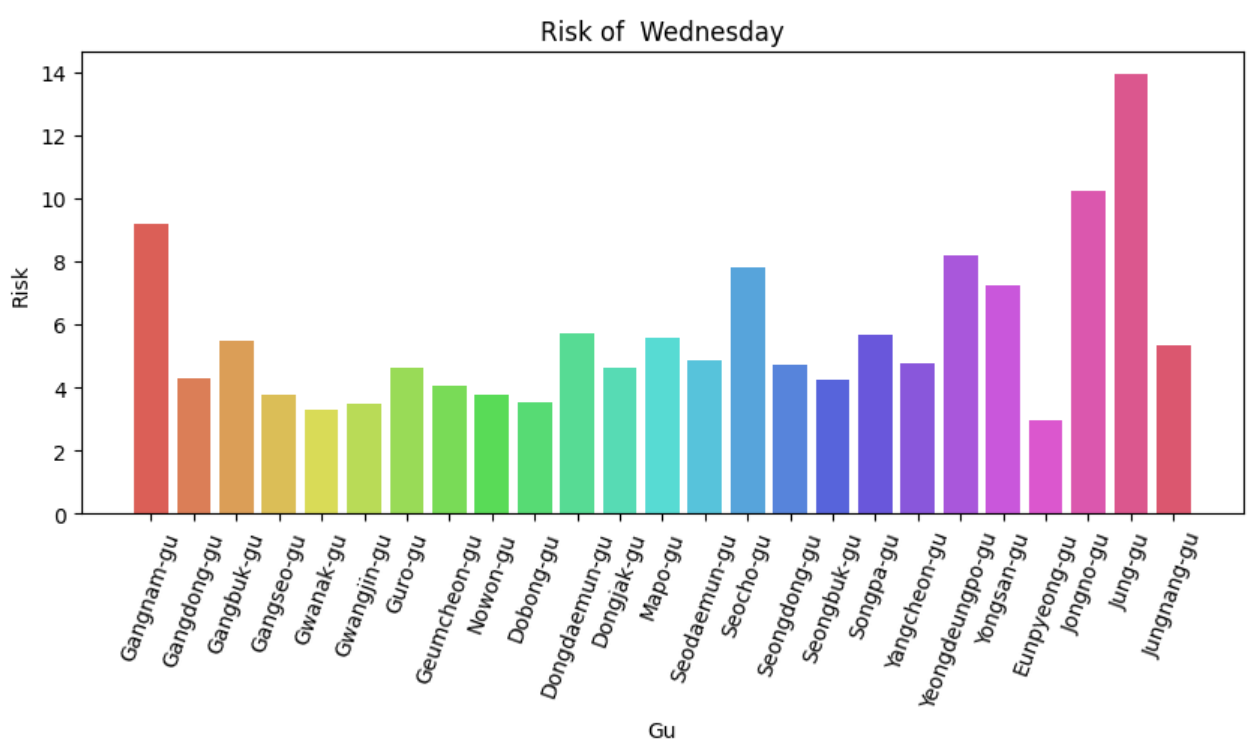
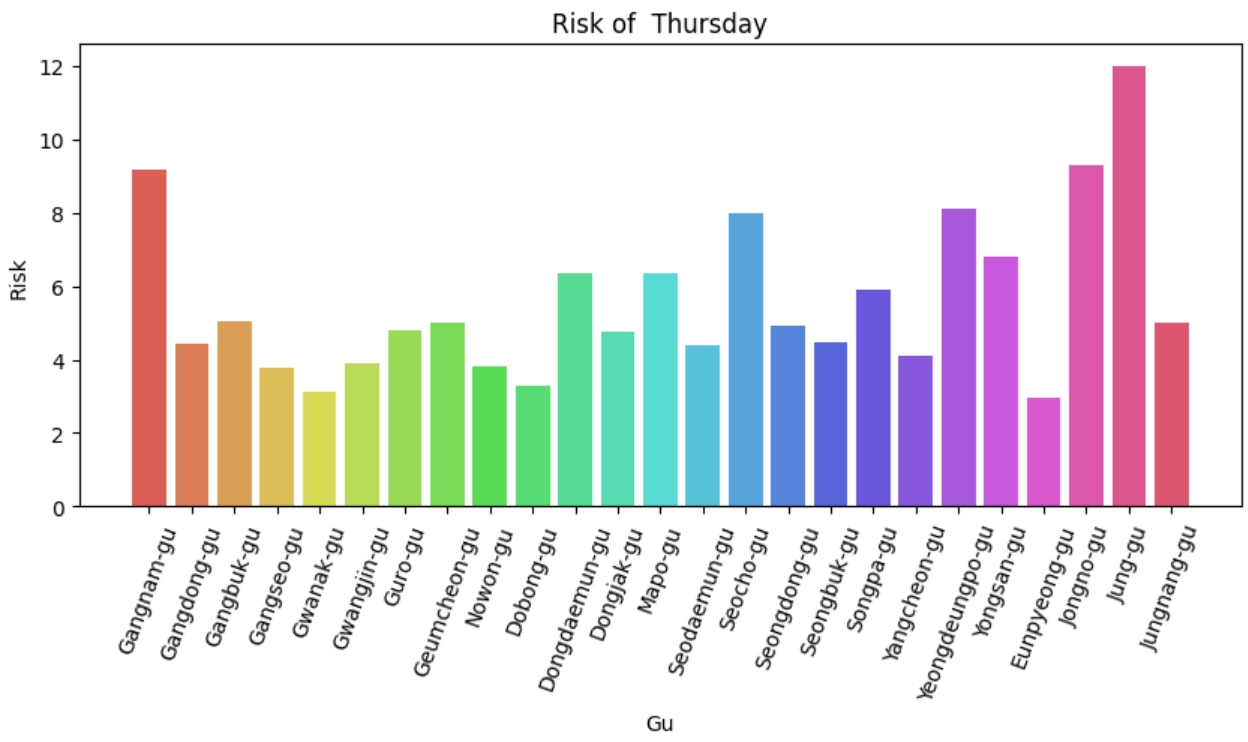
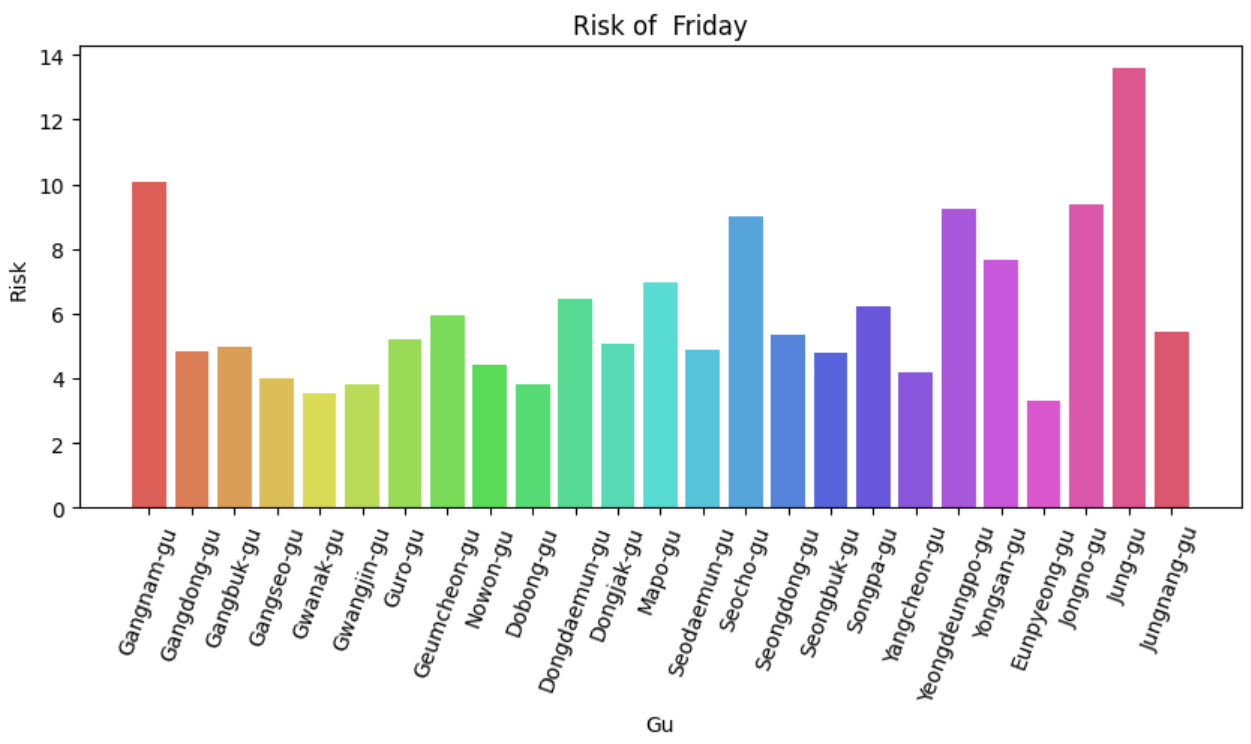
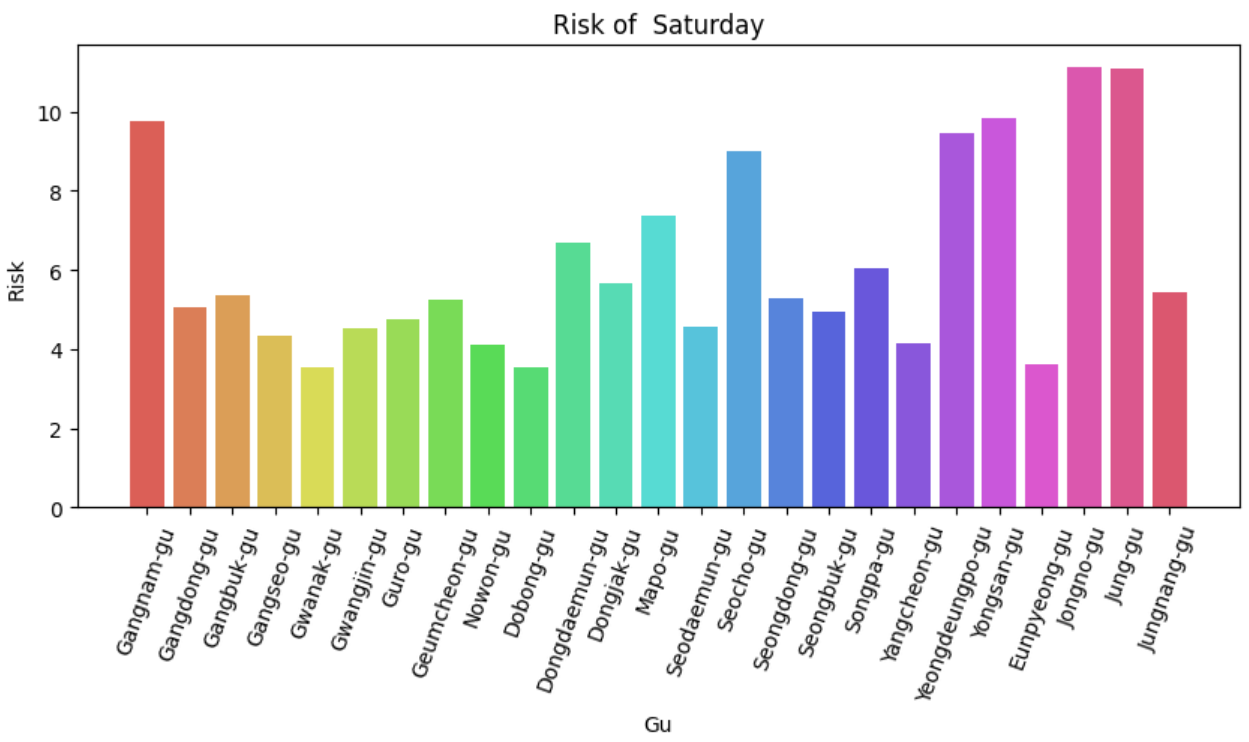
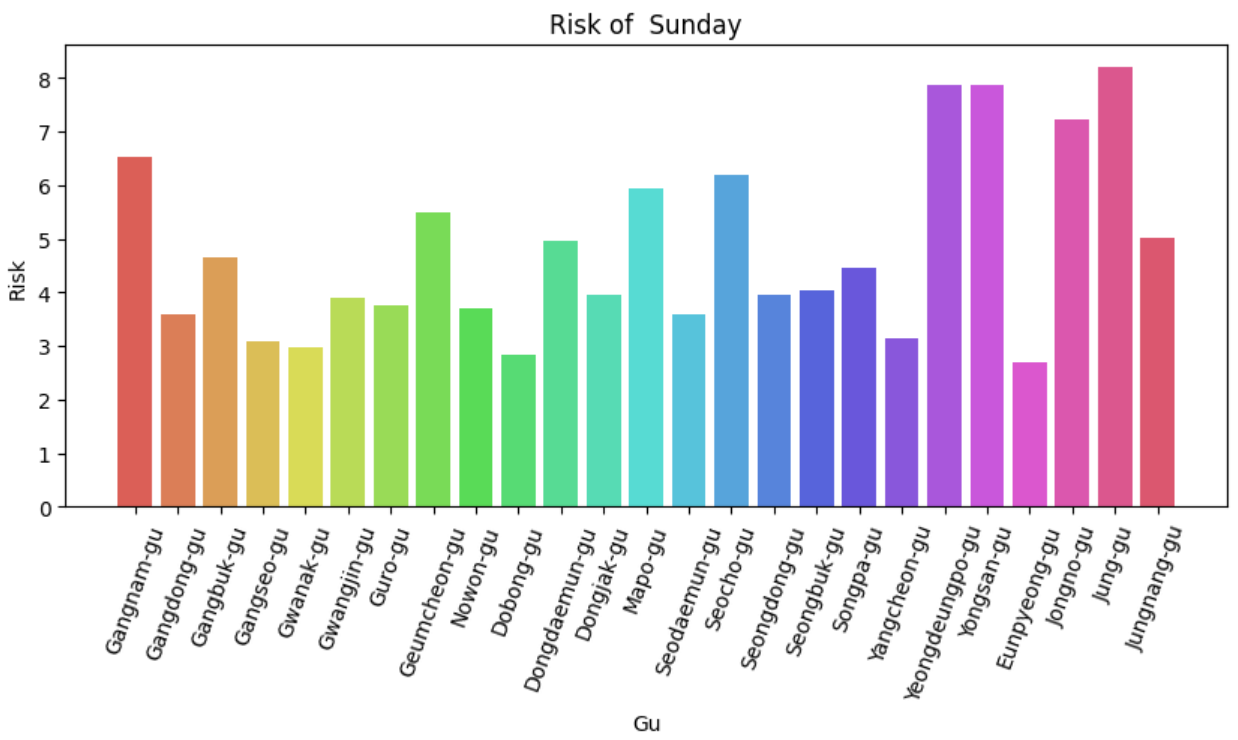
The predict data of “장지동”, “6월”, “수요일”, “0~23시” is D

Data Analysis of Accident Risk by Hour

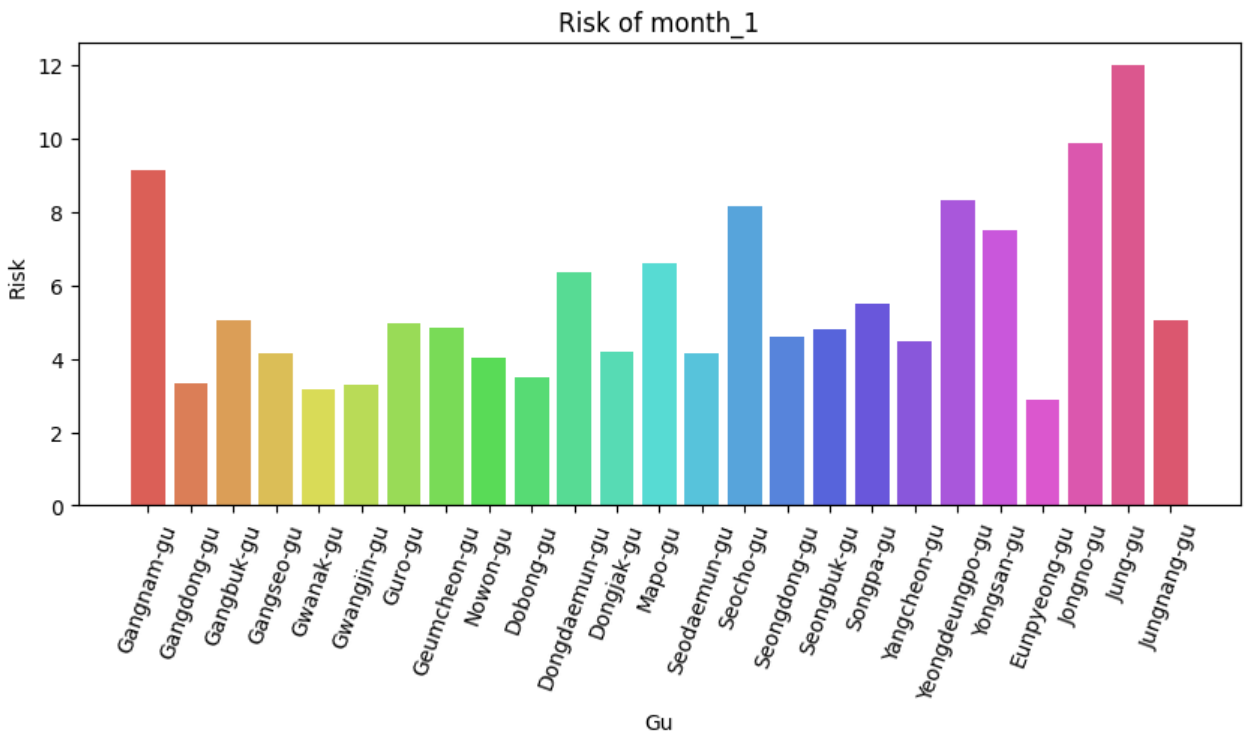
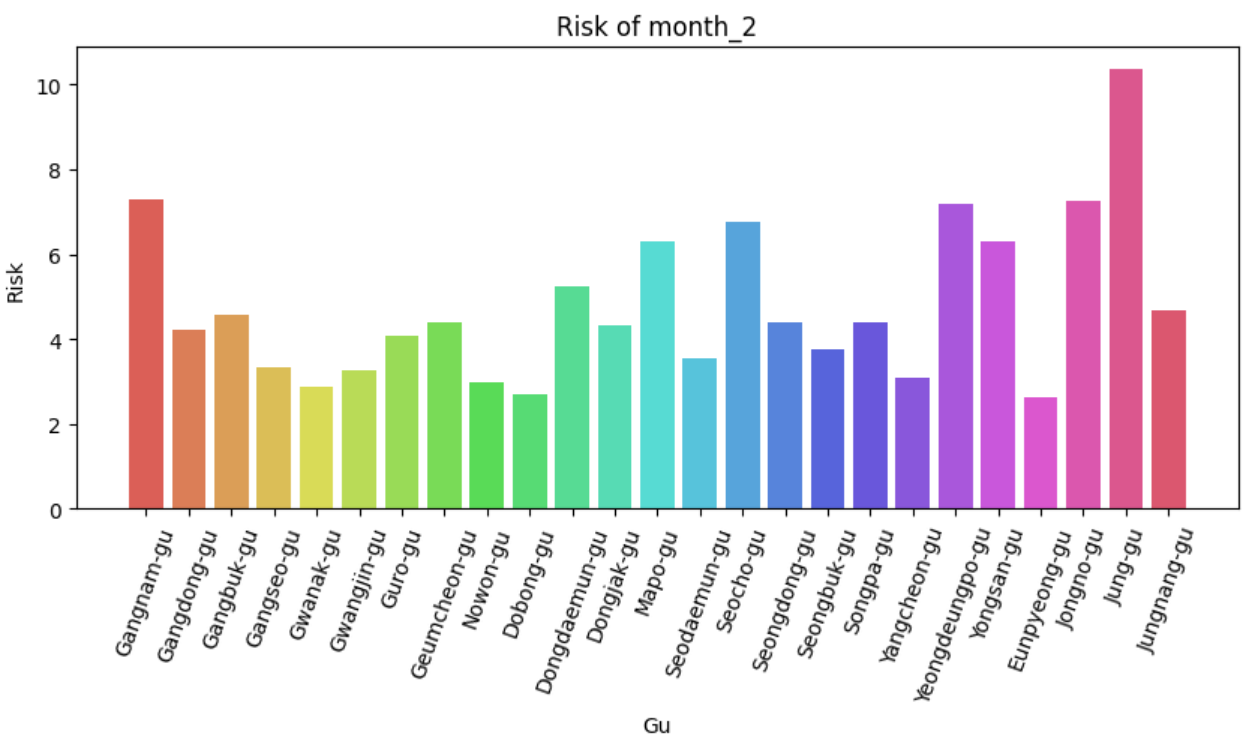
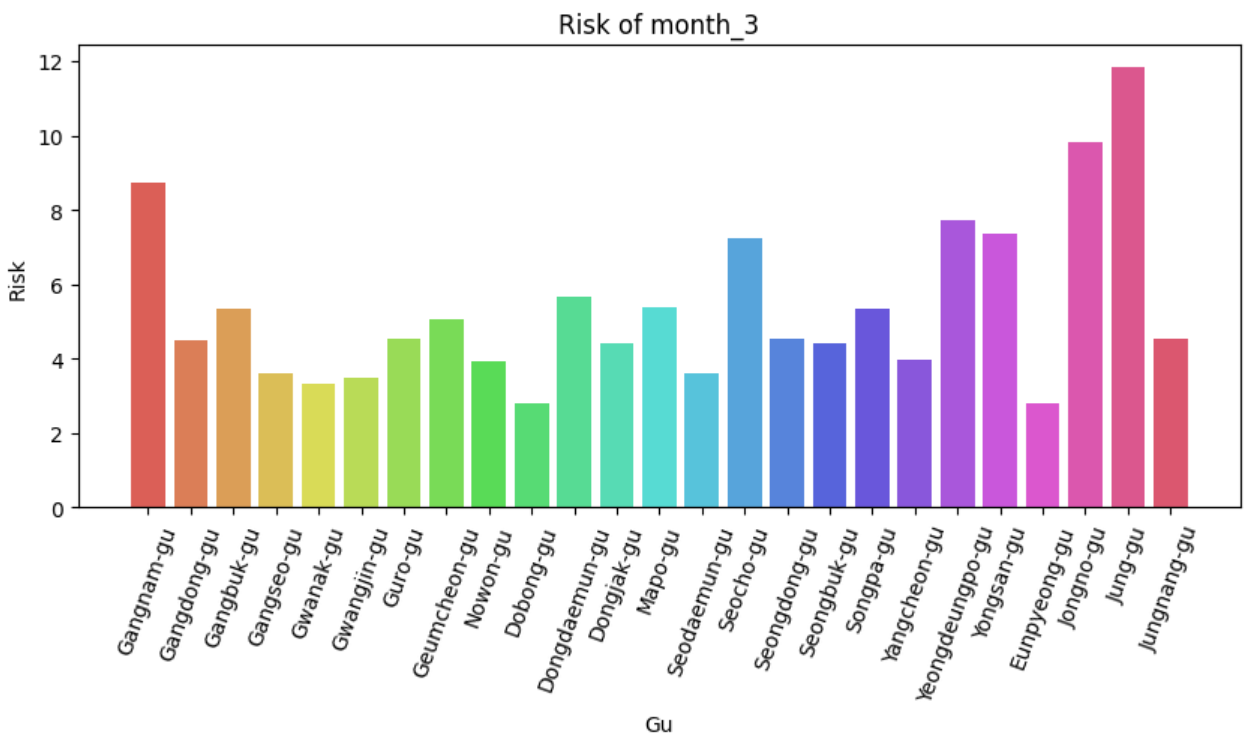
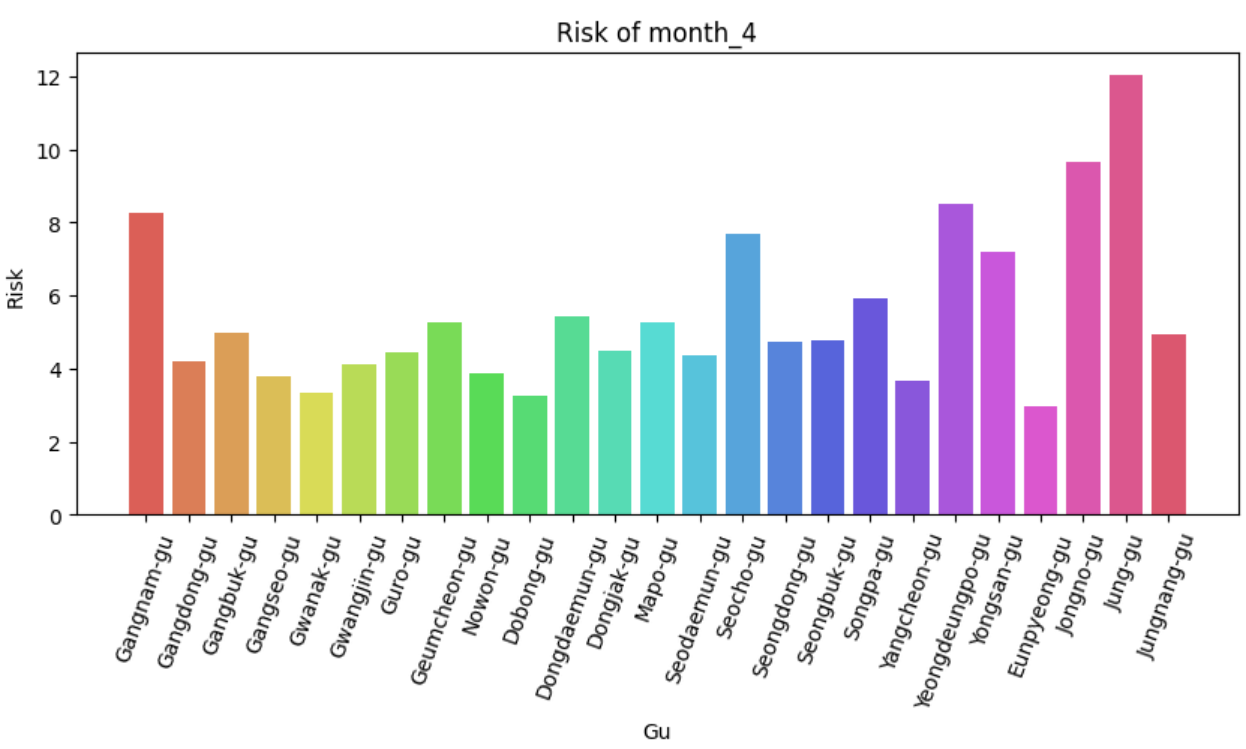
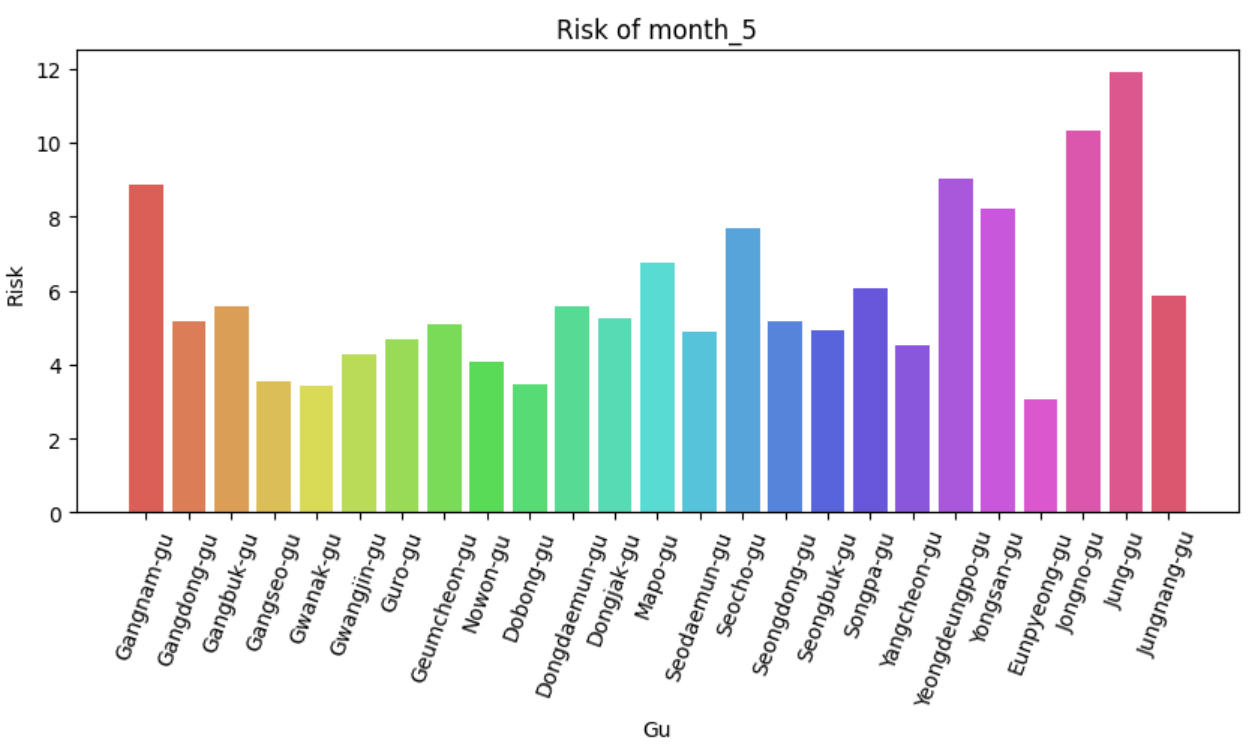
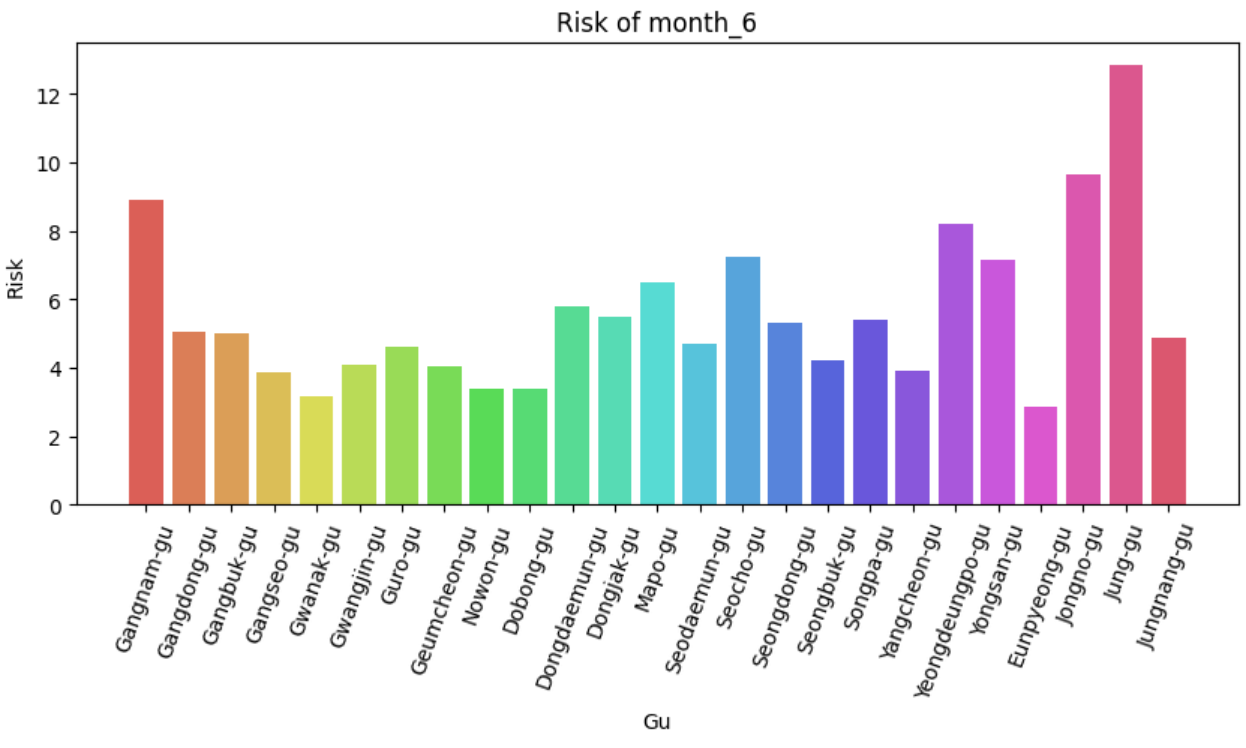
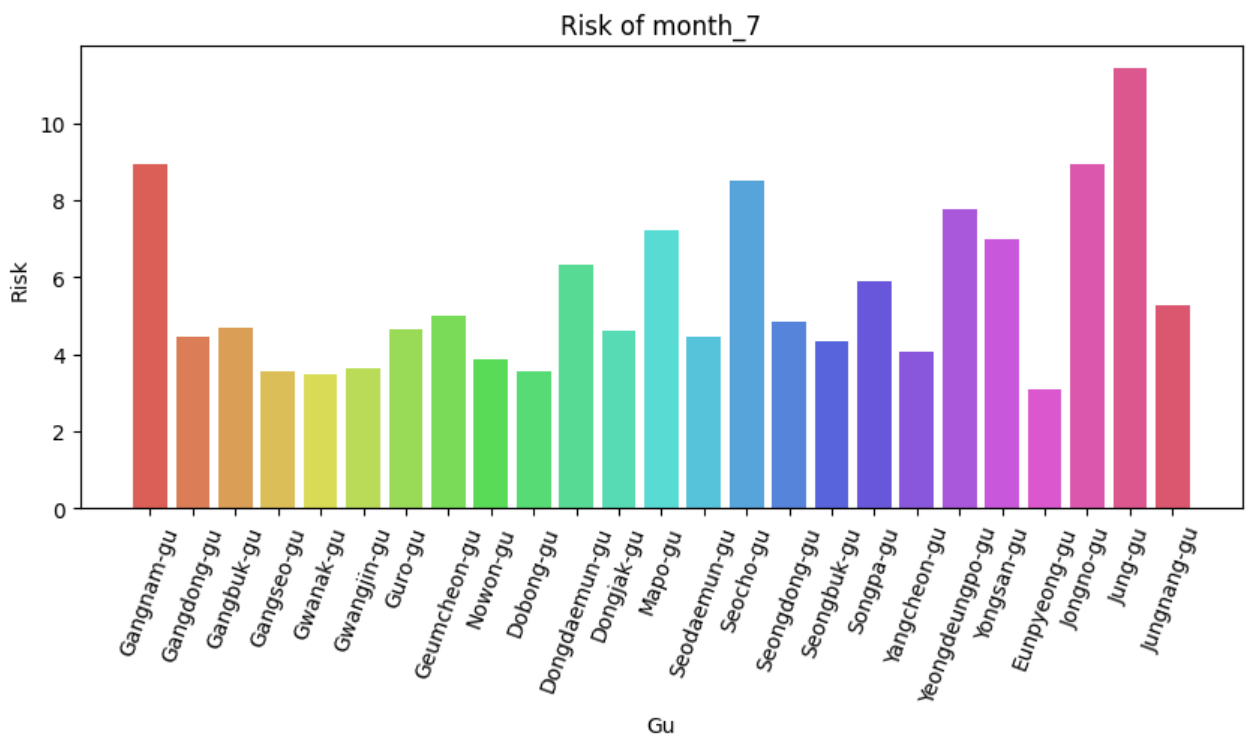
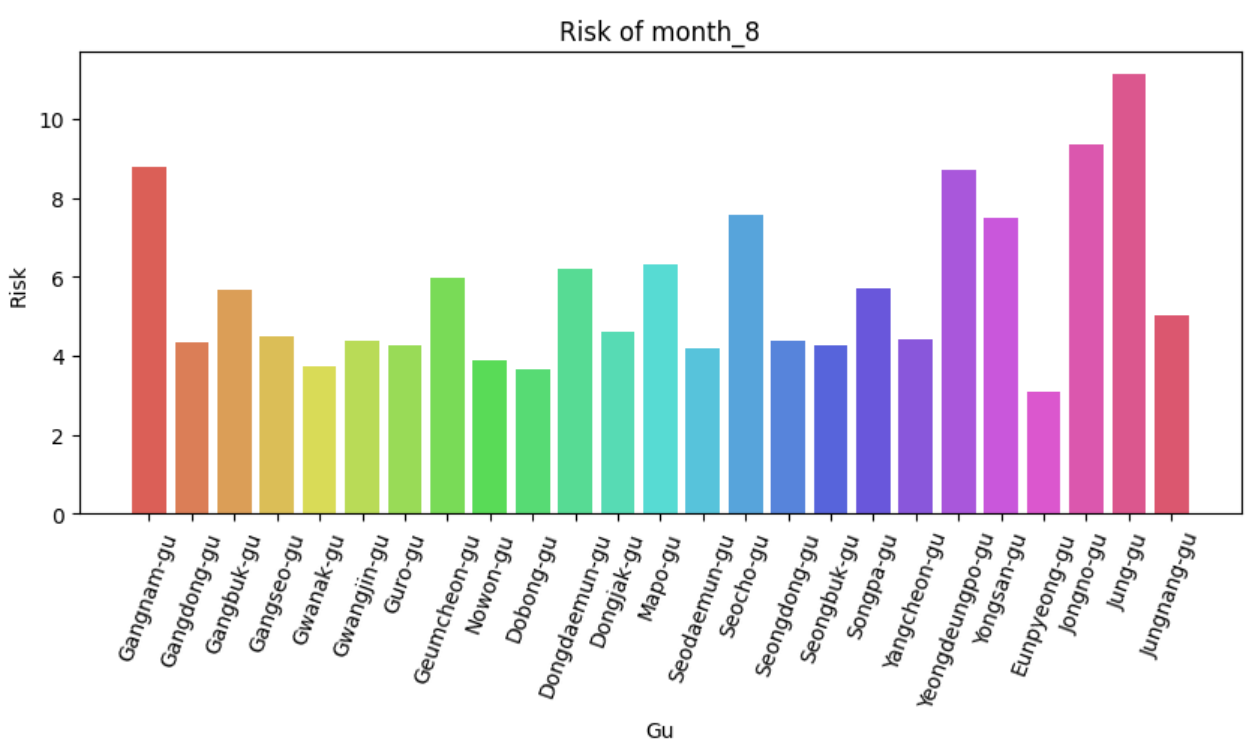
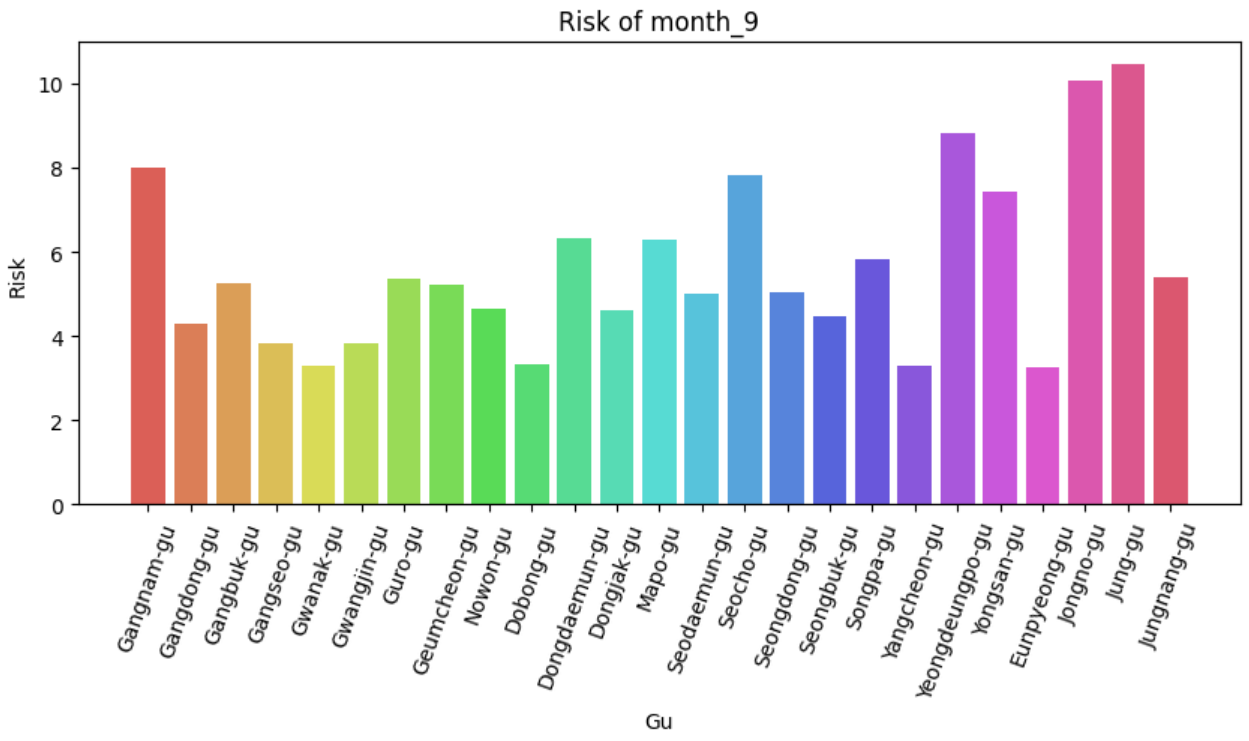
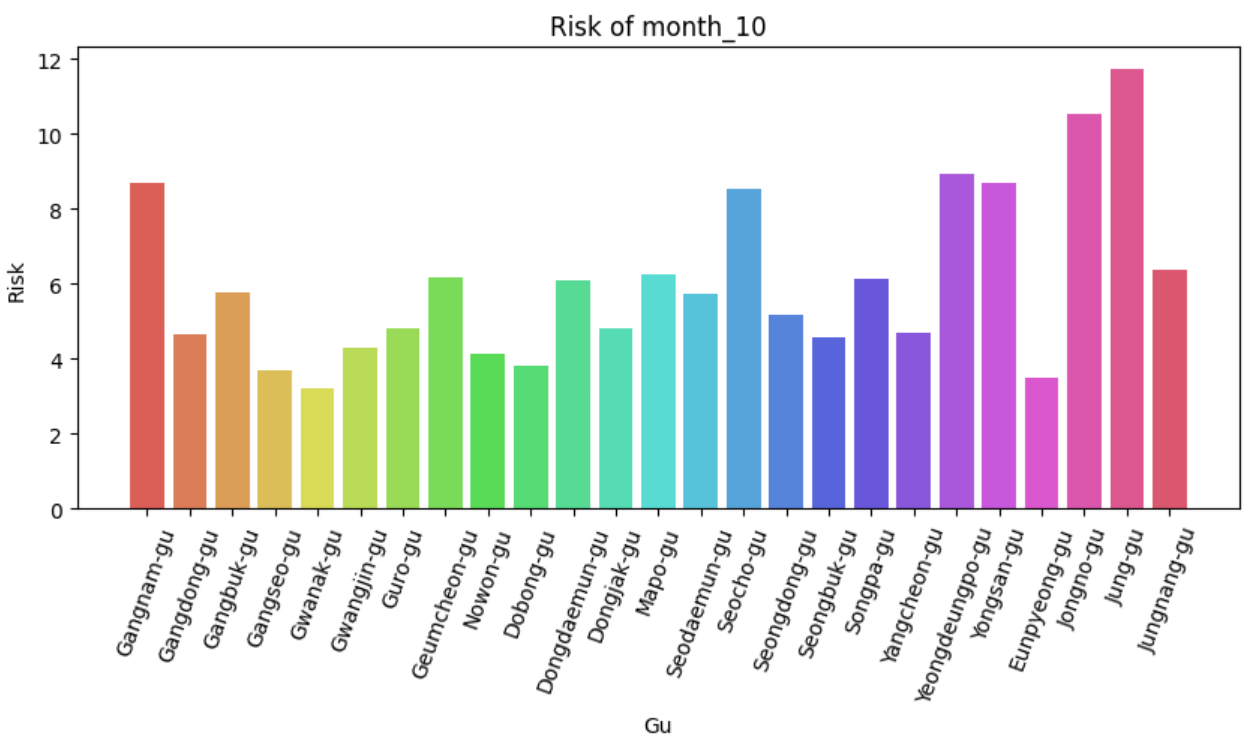
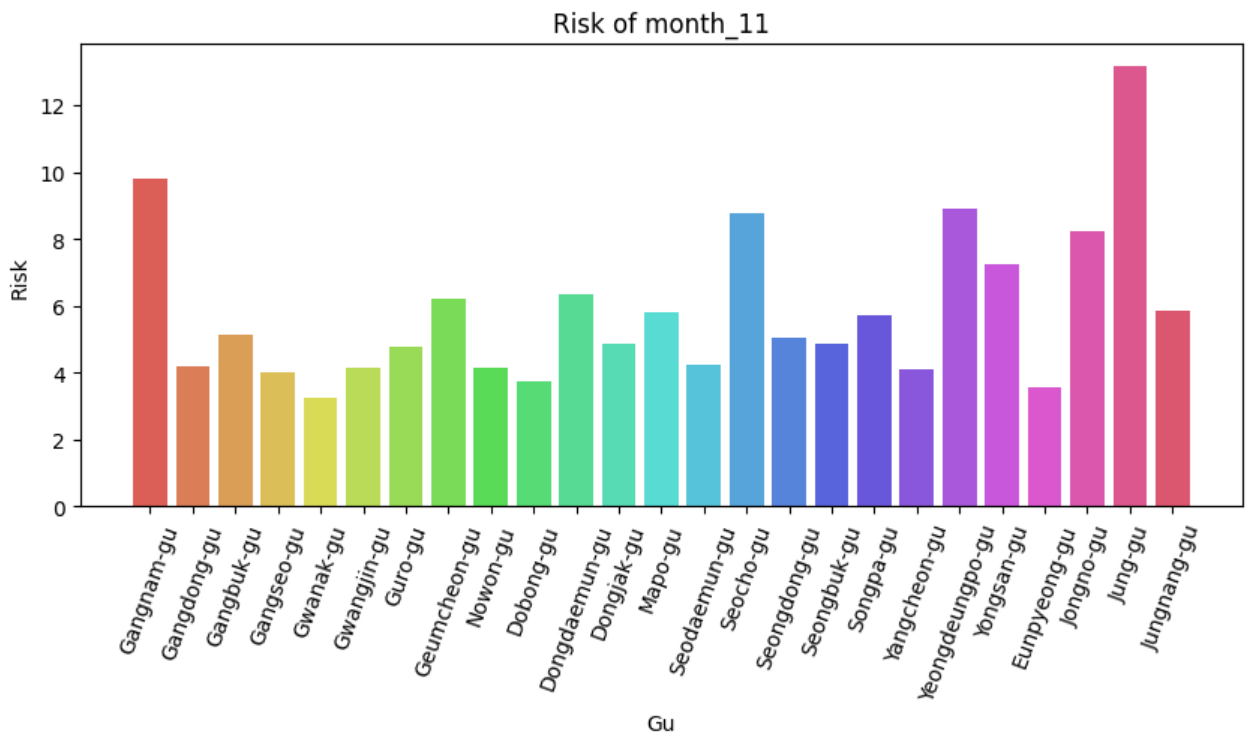
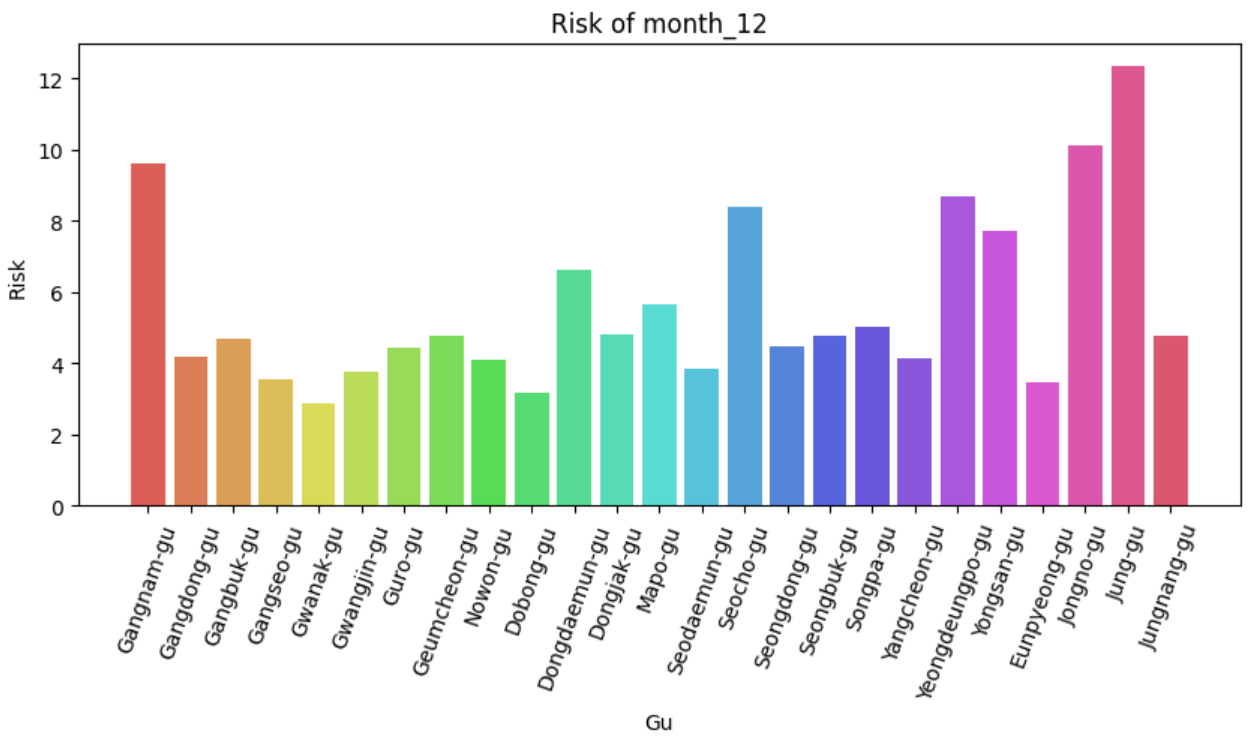
 

Data Analysis of Accident Risk by Week

Data Analysis of Accident Risk by Month

<External factors that can influence the analysis>

By using accident data based on date and time, it analyzes the accident probability for a given date and location. The most significant external factor that greatly influences this analysis is the weather. Even in areas and time periods where accidents are less likely to occur, the accident probability significantly increases on foggy or rainy days. According to the Road Traffic Authority(도로교통공단), the fatality rate in traffic accidents is more than three to four times higher on foggy days. Due to considering only specific dates and locations, our team's data analysis results are influenced by these external factors.

Matching the accident dataset's location and quarter with the population dataset, especially when the region names differ, was the most challenging part.

**Learning experience**

* Kim Sejung : At first, I thought of data modeling simply. However, in the process of generating the decision tree, I found that the accuracy depends on how deep the decision tree will be and how much train/test set will be used. Therefore, it was necessary to find the optimal parameters through the k-fold step and select a model with high accuracy. In this process, I learned that it is important to set optimal parameters for data modeling. We also learned that it takes a lot of effort because there are no indicators for correct modeling.
* Roh Hyeonji : The process of preprocessing and analyzing various data to produce one meaningful result value was both novel and rewarding.
* Yang Dain : Through data exploration, I was able to learn how data is structured and classified. During the process of selecting and validating data for data analysis, I learned how to read and review data objectively in order to achieve project goals. I thoroughly researched the format and types of data, as well as the content of the data and its relevance to the project. Additionally, I had the opportunity to contemplate which data is important and consider the relationships between different data, gaining my first experience in thinking about good data.
* Oh Jinyoung : I have learned about the importance and difficult of data preprocessing in data science. The preprocess of matching values across different data files and creating the desired dataset can be difficult, but it is crucial to perform this step well in order to obtain better results when running data models.

**Teamwork**

* Kim Sejung: 25% / Data preprocessing, Clustering, Classification, Evaluation
* Roh Hyeonji: 25% / Data preprocessing, Clustering, Classification, Evaluation
* Yang Dain: 25% / Data preprocessing, Clustering, Classification, Evaluation
* Oh Jinyoung: 25% / Data preprocessing, Clustering, Classification, Evaluation, Presentation

**GitHub url**

<https://github.com/OJOJIN/seoul-traffic-accidents-analysis>

Reference

* <https://bd.kma.go.kr/contest/downloadFile.do?fileCd=FILE008>
* <https://m.datanews.co.kr/m/m_article.html?no=78031>
* <https://www.youtube.com/watch?v=Q7xPoIGWhiM>