Term Project Report

Gotta Predict 'Em All !

โดย

65070501027	น.ส.	ธัญมน	ราชเวียง
65070501028	น.ส.	ธิชานนท์	สิทธิสมบูรณ์
65070501047	นาย	รอซีน	มูเก็ม
65070501058	นาย	สุวิจักขณ์	เรี่ยวแรงบุญญา
65070501068	นาย	คณัสส์	สุวรรณรัตน์

นักศึกษาชั้นปีที่ 2

เสนอ

ผศ.ดร.สันติธรรม พรหมอ่อน

รายงานเล่มนี้เป็นส่วนหนึ่งของรายวิชา **CPE232 Data Models**

คณะวิศวกรรมศาสตร์ สาขาคอมพิวเตอร์

Introduction

“Pokémon Go” เป็นหนึ่งในเกมมือถือที่ได้รับความนิยมอย่างแพร่หลายไปทั่วโลก ซึ่งได้มีการนำระบบการเชื่อมต่อเครือข่ายโทรศัพท์มือถือ และการจับตำแหน่งผ่าน GPS เข้ามาเป็นส่วนหนึ่งของเกม ทำให้ผู้เล่นสามารถจับโปเกมอนในโลกจริงได้ โดยที่โปเกมอนจะปรากฏตัวในตำแหน่งต่าง ๆ ของโลกจริง และมี "PokeStops" กับ "Gyms" ที่จะตั้งอยู่ตาม Landmark ในโลก ซึ่งนั่นเป็นจุดสำคัญในการเล่นเกม ในบางครั้งจะมีการกำหนดจุดเกิดของโปเกมอนในตำแหน่งพิเศษ ซึ่งนั้นถือเป็นจุดเด่นสำคัญที่ทำให้เกมนี้เป็นเกมที่ได้รับความนิยม รวมไปถึงเป็นสาเหตุที่ทำให้เกมนี้มีความสำคัญในเชิงสังคม เนื่องจากเป็นการผสมผสานระหว่างโลกเสมือนและโลกจริงทำให้ผู้คนต้องออกจากบ้านและเดินทางไปตามหาโปเกมอน ซึ่งมีผลในการกระตุ้นการเคลื่อนไหว การสื่อสารระหว่างคนในชุมชน และการเชื่อมโยงกับสถานที่ต่างๆ ที่เป็นจุดสำคัญของการเล่นเกม

แต่ถึงกระนั้น หนึ่งในความสนุกของเกมนี้ก็คือ ไม่ใช่ว่าเจ้าตัวโปเกมอนในเกมนี้จะโผล่อยู่ทุกที่ทุกเวลาเสมอไป โปเกมอนนั้นมีการเกิดในตำแหน่งต่าง ๆ ที่กระจายอยู่ทั่วโลกมีการเกิดในเวลาที่แตกต่างกันไป รวมไปถึงยังมีสภาพอากาศและจะมีการกำหนดเกิดในตำแหน่งพิเศษ ซึ่งเป็นส่วนหนึ่งของปัจจัยที่ส่งผลต่อชนิดและความหายากของโปเกมอนที่พบเจอ หากเราสามารถตรวจสอบเงื่อนไขได้ว่า ช่วงเวลา สภาพภูมิอากาศ และสถานที่ ที่โปเกมอนจะเกิดนั้นพบโปเกมอนหายากได้หรือไม่จะสามารถวางแผนได้อย่างถูกต้อง และรัดกุมมากยิ่งขึ้น ซึ่งจะเป็นการช่วยเหลือผู้เล่นให้สามารถสะสมโปเกมอนหายากได้มากขึ้นด้วยเวลาที่น้อยลง

ทั้งนี้ทางคณะผู้จัดทำจึงได้รวบรวมและสร้างแบบจำลองข้อมูลที่สามารถจำลองพร้อมกับทำนายได้ว่าหากจับโปเกมอนในเงื่อนไขหนึ่งจะสามารถได้โปเกมอนที่หายากหรือไม่ เพื่อให้ผู้เล่นสามารถนำข้อมูลที่ได้ไปใช้จับโปเกมอนที่มีความหายากต่าง ๆ โปเกมอนที่เกิดในพื้นที่เฉพาะและสามารถเพิ่มโอกาสการพบเจอโปเกมอนหายากได้ต่อไป

Data Explaination

1. **Dataset การปรากฏตัวของโปเกมอนในเกม Pokemon Go ในปี 2016**

   โดยจะมีรายละเอียดในแต่ละ column ที่สนใจนำมาใช้งาน ดังนี้

   - pokemonid ตัวเลขระบุโปเกมอนมีค่าตั้งแต่ 1-151 (Pokemon Generation 1)
   - latitude ค่าละติจูด
   - longitude ค่าลองจิจูด
   - appearedLocalTime เวลาที่ปรากฏตามเวลาท้องถิ่น
   - appearedTimeOfDay ช่วงเวลาที่ปรากฏ เช่น night, evening, afternoon, morning
   - terrainType รูปแบบภูมิประเทศ อ้างอิงจากดัชนีจำแนกสิ่งที่ปกคลุมหน้าดิน (IGBP)
   - closeToWater อยู่ใกล้แหล่งน้ำหรือไม่
   - city ชื่อเมือง
   - continent ชื่อทวีป
   - weather สภาพภูมิอากาศ เช่น Foggy Clear, PartlyCloudy, MostlyCloudy, Overcast
   - temperature อุณหภูมิ
   - windSpeed ความเร็วลม
   - windBearing ทิศทางลม
   - pressure แรงกดอากาศ(บาร์)
   - weatherIcon สภาพภูมิอากาศที่ตัวเกมแจ้ง
   - population density ความหนาแน่นประชากร ต่อ 1 ตารางกม.
   - gymDistanceKm ระยะห่างจาก pokemon gym (กม.)
   - pokestopDistanceKm ระยะห่างจาก pokestop (กม.)

2. **Dataset ข้อมูล Pokemon ตาม pokemonid** เนื่องจาก Dataset
   แรกไม่ได้มีการระบุชื่อของตัวโปเกมอนที่พบเจอจึงมีการนำ Dataset นี้มาช่วยในการระบุค่าดังนี้
   - Number ตัวเลขระบุโปเกมอน
   - Name ชื่อของโปเกมอน
   - Type 1 ธาตุของโปเกมอน
   - Type 2 ธาตุที่ 2 ของโปเกมอน (หากมี)

3. **Data Pokemon ระดับความหายากของโปเกมอนแต่ล่ะตัวในเกม Pokemon GO**
   - Number ตัวเลขระบุโปเกมอน
   - Rarity ระดับความหายากของโปเกมอน เช่น Common, Uncommon, Rare, Super rare, Very rare,

Data preparation process and results

Import Data

นำเข้าข้อมูล การปรากฏตัวของโปเกมอนในพื้นที่ต่างๆ ของเกม Pokemon Go

```
[1041] import pandas as pd

[1042] from google.colab import drive
       drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

นำเข้าข้อมูล การปรากฏตัวของโปเกมอนในเกม Pokemon Go

       df = pd.read_csv('/content/drive/MyDrive/300k_csv/300k.csv')

<ipython-input-1043-efe82204873c>:1: DtypeWarning: Columns (49) have mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv('/content/drive/MyDrive/300k_csv/300k.csv')
```

หลังจากที่นำ data เข้ามาก็ทำการเลือก columns ในส่วนที่สนใจและคาดว่าเป็นประโยชน์

```
selected_columns = ['pokemonId', 'latitude', 'longitude', 'appearedLocalTime', 'appearedTimeOfDay', 'terrainType', 'closeToWater', 'city', 'continent', 'weather', 'temperature', 'windSpeed', 'windBearing', 'pressure', 'weatherIcon', 'population_density', 'gymDistanceKm', 'pokestopDistanceKm']
pokemon_df = df.loc[:, selected_columns]
pokemon_df.head()
```

	pokemonId	latitude	longitude	appearedLocalTime	appearedTimeOfDay	terrainType	closeToWater	city	continent	weather	temperature	windSpeed	windBearing	pressure	weatherIcon	population_density	gymDistanceKm	pokestopDistanceKm
0	16	20.525745	-97.460829	2016-09-08T03:57:45	night	14	False	Mexico_City	America	Foggy	25.5	4.79	269	1018.02	fog	2431.2341	0.049869	0.081776
1	133	20.523695	-97.461167	2016-09-08T03:57:37	night	14	False	Mexico_City	America	Foggy	25.5	4.79	269	1018.02	fog	2431.2341	0.259156	0.195622
2	16	38.903590	-77.199780	2016-09-08T03:57:25	night	13	False	New_York	America	Clear	24.2	4.29	218	1015.29	clear-night	761.8856	0.489886	0.338602
3	13	47.665903	-122.312561	2016-09-08T03:56:22	night	0	True	Los_Angeles	America	PartlyCloudy	15.6	5.84	160	1020.52	partly-cloudy-night	4842.1626	0.359309	0.109479
4	133	47.666454	-122.311628	2016-09-08T03:56:08	night	0	True	Los_Angeles	America	PartlyCloudy	15.6	5.84	160	1020.52	partly-cloudy-night	4842.1626	0.210543	0.040364

นำข้อมูลของ Pokemon เข้าตาม pokemonid

```
ข้อมูล Pokemon ตาม pokemonid

[ ] df = pd.read_csv('/content/drive/MyDrive/300k_csv/All_Pokemon.csv')

df.head()
```

	Number	Name	Type 1	Type 2	Abilities	HP	Att	Def	Spa	Spd	...	Against Bug	Against Rock	Against Ghost	Against Dragon	Against Dark	Against Steel	Against Fairy	Height	Weight	BMI
0	1	Bulbasaur	Grass	Poison	['Chlorophyll', 'Overgrow']	45	49	49	65	65	...	1.0	1.0	1.0	1.0	1.0	1.0	0.5	0.7	6.9	14.1
1	2	Ivysaur	Grass	Poison	['Chlorophyll', 'Overgrow']	60	62	63	80	80	...	1.0	1.0	1.0	1.0	1.0	1.0	0.5	1.0	13.0	13.0
2	3	Venusaur	Grass	Poison	['Chlorophyll', 'Overgrow']	80	82	83	100	100	...	1.0	1.0	1.0	1.0	1.0	1.0	0.5	2.0	100.0	25.0
3	3	Mega Venusaur	Grass	Poison	['Thick Fat']	80	100	123	122	120	...	1.0	1.0	1.0	1.0	1.0	1.0	0.5	2.4	155.5	27.0
4	4	Charmander	Fire	NaN	['Blaze', 'Solar Power']	39	52	43	60	50	...	0.5	2.0	1.0	1.0	1.0	0.5	0.5	0.6	8.5	23.6

5 rows × 44 columns

เนื่องจากใน dataset แรกของเรามีเลข pokemonid อยู่เพียง 151 ตัวหรือแค่ pokemon generation 1 เท่านั้น แล้วจึงเลือก columns ในส่วนที่สนใจและคาดว่าเป็นประโยชน์ของ pokemon generation 1

```
[ ] df = df[df['Generation'] == 1]
```

```
selected_columns = ['Number','Name','Type 1','Type 2']
pokedex_df = df.loc[:, selected_columns]
pokedex_df.head()
```

	Number	Name	Type 1	Type 2
0	1	Bulbasaur	Grass	Poison
1	2	Ivysaur	Grass	Poison
2	3	Venusaur	Grass	Poison
4	4	Charmander	Fire	NaN
5	5	Charmeleon	Fire	NaN

Next steps: View recommended plots

ทำการเปลี่ยนชื่อ column จาก Number ไปเป็น pokemonId เพื่อนำไปรวมกับ dataset

```
new_column_names = {
    'Number': 'pokemonId',
}
pokedex_df = pokedex_df.rename(columns=new_column_names)
pokedex_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 151 entries, 0 to 194
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   pokemonId  151 non-null    int64
 1   Name       151 non-null    object
 2   Type 1     151 non-null    object
 3   Type 2     67 non-null     object
dtypes: int64(1), object(3)
memory usage: 5.9+ KB
```

```
[ ] pokemon_df = pd.merge(pokemon_df, pokedex_df, on='pokemonId',how='left')
```

นำเข้าข้อมูล ระดับความหายากของโปเกมอน ตาม pokemonid เข้าไปรวมใน dataset แรก

```
pokemon_rarity = {
    1: 'Rare', 2: 'Rare', 3: 'Rare', 4: 'Rare', 5: 'Rare', 6: 'Rare',
    7: 'Rare', 8: 'Rare', 9: 'Rare', 10: 'Uncommon', 11: 'Uncommon',
    12: 'Uncommon', 13: 'Common', 14: 'Common', 15: 'Common', 16: 'Common',
    17: 'Common', 18: 'Common', 19: 'Common', 20: 'Common', 21: 'Uncommon',
    22: 'Uncommon', 23: 'Uncommon', 24: 'Uncommon', 25: 'Rare', 26: 'Rare',
    27: 'Rare', 28: 'Rare', 29: 'Uncommon', 30: 'Uncommon', 31: 'Uncommon',
    32: 'Uncommon', 33: 'Uncommon', 34: 'Uncommon', 35: 'Rare', 36: 'Rare',
    37: 'Rare', 38: 'Rare', 39: 'Rare', 40: 'Rare', 41: 'Common', 42: 'Common',
    43: 'Uncommon', 44: 'Uncommon', 45: 'Uncommon', 46: 'Uncommon', 47: 'Uncommon',
    48: 'Uncommon', 49: 'Uncommon', 50: 'Rare', 51: 'Rare', 52: 'Rare', 53: 'Rare',
    54: 'Rare', 55: 'Rare', 56: 'Rare', 57: 'Rare', 58: 'Uncommon', 59: 'Uncommon',
    60: 'Uncommon', 61: 'Uncommon', 62: 'Uncommon', 63: 'Rare', 64: 'Rare', 65: 'Rare',
    66: 'Rare', 67: 'Rare', 68: 'Rare', 69: 'Uncommon', 70: 'Uncommon', 71: 'Uncommon',
    72: 'Rare', 73: 'Rare', 74: 'Rare', 75: 'Rare', 76: 'Rare', 77: 'Rare', 78: 'Rare',
    79: 'Rare', 80: 'Rare', 81: 'Rare', 82: 'Rare', 83: 'Super Rare', 84: 'Uncommon',
    85: 'Uncommon', 86: 'Rare', 87: 'Rare', 88: 'Very Rare', 89: 'Very Rare', 90: 'Rare',
    91: 'Rare', 92: 'Uncommon', 93: 'Uncommon', 94: 'Uncommon', 95: 'Rare', 96: 'Rare',
    97: 'Uncommon', 98: 'Rare', 99: 'Uncommon', 100: 'Rare', 101: 'Rare', 102: 'Rare',
    103: 'Rare', 104: 'Rare', 105: 'Rare', 106: 'Very Rare', 107: 'Very Rare', 108: 'Very Rare',
    109: 'Rare', 110: 'Rare', 111: 'Rare', 112: 'Rare', 113: 'Very Rare', 114: 'Rare',
    115: 'Super Rare', 116: 'Rare', 117: 'Rare', 118: 'Rare', 119: 'Rare', 120: 'Uncommon',
    121: 'Uncommon', 122: 'Super Rare', 123: 'Rare', 124: 'Rare', 125: 'Rare', 126: 'Rare',
    127: 'Uncommon', 128: 'Rare', 129: 'Very Rare', 130: 'Very Rare', 131: 'Super Rare',
    132: 'Super Rare', 133: 'Rare', 134: 'Rare', 135: 'Rare', 136: 'Rare', 137: 'Very Rare',
    138: 'Rare', 139: 'Rare', 140: 'Rare', 141: 'Rare', 142: 'Very Rare', 143: 'Rare',
    144: 'Super Rare', 145: 'Super Rare', 146: 'Super Rare', 147: 'Rare', 148: 'Rare',
    149: 'Rare', 150: 'Super Rare', 151: 'Super Rare'
}
```

```
[ ] pokemon_df['rarity'] = pokemon_df['pokemonId'].map(pokemon_rarity)
```

สุดท้ายจะได้ dataset ออกมาในรูปแบบดังกล่าว

```
pokemon_df.head(5)
```

	pokemonId	latitude	longitude	appearedLocalTime	appearedTimeOfDay	terrainType	closeToWater	city	continent	weather	...	windBearing	pressure	weatherIcon	population_density	gymDistanceKm	pokestopDistanceKm	Name	Type 1	Type 2	rarity
0	16	20.525745	-97.460829	2016-09-08T03:57:45	night	14	False	Mexico_City	America	Foggy	...	269	1018.02	fog	2431.2341	0.049869	0.081776	Pidgey	Normal	Flying	Common
1	133	20.523695	-97.461167	2016-09-08T03:57:37	night	14	False	Mexico_City	America	Foggy	...	269	1018.02	fog	2431.2341	0.259156	0.195622	Eevee	Normal	NaN	Rare
2	16	38.903590	-77.199780	2016-09-08T03:57:25	night	13	False	New_York	America	Clear	...	218	1015.29	clear-night	761.8856	0.489886	0.338602	Pidgey	Normal	Flying	Common
3	13	47.665903	-122.312561	2016-09-08T03:56:22	night	0	True	Los_Angeles	America	PartlyCloudy	...	160	1020.52	partly-cloudy-night	4842.1626	0.359309	0.109479	Weedle	Bug	Poison	Common
4	133	47.666454	-122.311628	2016-09-08T03:56:08	night	0	True	Los_Angeles	America	PartlyCloudy	...	160	1020.52	partly-cloudy-night	4842.1626	0.210543	0.040364	Eevee	Normal	NaN	Rare

5 rows × 22 columns

Data Cleaning

เนื่องจาก data type ของ appearedLocalTime เป็น object จึงทำการเปลี่ยนให้เป็น datetime เพื่อให้ง่ายต่อการนำไปใช้งาน

```
[1347] pokemon_df['appearedLocalTime'] = pd.to_datetime(pokemon_df['appearedLocalTime'], format='%Y-%m-%dT%H:%M:%S')
```

```
pokemon_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 296021 entries, 0 to 296020
Data columns (total 22 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   pokemonId           296021 non-null  int64
 1   latitude            296021 non-null  float64
 2   longitude           296021 non-null  float64
 3   appearedLocalTime   296021 non-null  datetime64[ns]
 4   appearedTimeOfDay   296021 non-null  object
 5   terrainType         296021 non-null  int64
 6   closeToWater        296021 non-null  bool
 7   city                296021 non-null  object
 8   continent           296021 non-null  object
 9   weather             296021 non-null  object
 10  temperature         296021 non-null  float64
 11  windSpeed           296021 non-null  float64
 12  windBearing         296021 non-null  int64
 13  pressure            296021 non-null  float64
 14  weatherIcon         296021 non-null  object
 15  population_density  296021 non-null  float64
 16  gymDistanceKm       296021 non-null  float64
 17  pokestopDistanceKm  296021 non-null  object
 18  Name                296021 non-null  object
 19  Type 1              296021 non-null  object
 20  Type 2              151211 non-null  object
 21  rarity              296021 non-null  object
dtypes: bool(1), datetime64[ns](1), float64(7), int64(3), object(10)
memory usage: 47.7+ MB
```

ตรวจสอบค่าหา null ในแต่ละ column และพบว่ามีเพียง column ‘Type 2’ ที่มีค่าเป็น null อยู่ ซึ่งสามารถยอมรับได้เนื่องจากโปเกมอนบางตัวมีเพียงธาตุเดียว

```
pokemon_df.isnull().any()
```

```
pokemonId             False
latitude              False
longitude             False
appearedLocalTime     False
appearedTimeOfDay     False
terrainType           False
closeToWater          False
city                  False
continent             False
weather               False
temperature           False
windSpeed             False
windBearing           False
pressure              False
weatherIcon           False
population_density    False
gymDistanceKm         False
pokestopDistanceKm    False
Name                  False
Type 1                False
Type 2                 True
rarity                False
dtype: bool
```

ตรวจสอบแถวที่มีค่าซ้ำกัน และหลังจากตรวจสอบแล้วไม่พบแถวที่มีค่าซ้ำกัน

```
[1350] pokemon_df[pokemon_df.duplicated()].head()
```

	pokemonId	latitude	longitude	appearedLocalTime	appearedTimeOfDay	terrainType	closeToWater	city	continent	weather	...	windBearing	pressure	weatherIcon	population_density	gymDistanceKm	pokestopDistanceKm	Name	Type 1	Type 2	rarity

0 rows × 22 columns

เนื่องจาก datatype ของ pokestopDistanceKm เป็น object ซึ่งควรเป็น float จึงทำการตรวจสอบหาค่าที่ไม่ใช่ตัวเลข

```
non_numeric_values = pokemon_df.loc[~pokemon_df['pokestopDistanceKm'].apply(pd.to_numeric, errors='coerce').notna(), 'pokestopDistanceKm']

print(non_numeric_values)
```

```
24570     ?
33459     ?
45025     ?
47999     ?
48090     ?
60374     ?
61009     ?
63813     ?
64236     ?
64515     ?
66526     ?
67702     ?
67995     ?
68305     ?
75340     ?
75836     ?
94313     ?
94440     ?
132945    ?
133599    ?
136223    ?
142960    ?
143493    ?
146630    ?
147084    ?
161321    ?
167224    ?
206038    ?
210427    ?
221327    ?
221580    ?
227290    ?
227315    ?
231451    ?
261021    ?
272505    ?
273387    ?
273987    ?
284535    ?
Name: pokestopDistanceKm, dtype: object
```

จากการตรวจสอบพบค่าที่ไม่ใช่ตัวเลขซึ่งเป็นส่วนน้อยเทียบกับจำนวนแถวทั้งหมด จึงทำการ drop แถวนั้นทิ้งและเปลี่ยน data type ให้เป็น float

```
[1352] pokemon_df.drop(pokemon_df[pokemon_df['pokestopDistanceKm'] == '?'].index, inplace=True)

[1353] pokemon_df['pokestopDistanceKm'] = pokemon_df['pokestopDistanceKm'].astype(float)
```

เนื่องจาก Terrian type แสดงค่าเป็นตัวเลขแทนภูมิประเทศในแต่ล่ะแบบ

ทำให้ยากต่อการระบุจึงเปลี่ยนค่าที่อธิบายจากตัวเลขเป็นลักษณะภูมิประเทศ

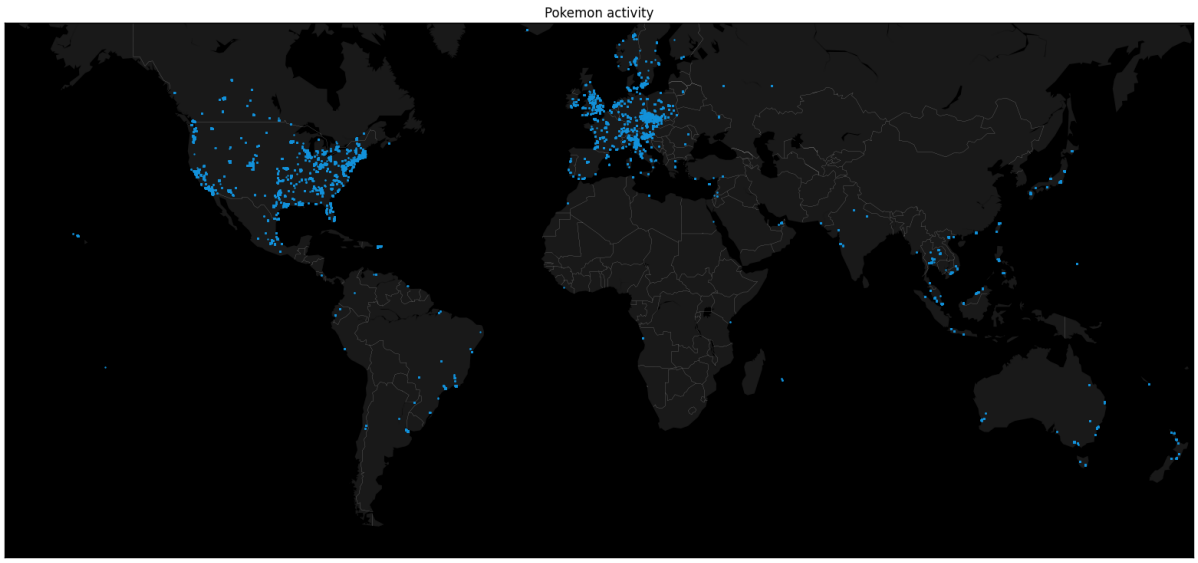
```
terrain_mapping = {
    0: 'Water Bodies',
    1: 'Evergreen Needleleaf Forest',
    2: 'Evergreen Broadleaf Forest',
    3: 'Deciduous Needleleaf Forest',
    4: 'Deciduous Broadleaf Forest',
    5: 'Mixed Forest',
    6: 'Closed Shrublands',
    7: 'Open Shrublands',
    8: 'Woody Savannas',
    9: 'Savannas',
    10: 'Grasslands',
    11: 'Permanent Wetlands',
    12: 'Croplands',
    13: 'Urban and Built-up',
    14: 'Cropland/Natural Vegetation Mosaics',
    15: 'Permanent Snow and Ice',
    16: 'Barren'
}

pokemon_df['terrainType'] = pokemon_df['terrainType'].map(terrain_mapping)
```

EDA and Visualization of data

1. Encounter Location of Dataset (All time)

ต้องการทราบการปรากฏของโปเกมอนในชุดข้อมูลบนแผนที่โลก

```
plt.figure(1, figsize=(20,10))
m1 = Basemap(projection='merc',
             llcrnrlat=-60,
             urcrnrlat=65,
             llcrnrlon=-180,
             urcrnrlon=180,
             lat_ts=0,
             resolution='c')

m1.fillcontinents(color='#191919',lake_color='#000000') # dark grey land, black lakes
m1.drawmapboundary(fill_color='#000000')                # black background
m1.drawcountries(linewidth=0.1, color="w")              # thin white line for country borders


x, y = m1(pokemon_df.longitude.tolist(),pokemon_df.latitude.tolist())
m1.scatter(x,y, s=3, c="#1292db", lw=0, alpha=1, zorder=5)
plt.title("Pokemon activity")
plt.show()
```



จากภาพ จะเห็นได้ว่าข้อมูลส่วนใหญ่ของการจับโปเกม่อนใน dataset นี้จะมาจากทวีปยุโรปกับสหรัฐอเมริกาเป็นหลัก ถ้านอกจากยุโรปแล้วจะอยู่ประเทศและเมืองที่มีคนท่องเที่ยวอยู่บ้าง เช่น ญี่ปุ่น ออสเตเรีย ไทย มาเลเซีย บราซิล เม็กซิโก UAE ที่น่าสนใจเพิ่มเติมคือ ในจีนมีการจับโปเกม่อนแค่จุดเดียว นั่นเพราะสาเหตุคือจีนแบน Service ของ Google Map ซึ่ง Pokemon Go ใช้อ้างอิง แต่มีโซนเล็ก ๆ ที่ยังคงสามารถจับได้เพราะผ่านการโดนแบน

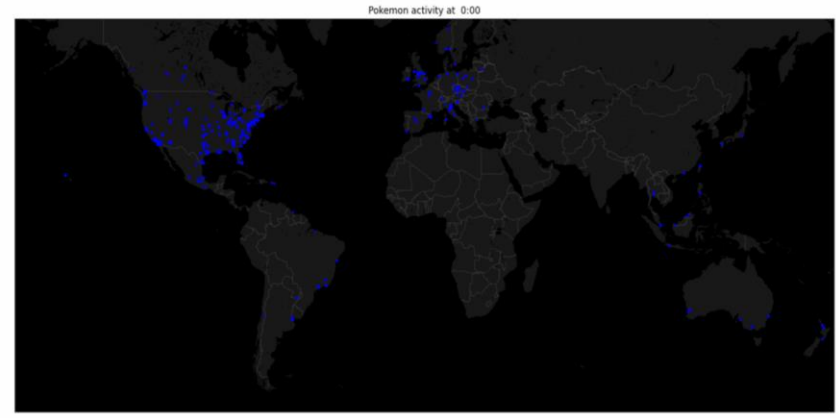
2. Encounter Location of Dataset (by hour)

ต้องการทราบการปรากฏของโปเกมอนในเกมเทียบกับแผนที่โลกในแต่ละชม.ตามเวลาท้องถิ่น

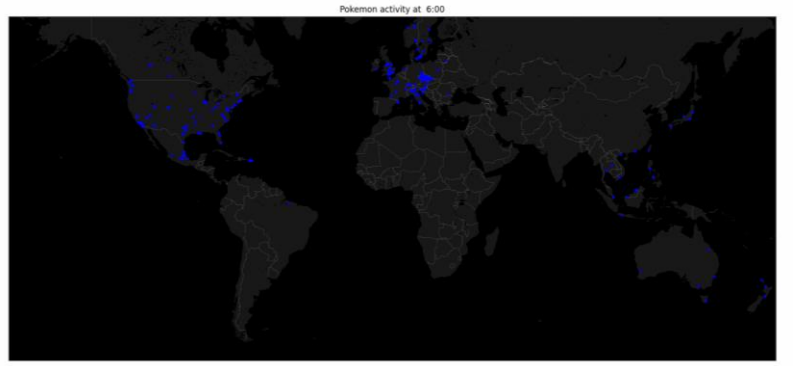
```
[30] pokemon_df['appearedHour'] = pokemon_df['appearedLocalTime'].dt.hour
```

```
time_groups = pokemon_df.groupby('appearedHour')
plt.figure(1, figsize=(20,10))
m = map = Basemap(
    projection='merc',
    llcrnrlat=-60,
             urcrnrlat=65,
             llcrnrlon=-180,
             urcrnrlon=180,
             lat_ts=0,
    resolution='i')
m.fillcontinents(color='#191919',lake_color='#000000') # dark grey land, black lakes
m.drawmapboundary(fill_color='#000000')                # black background
m.drawcountries(linewidth=0.1, color="w")              # thin white line for country borders

x,y = m(0, 0)
point = m.plot(x, y, 'o', markersize=2, color='b')[0]
def init():
    point.set_data([], [])
    return point,

def animate(i):
    lon = time_groups.get_group(i)['longitude'].values
    lat = time_groups.get_group(i)['latitude'].values
    x, y = m(lon ,lat)
    point.set_data(x,y)
    plt.title('Pokemon activity at %2d:00' % (i))
    return point,

output = animation.FuncAnimation(plt.gcf(), animate, init_func=init, frames=24, interval=500, blit=True, repeat=False)
output.save("pokemon.gif", writer='imagemagick')
plt.show()
```

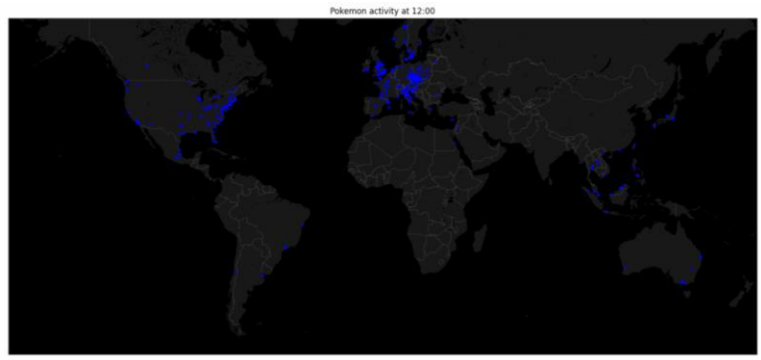
Time 00:00



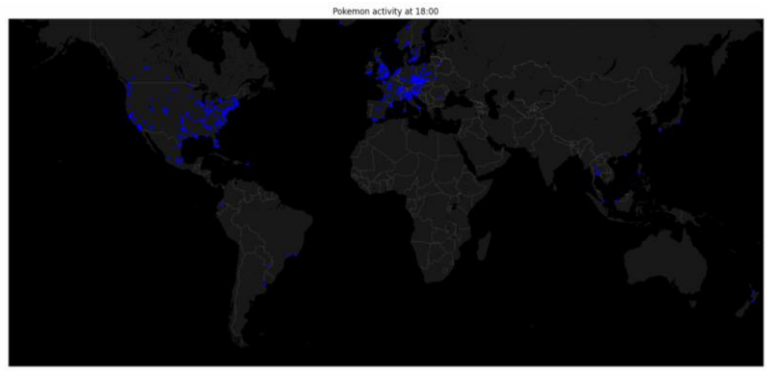
Time 06:00



Time 12:00



Time 18:00



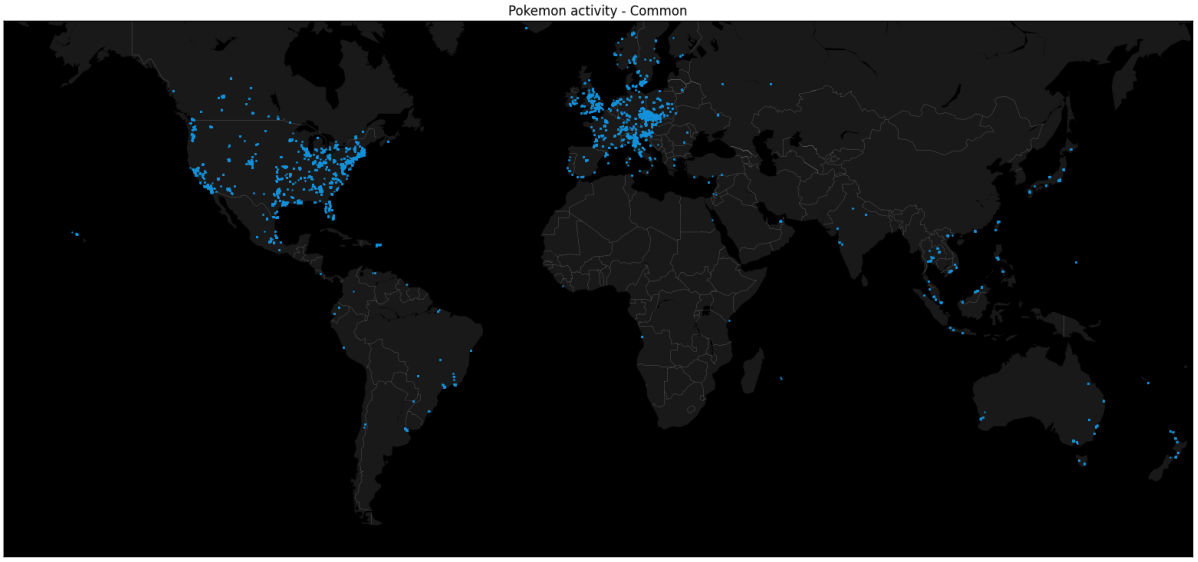
จากภาพคือภาพตัวอย่างที่มาจาก Gif แสดงการจับโปเกม่อนโดยแต่ละช่วงเวลา ที่ได้สร้างขึ้น จะเห็นได้ว่าในช่วงเวลาประมาณ 18 นาฬิกาถึงเที่ยงคืนจะมีรายงานพบโปเกมอนที่เริ่มจะลดลงจากยุโรปแล้วไปชุกชุมอยู่ที่สหรัฐอเมริกา เมื่อพอช่วงเที่ยงคืนถึง 6 นาฬิกาความหนาแน่นจะเริ่มเพิ่มขึ้นในยุโรปและลดลงในอเมริกา ในช่วง 6 นาฬิกาถึงเที่ยงวันความหนาแน่นจะเพิ่มขึ้นในทั้งยุโรปและอเมริกา รวมถึงเอเชียก็เริ่มหนาแน่นขึ้นเช่นกัน และในช่วงเที่ยงวันถึง 18 นาฬิกาความหนาแน่นก็จะเพิ่มขึ้นเล็กน้อยทั้งยุโรปและสหรัฐอเมริกา แต่ในเอเชียจะลดลงไป

3. Rarity by location

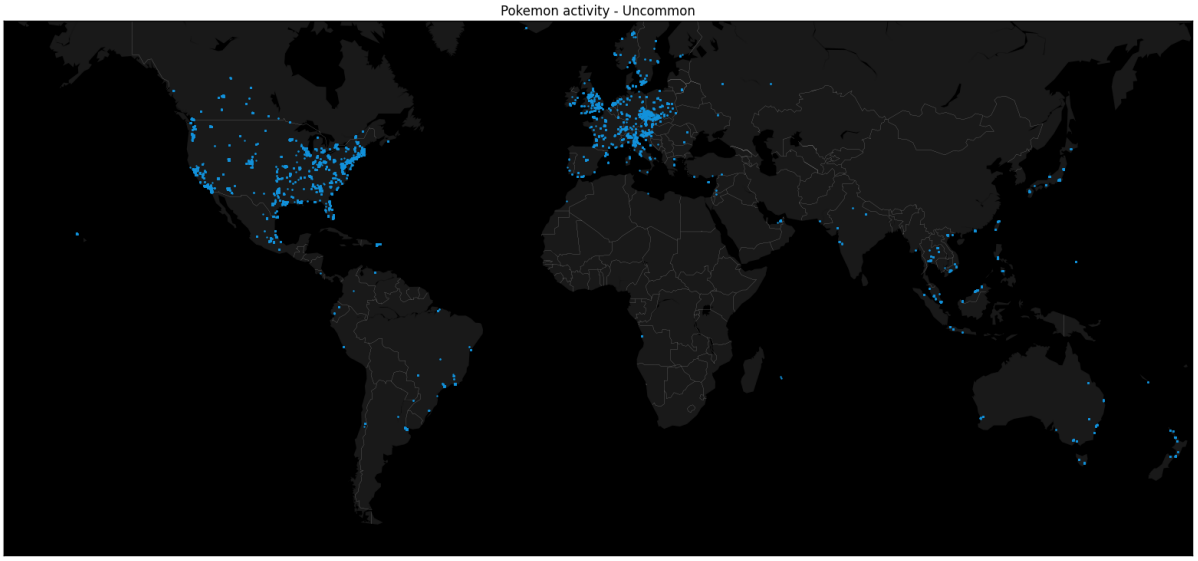
สร้างแผนภาพแผนที่โลกตำแหน่งการพบโปเกมอนในแต่ละระดับความหายาก

```
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap

# Assuming 'pokemon_df' is your DataFrame and 'rarity' is a column in it

rarities = ["Common", "Rare", "Uncommon", "Super Rare", "Very Rare"]

# Create a separate map for each rarity
for rarity in rarities:
    plt.figure(figsize=(20, 10))
    m = Basemap(projection='merc',
                llcrnrlat=-60,
                urcrnrlat=65,
                llcrnrlon=-180,
                urcrnrlon=180,
                lat_ts=0,
                resolution='c')

    m.fillcontinents(color='#191919', lake_color='#000000') # dark grey land, black lakes
    m.drawmapboundary(fill_color='#000000')                 # black background
    m.drawcountries(linewidth=0.1, color="w")               # thin white line for country borders

    # Filter DataFrame by rarity
    filtered_df = pokemon_df[pokemon_df['rarity'] == rarity]

    x, y = m(filtered_df.longitude.tolist(), filtered_df.latitude.tolist())
    m.scatter(x, y, s=3, c="#1292db", lw=0, alpha=1, zorder=5)
    plt.title(f"Pokemon activity - {rarity}")
    plt.show()
```

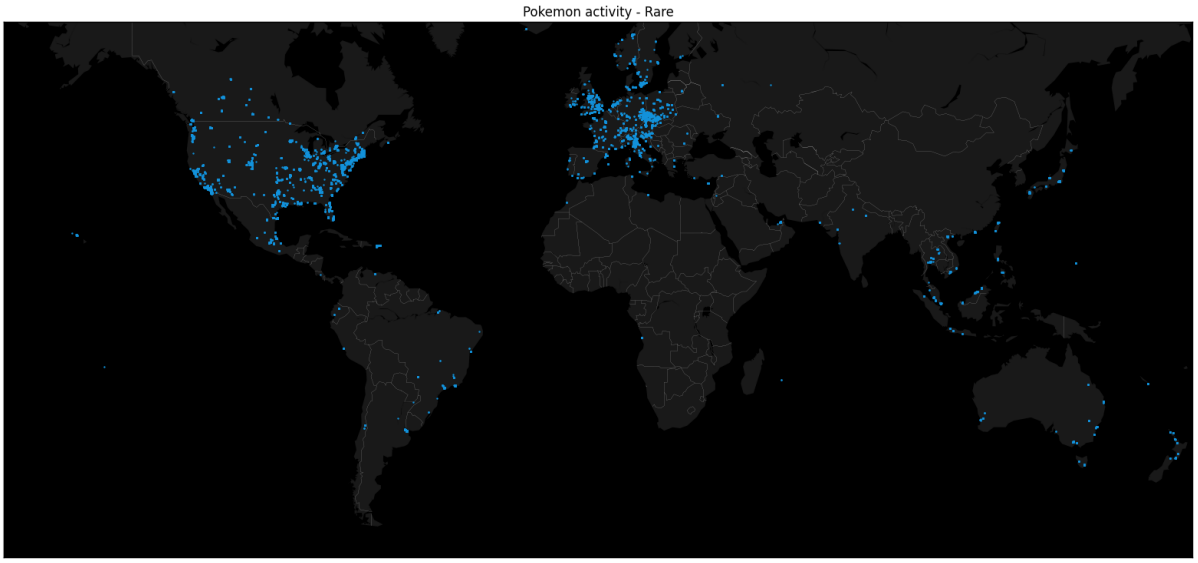
ระดับ Common



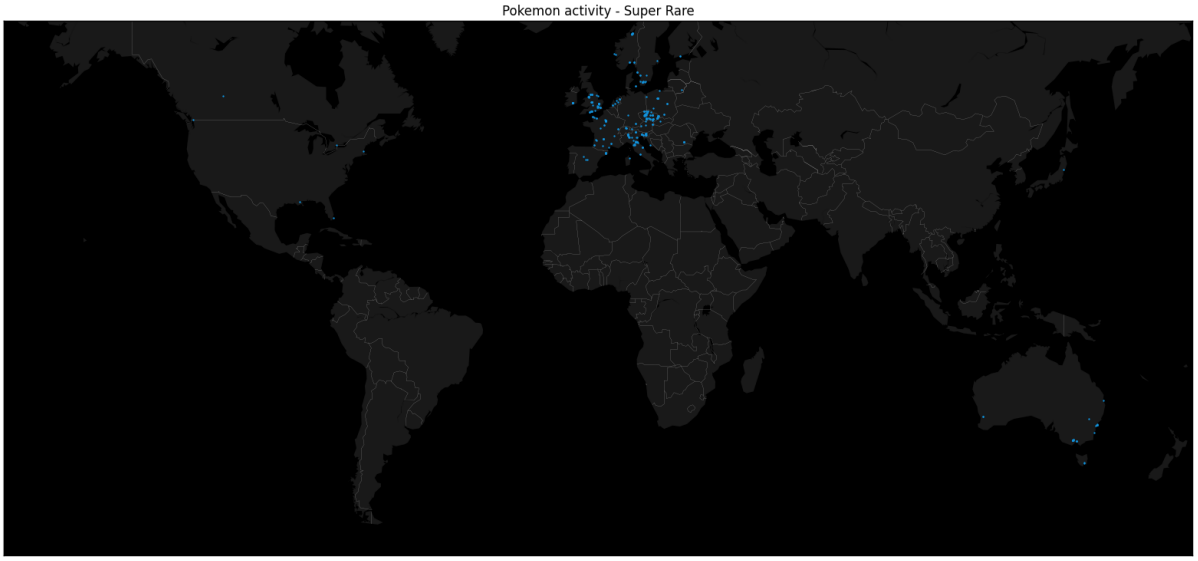
ระดับ Uncommon



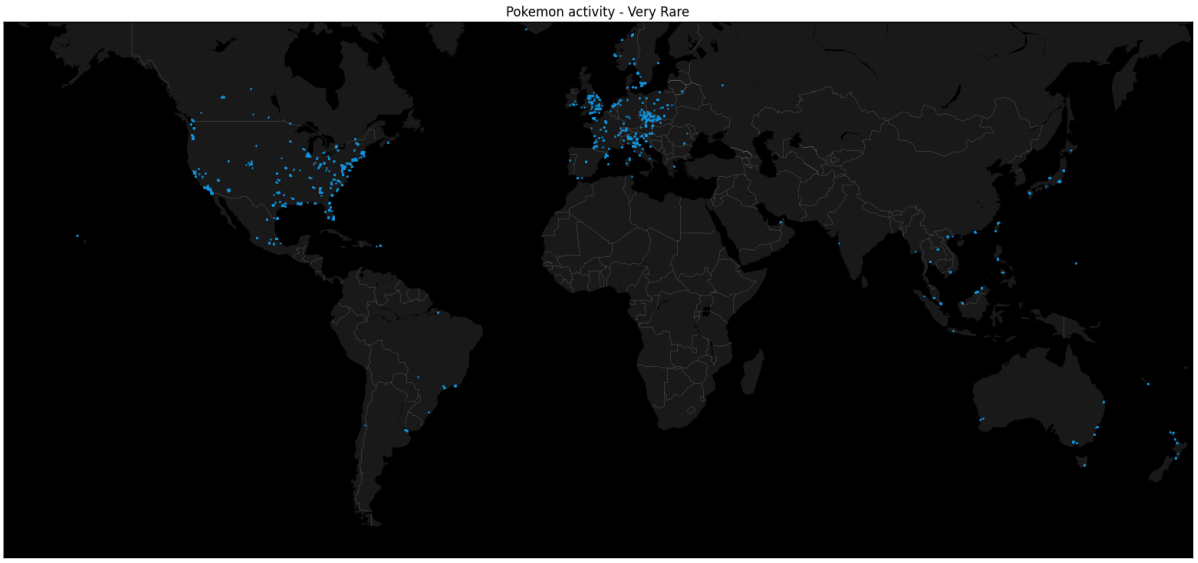
ระดับ Rare



ระดับ Super rare



ระดับ Very rare



จากภาพ จะเห็นได้ว่ามีการกระจายตัวการพบโปเกมอนตามทวีปยุโรปกับสหรัฐอเมริกาเป็นหลักถ้านอกจากยุโรปแล้ว จะอยู่ประเทศและเมืองที่มีคนท่องเที่ยวอยู่บ้าง เช่น ญี่ปุ่น ออสเตเรีย ไทย มาเลเซีย บราซิล เม็กซิโก UAE โดยในแต่ละระดับความหายากของโปเกมอน Common, Uncommon, Rare, Very Rare ก็จะมีการพบเจอมากไปน้อยตามลำดับแต่จะมีโปเกมอนที่ระดับความหายาก Super Rare ที่มีการพบเจอน้อยที่สุดอยู่แค่เพียงบริเวณทวีปยุโรปเพียงอย่างเดียวซึ่งทั้งนี้การที่โปเกมอนที่ระดับความหายาก Super Rare มีการพบเจอได้ยากกว่าระดับ Very Rare อาจจะเป็นเพราะเวอร์ชชั่นเกม ณ ขณะนั้นไม่ได้มีโปเกมอนที่ระดับความหายาก Super Rare ที่หลากหลายจึงทำให้มีการพบเจอได้ยากกว่าระดับ Very Rare

4. Pokemon type by weather

ต้องการทราบความสัมพันธ์ของสภาพอากาศว่ามีผลต่อการปรากฏตัวของโปเกมอนในแต่ละธาตุหรือไม่ เริ่มจากทำการนับธาตุของโปเกมอนที่พบเจอในสภาพอากาศแต่ละแบบ เนื่องด้วยโปเกมอนบางตัวมีสองธาตุ จึงใช้วิธีการนับโปเกมอนจากธาตุที่ 1 แล้วบวกรวมกับการนับโปเกมอนจากธาตุที่ 2

```
weather_counts_by_type_1 = pokemon_df.pivot_table(index='weather', columns='Type 1', aggfunc='size', fill_value=0)
```

```
weather_counts_by_type_2 = pokemon_df.pivot_table(index='weather', columns='Type 2', aggfunc='size', fill_value=0)
```

```
weather_counts_by_type_combine = weather_counts_by_type_1.add(weather_counts_by_type_2, fill_value=0)
```

```
weather_counts_by_type_combine.head(10)
```

weather	Bug	Dragon	Electric	Fairy	Fighting	Fire	Flying	Ghost	Grass	Ground	Ice	Normal	Poison	Psychic	Rock	Steel	Water
Breezy	140.0	1.0	62.0	13	10	21.0	223.0	9.0	43	21	1	352.0	223	52	18	25.0	214
BreezyandMostlyCloudy	8.0	0.0	2.0	1	1	0.0	24.0	0.0	1	1	0	28.0	10	5	0	1.0	24
BreezyandOvercast	33.0	0.0	13.0	2	0	3.0	42.0	1.0	18	0	0	64.0	63	10	0	3.0	66
BreezyandPartlyCloudy	121.0	0.0	4.0	10	0	8.0	147.0	5.0	34	3	0	263.0	150	30	1	2.0	43
Clear	32451.0	385.0	1600.0	2833	1862	3055.0	46265.0	1254.0	11525	5450	397	73997.0	42812	8940	3044	580.0	20257
DangerouslyWindy	20.0	0.0	1.0	0	0	0.0	29.0	0.0	8	0	0	50.0	16	0	1	0.0	11
Drizzle	959.0	11.0	30.0	78	14	21.0	1256.0	42.0	244	31	12	2015.0	1102	237	23	9.0	480
DrizzleandBreezy	21.0	0.0	1.0	0	0	3.0	49.0	8.0	7	1	2	83.0	31	14	0	0.0	23
Dry	190.0	0.0	5.0	16	8	7.0	248.0	11.0	57	26	3	338.0	249	68	16	2.0	119
DryandMostlyCloudy	7.0	0.0	0.0	1	0	0.0	22.0	0.0	4	0	0	34.0	11	1	0	0.0	8

เนื่องจากข้อมูลมีระยะห่างมากเกินไปจึงทำการ normalize โดยหารด้วยผลรวมของทั้งแถว

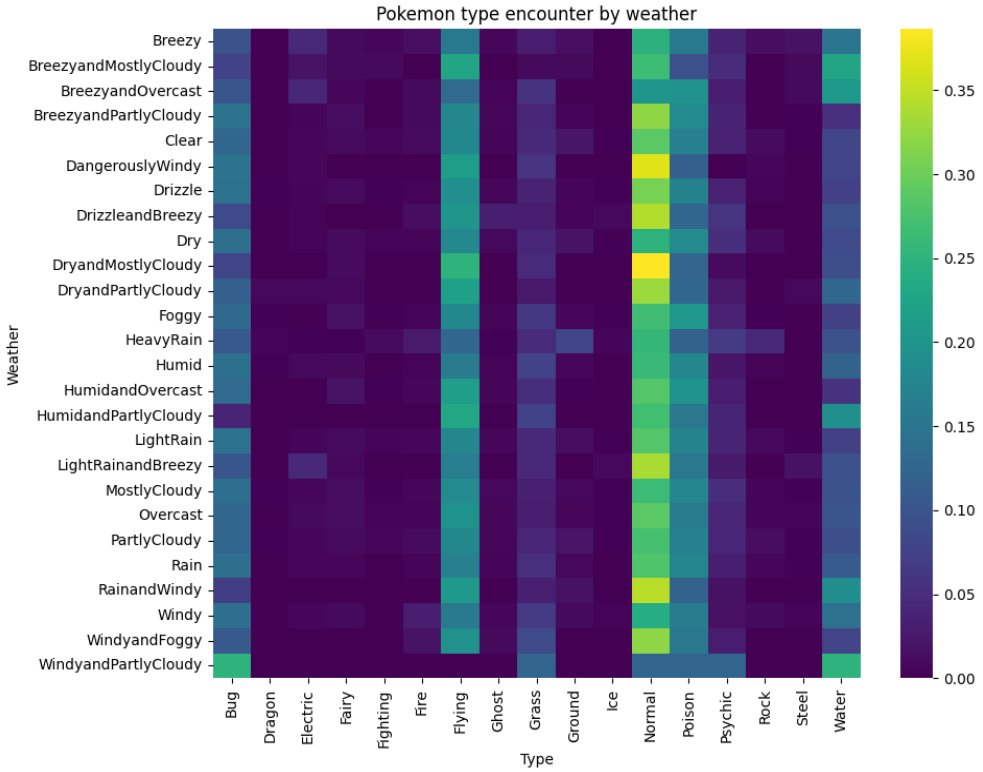
```
weather_counts_by_type_normalize = weather_counts_by_type_combine.div(weather_counts_by_type_combine.sum(axis=1), axis=0)
```

```
weather_counts_by_type_normalize.head(10)
```

weather	Bug	Dragon	Electric	Fairy	Fighting	Fire	Flying	Ghost	Grass	Ground	Ice	Normal	Poison	Psychic	Rock	Steel	Water
Breezy	0.098039	0.000700	0.043417	0.009104	0.007003	0.014706	0.156162	0.006303	0.030112	0.014706	0.000700	0.246499	0.156162	0.036415	0.012605	0.017507	0.149860
BreezyandMostlyCloudy	0.075472	0.000000	0.018868	0.009434	0.009434	0.000000	0.226415	0.000000	0.009434	0.009434	0.000000	0.264151	0.094340	0.047170	0.000000	0.009434	0.226415
BreezyandOvercast	0.103774	0.000000	0.040881	0.006289	0.000000	0.009434	0.132075	0.003145	0.056604	0.000000	0.000000	0.201258	0.198113	0.031447	0.000000	0.009434	0.207547
BreezyandPartlyCloudy	0.147381	0.000000	0.004872	0.012180	0.000000	0.009744	0.179050	0.006090	0.041413	0.003654	0.000000	0.320341	0.182704	0.036541	0.001218	0.002436	0.052375
Clear	0.126413	0.001500	0.006233	0.011036	0.007253	0.011901	0.180225	0.004885	0.044896	0.021230	0.001547	0.288255	0.166774	0.034826	0.011858	0.002259	0.078911
DangerouslyWindy	0.147059	0.000000	0.007353	0.000000	0.000000	0.000000	0.213235	0.000000	0.058824	0.000000	0.000000	0.367647	0.117647	0.000000	0.007353	0.000000	0.080882
Drizzle	0.146100	0.001676	0.004570	0.011883	0.002133	0.003199	0.191347	0.006399	0.037172	0.004723	0.001828	0.306977	0.167885	0.036106	0.003504	0.001371	0.073126
DrizzleandBreezy	0.086420	0.000000	0.004115	0.000000	0.000000	0.012346	0.201646	0.032922	0.028807	0.004115	0.008230	0.341564	0.127572	0.057613	0.000000	0.000000	0.094650
Dry	0.139398	0.000000	0.003668	0.011739	0.005869	0.005136	0.181952	0.008070	0.041820	0.019076	0.002201	0.247982	0.182685	0.049890	0.011739	0.001467	0.087307
DryandMostlyCloudy	0.079545	0.000000	0.000000	0.011364	0.000000	0.000000	0.250000	0.000000	0.045455	0.000000	0.000000	0.386364	0.125000	0.011364	0.000000	0.000000	0.090909

และนำมาสร้างเป็นกราฟได้ดังนี้

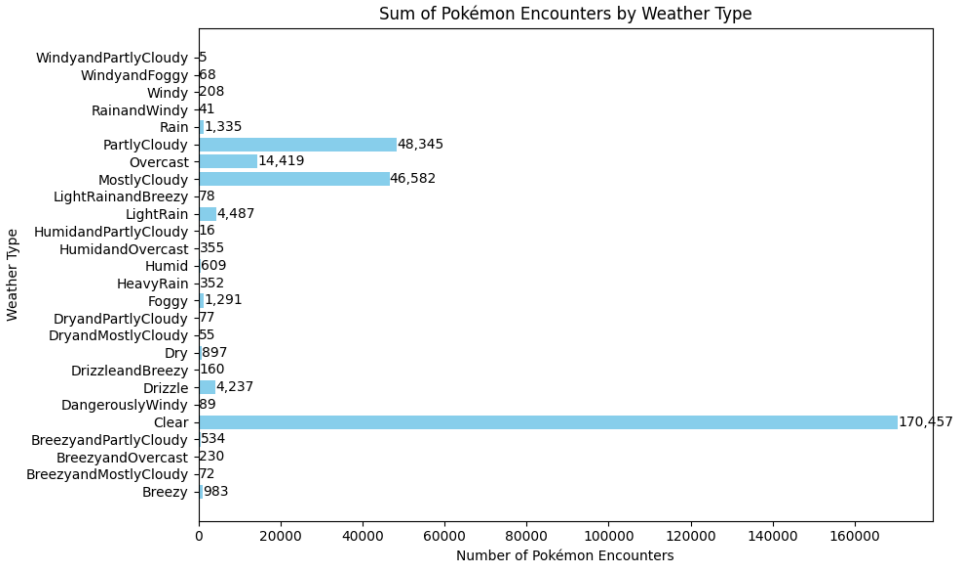
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 8))
sns.heatmap(weather_counts_by_type_normalize, cmap='viridis')
plt.title('Pokemon type encounter by weather')
plt.xlabel('Type')
plt.ylabel('Weather')
plt.show()
```



เมื่อนำมา plot เป็น heatmap พบว่าสภาพอากาศไม่ได้ต่อการเจอโปเกม่อนในแต่ละธาตุที่ปรากฏมากนัก เนื่องด้วยจะเจอ Normal เป็นส่วนใหญ่อยู่แล้วแต่ว่าถ้าหากเทียบกับในธาตุด้วยกันเองจะเห็นว่าในบางสภาพอากาศ บางธาตุจะมีโอกาสเกิดมากกว่าปกติ ในสภาพอากาศ Windy and Partly Cloudy จะมีโอกาสเจอโปเกม่อนนอกจากธาตุ Normal ได้ค่อนข้างสูง เนื่องด้วยเจอธาตุ Water กับ Bug ได้มาก ในข้อมูลจะไม่พบธาตุ Flying เลย ในสภาพอากาศ Humid and Partly Cloudy จะเจอธาตุน้ำได้เยอะเช่นกัน ธาตุ Normal จะน้อยลง แต่ว่าก็จะเจอธาตุ Bug ค่อนข้างน้อย ธาตุสายฟ้าโดยปกติจะไม่ค่อยได้พบเห็น แต่ว่าโอกาสที่จะพบเจอจะเพิ่มขึ้นมานิดหนึ่งในสภาพอากาศที่มีคำว่า "Breezy" อยู่ ใน Heavy Rain ธาตุ Ground และ Rock จะมีโอกาสได้เจอมากที่สุดเมื่อเทียบกับสภาพอากาศอื่น โดยที่ทั้งสองธาตุมีส่วนที่เกี่ยวข้องกับผืนปฐพีเช่นกันหลังจากที่ ดูสถิติสภาพอากาศเทียบกับโปเกมอนที่สามารถจับได้ (สภาพอากาศนี้จับโปเกม่อนได้กี่ตัวใน Data)

สร้างกราฟแสดงจำนวนโปเกมอนที่พบในสภาพอากาศในแต่ล่ะแบบ

```python
import pandas as pd
import matplotlib.pyplot as plt

# Group by 'weather' and count the occurrences
weather_counts = pokemon_df.groupby('weather').size().reset_index(name='encounter_count')

# Create a horizontal bar plot
plt.figure(figsize=(10, 6))
bars = plt.barh(weather_counts['weather'], weather_counts['encounter_count'], color='skyblue')

# Add labels and title
plt.ylabel('Weather Type')
plt.xlabel('Number of Pokémon Encounters')
plt.title('Sum of Pokémon Encounters by Weather Type')

# Add data labels to each bar
for bar in bars:
    plt.text(bar.get_width(), bar.get_y() + bar.get_height()/2, f'{bar.get_width():,}', va='center')

plt.tight_layout()  # Adjust layout for better fit

# Display the plot
plt.show()
```

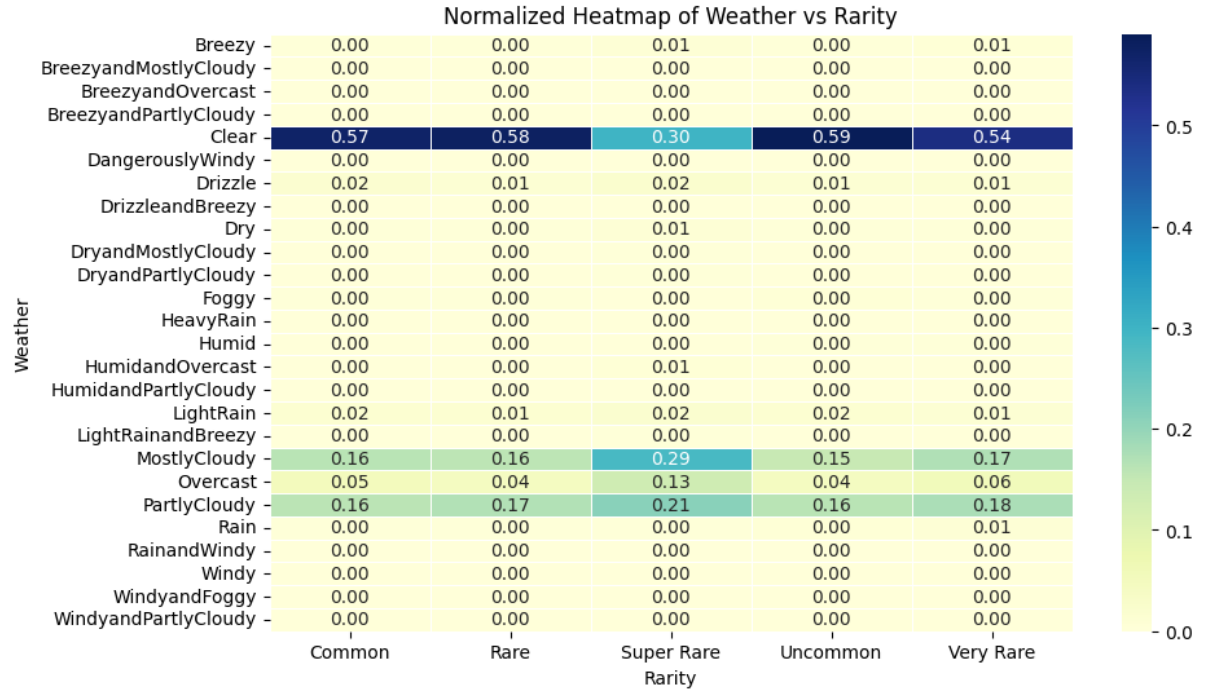


จากกราฟแสดงจำนวนโปเกมอนที่พบในสภาพอากาศในแต่ละแบบพบว่าในสภาพอากาศปลอดโปร่งจะสามารถพบเจอโปเกมอนได้เยอะที่สุดในชุดข้อมูลนี้ ตามด้วยสภาพอากาศมีเมฆเล็กน้อยและเป็นส่วนใหญ่ เมฆครึ้มตามลำดับ และจะมีการพบเจอโปเกมอนน้อยที่สุด ณ สภาพอากาศมีลมและมีเมฆเป็นบางส่วน เนื่องด้วย ผู้เล่นเกม นี้มักเล่นเกมในช่วงสภาพอากาสปลอดโปร่ง เพราะว่าเป็นสภาพอากาศที่เหมาะสำหรับการเดินเท้ามากที่สุดทำให้จำนวนโปเกมอนที่พบได้ในสภาพอากาศปลอดโปร่งของชุดข้อมูลนี้มีมากที่สุดส่วนในสภาพอากาศที่พบได้รองลงมาอย่างเมฆครึ้มหรือเมฆหนาก็เป็นสภาพอากาศที่พบได้ทั่วไปและสามารถพอเดินเท้าออกไปได้บ้าง จึงทำให้มีรายงานการพบโปเกมอนในลำดับที่รองลงมากลับกับในสภาพอากาศอื่น ๆ อาจพบเจอได้แค่บางฤดูกาล

5. Pokemon Rarity by weather

สร้างกราฟแสดงจำนวนโปเกมอนที่หายากในแต่ระดับกับสภาพอากาศในแต่ละแบบ

```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a pivot table to count occurrences of each combination of 'weather' and 'rarity'
pivot_table = pokemon_df.pivot_table(index='weather', columns='rarity', aggfunc='size', fill_value=0)

# Normalize the data by dividing each cell by the sum of its respective row
pivot_table_normalized = pivot_table.div(pivot_table.sum(axis=0), axis=1)

# Plot heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(pivot_table_normalized, cmap="YlGnBu", annot=True, fmt=".2f", linewidths=.5)
plt.title('Normalized Heatmap of Weather vs Rarity')
plt.xlabel('Rarity')
plt.ylabel('Weather')
plt.show()
```



จาก Heat map

แสดงความสัมพันธ์ระหว่างระดับความหายากของโปเกมอนที่เจอกับสภาพอากาศในแต่ล่ะแบบพบว่า

ในการพบโปเกมอนระดับ Common, Uncommon, Rare, Vert rare

จะสามารถพบเจอได้ดีที่สุดในสภาพอากาศปลอดโปร่ง ตามด้วย มีเมฆเล็กน้อยและเป็นส่วนใหญ่

เมฆครึ้มตามลำดับ แต่ในระดับความหายากที่ระดับ Super rare

จะมีจำนวนการพบเจอที่ใกล้เคียงกันในสภาพอากาศปลอดโปร่งและมีเมฆเล็กน้อย ตามด้วยมีเมฆเป็นส่วนใหญ่

และเมฆครึ้มตามลำดับ

6. Pokemon type by time of the day

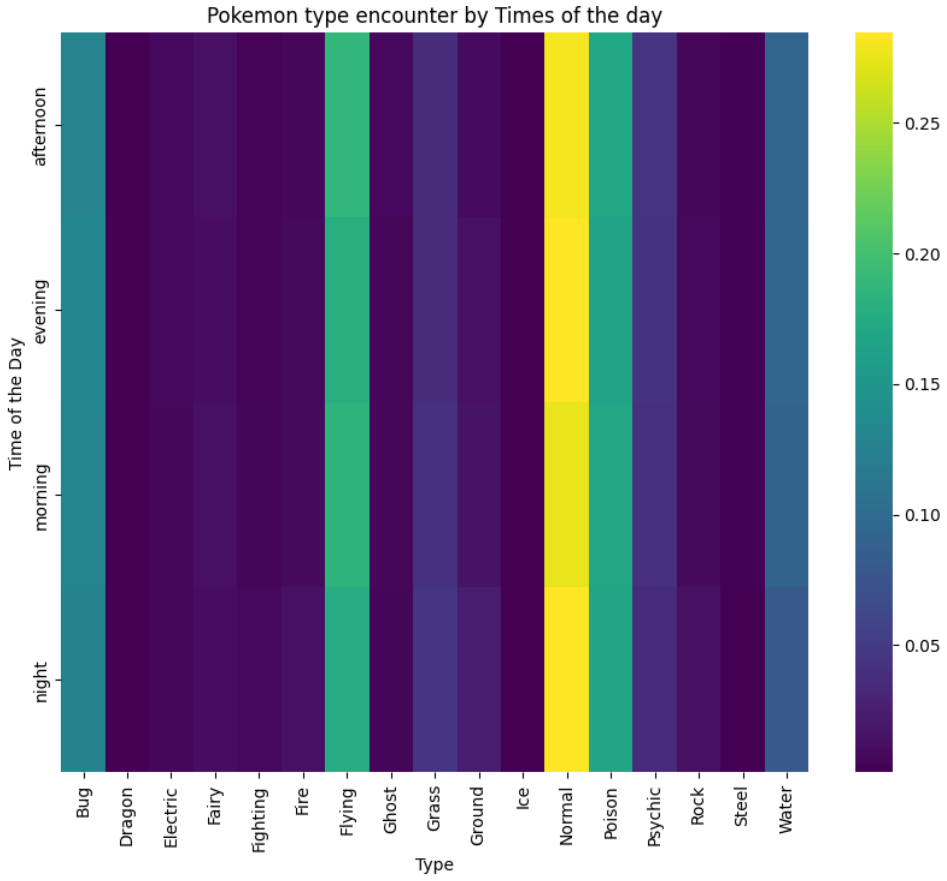
ต้องการทราบความสัมพันธ์ของช่วงเวลาของวันว่ามีผลต่อการปรากฏตัวของโปเกมอนในแต่ละธาตุหรือไม่ เนื่องจากข้อมูลมีระยะห่างมากเกินไปจึงทำการ normalize โดยหารด้วยผลรวมของทั้งแถวเช่นกัน

```
TimeOfDay_counts_by_type_1 = pokemon_df.pivot_table(index='appearedTimeOfDay', columns='Type 1', aggfunc='size', fill_value=0)
TimeOfDay_counts_by_type_2 = pokemon_df.pivot_table(index='appearedTimeOfDay', columns='Type 2', aggfunc='size', fill_value=0)

TimeOfDay_counts_by_type_combine = TimeOfDay_counts_by_type_1.add(TimeOfDay_counts_by_type_2, fill_value=0)

TimeOfDay_counts_by_type_normalize = TimeOfDay_counts_by_type_combine.div(TimeOfDay_counts_by_type_combine.sum(axis=1), axis=0)
TimeOfDay_counts_by_type_combine.head()
```

appearedTimeOfDay	Bug	Dragon	Electric	Fairy	Fighting	Fire	Flying	Ghost	Grass	Ground	Ice	Normal	Poison	Psychic	Rock	Steel	Water
afternoon	9011.0	104.0	541.0	885	272	431.0	13072.0	482.0	2487	644	170	19532.0	11866	3074	422	216.0	6480
evening	8387.0	127.0	470.0	667	276	531.0	11478.0	383.0	2330	836	133	18114.0	10573	2715	499	199.0	5956
morning	11767.0	122.0	538.0	1142	504	810.0	16737.0	568.0	3642	1404	176	24899.0	15445	3653	794	234.0	8180
night	28463.0	311.0	1353.0	2423	1630	2825.0	39795.0	1143.0	10147	5013	353	63278.0	37568	7711	2918	481.0	17786

และนำมาสร้างเป็นกราฟได้ดังนี้

```
plt.figure(figsize=(10, 8))
sns.heatmap(TimeOfDay_counts_by_type_normalize, cmap='viridis')
plt.title('Pokemon type encounter by Times of the day')
plt.xlabel('Type')
plt.ylabel('Time of the Day')
plt.show()
```

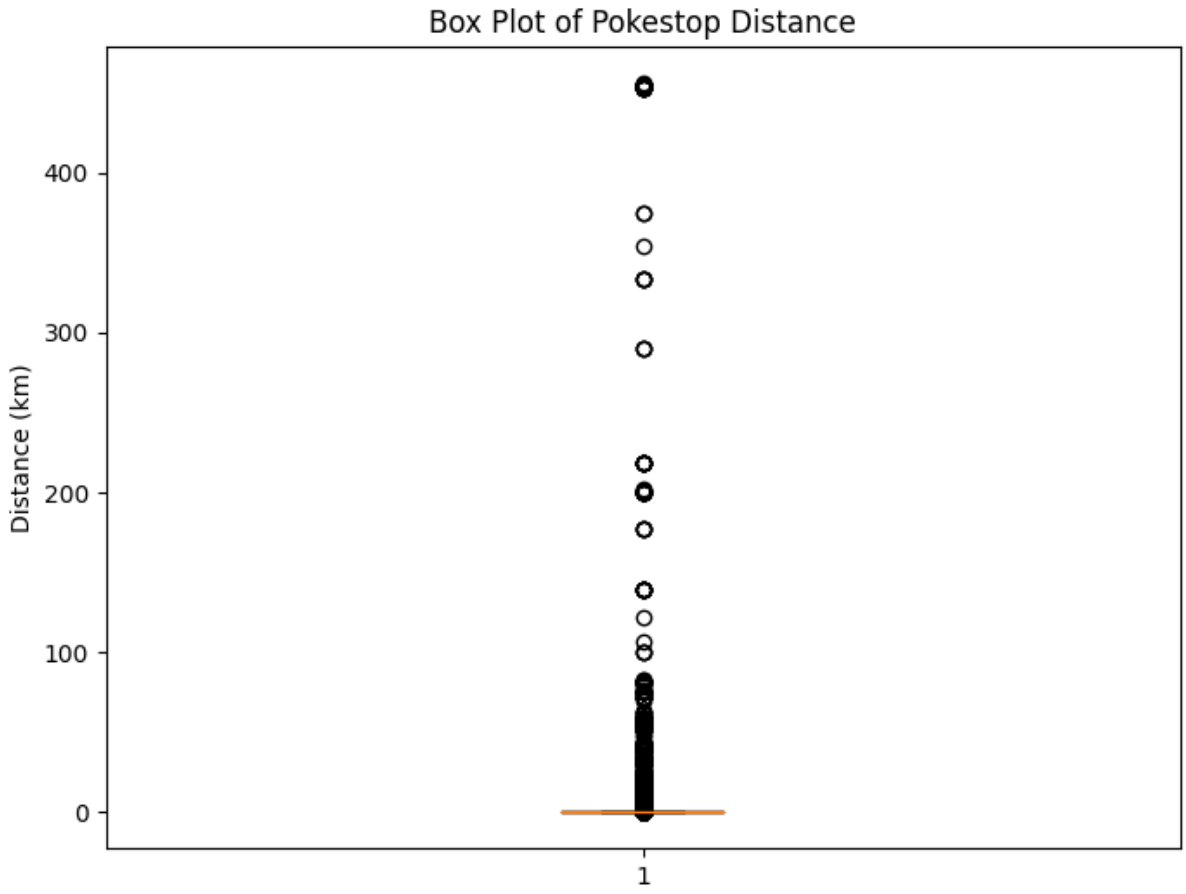


เมื่อนำมา plot เป็น heatmap พบว่าช่วงเวลาของวันไม่ได้มีผลต่อธาตุของโปเกม่อนที่ปรากฏขนาดนั้น การเกิดในแต่ละช่วงยังมีสัดส่วนการเกิดของแต่ละธาตุอยู่คงเดิม

7. Gym and Pokestop distance by pokemon encounter

ต้องการทราบการแจกแจงของระยะห่างระหว่างโปเกมอนที่พบกับ Pokestop และกับ Pokemon gym ว่ามีโปเกมอนมีการแจกแจงในการปรากฏตัวอย่างไร โดยจะเริ่มตรวจสอบโดยใช้ Box plot

```
import matplotlib.pyplot as plt

# Assuming your DataFrame is named 'pokemon_df'

# Create a box plot of the 'pokestopDistanceKm' column
plt.figure(figsize=(8, 6))
plt.boxplot(pokemon_df['pokestopDistanceKm'])
plt.title('Box Plot of Pokestop Distance')
plt.ylabel('Distance (km)')
plt.show()
```



จากการนำข้อมูลระยะห่างระว่างโปเกมอนที่พบและ pokestop ไปแสดงเป็น box plot พบว่ามี outlier อยู่มากจึงทำการเลือกข้อมูลเพียงส่วนที่อยู่ใน Q1-Q3

```python
import pandas as pd

# Assuming your DataFrame is named 'pokemon_df'

# Calculate the first quartile (Q1) and third quartile (Q3)
Q1 = pokemon_df['pokestopDistanceKm'].quantile(0.25)
Q3 = pokemon_df['pokestopDistanceKm'].quantile(0.75)

# Calculate the interquartile range (IQR)
IQR = Q3 - Q1

# Calculate the lower and upper bounds of the box
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter the data to include only values within the box
filtered_data = pokemon_df[(pokemon_df['pokestopDistanceKm'] >= lower_bound) & (pokemon_df['pokestopDistanceKm'] <= upper_bound)]

# Now 'filtered_data' contains only the rows with 'pokestopDistanceKm' values within the box
```

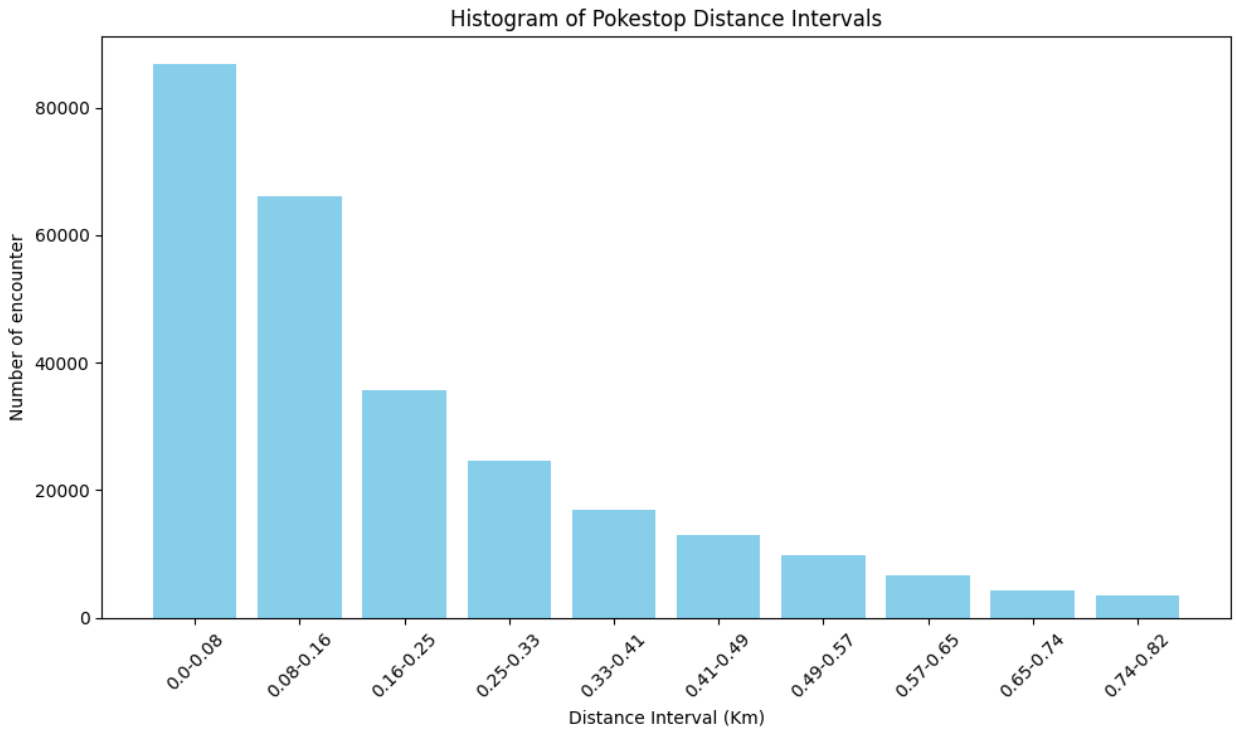
จากนั้น ทำการแบ่งค่าออกเป็นช่วงๆทั้งหมด 10 ช่วง เพื่อให้ง่ายต่อการสร้างกราฟ

```python
import pandas as pd

# Assuming your DataFrame is named 'filtered_data'

# Determine the minimum and maximum values of the 'pokestopDistanceKm' column
min_value = filtered_data['pokestopDistanceKm'].min()
max_value = filtered_data['pokestopDistanceKm'].max()

# Specify the number of intervals (bins) you want
num_intervals = 10  # Adjust this value as needed

# Calculate the interval width
interval_width = (max_value - min_value) / num_intervals

# Create the intervals
bins = [min_value + i * interval_width for i in range(num_intervals)]
bins.append(max_value)  # Append the maximum value to include all data

# Use pd.cut() to bin the values into intervals
interval_labels = [f"{round(bins[i], 2)}-{round(bins[i+1], 2)}" for i in range(len(bins) - 1)]
filtered_data['pokestopDistanceKm_interval'] = pd.cut(filtered_data['pokestopDistanceKm'], bins=bins, labels=interval_labels)

# Count the occurrences of each interval using pd.value_counts()
interval_counts = filtered_data['pokestopDistanceKm_interval'].value_counts().sort_index()

# Create a DataFrame from the interval counts
pokestop_interval_counts_df = pd.DataFrame({'Interval': interval_counts.index, 'Count': interval_counts.values})

# Print the resulting table
print(pokestop_interval_counts_df)
```

```
    Interval  Count
0   0.0-0.08  86756
1  0.08-0.16  66010
2  0.16-0.25  35710
3  0.25-0.33  24627
4  0.33-0.41  16947
5  0.41-0.49  12941
6  0.49-0.57   9875
7  0.57-0.65   6603
8  0.65-0.74   4373
9  0.74-0.82   3468
```

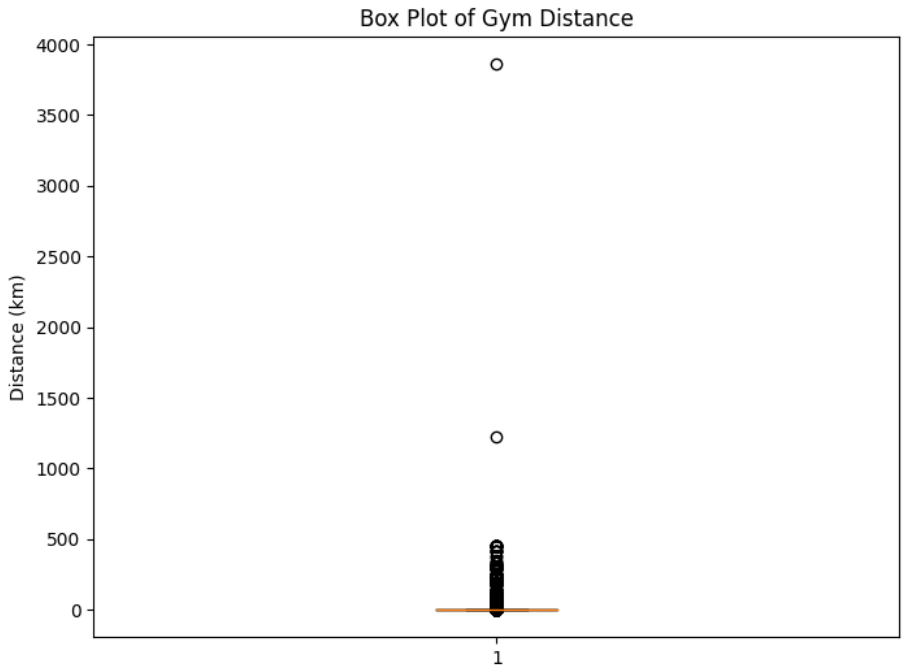
และสามารถสร้างกราฟได้ดังนี้

```
import matplotlib.pyplot as plt

# Plot the histogram
plt.figure(figsize=(10, 6))
plt.bar(pokestop_interval_counts_df['Interval'], pokestop_interval_counts_df['Count'], color='skyblue')
plt.xlabel('Distance Interval (Km)')
plt.ylabel('Number of encounter')
plt.title('Histogram of Pokestop Distance Intervals')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()
```



จากการนำข้อมูลที่ได้มาทำเป็นกราฟพบว่า ระยะห่างระหว่างโปเกมอนที่พบกับ Pokestop ว่ามีผลต่อความถี่การปรากฏตัวของโปเกมอนในชุดข้อมูลนี้ ซึ่งยิ่งความห่างระหว่างตัวโปเกมอนที่พบกับ Pokestop มีค่าน้อยก็จะมีการเจอที่มากขึ้นโดยพบมากสุดที่ช่วงระยะห่าง 0-0.08 กิโลเมตร และจะพบน้อยลงมาเรื่อยๆเมื่อระยะห่างมีค่าเพิ่มขึ้นจนถึงช่วง 0.74-0.82 กิโลเมตร

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
plt.boxplot(pokemon_df['gymDistanceKm'])
plt.title('Box Plot of Gym Distance')
plt.ylabel('Distance (km)')
plt.show()
```



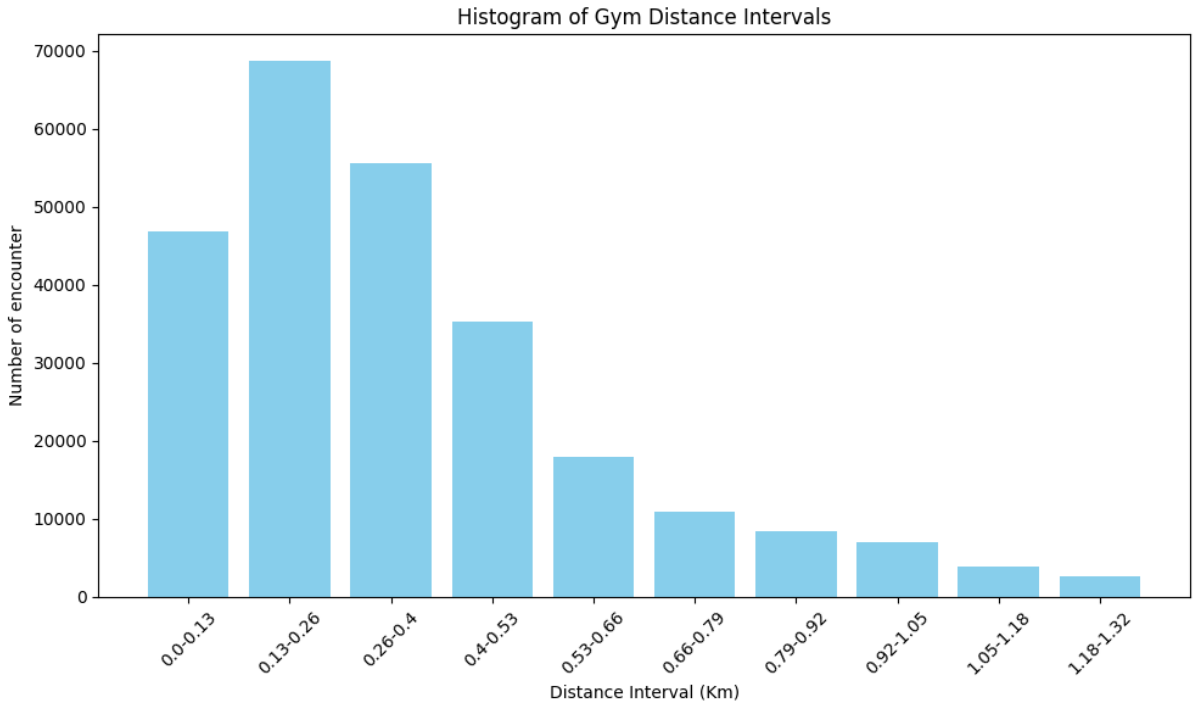
จากการนำข้อมูลระยะห่างระว่างโปเกมอนที่พบและ pokemon gym ไปแสดงเป็น box plot พบว่ามี outlier อยู่มากจึงทำการเลือกข้อมูลเพียงส่วนที่อยู่ใน Q1-Q3 และ ทำการแบ่งค่าออกเป็นช่วงๆทั้งหมด 10 ช่วง เช่นกัน

```python
Q1 = pokemon_df['gymDistanceKm'].quantile(0.25)
Q3 = pokemon_df['gymDistanceKm'].quantile(0.75)

IQR = Q3 - Q1

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

filtered_data = pokemon_df[(pokemon_df['gymDistanceKm'] >= lower_bound) & (pokemon_df['gymDistanceKm'] <= upper_bound)]


min_value = filtered_data['gymDistanceKm'].min()
max_value = filtered_data['gymDistanceKm'].max()

num_intervals = 10

interval_width = (max_value - min_value) / num_intervals

bins = [min_value + i * interval_width for i in range(num_intervals)]
bins.append(max_value)

interval_labels = [f"{round(bins[i], 2)}-{round(bins[i+1], 2)}" for i in range(len(bins) - 1)]
filtered_data['gymDistanceKm_interval'] = pd.cut(filtered_data['gymDistanceKm'], bins=bins, labels=interval_labels)

interval_counts = filtered_data['gymDistanceKm_interval'].value_counts().sort_index()

gym_interval_counts_df = pd.DataFrame({'Interval': interval_counts.index, 'Count': interval_counts.values})

print(gym_interval_counts_df)
```

```
    Interval  Count
0   0.0-0.13  46850
1  0.13-0.26  68638
2   0.26-0.4  55537
3    0.4-0.53  35218
4  0.53-0.66  17986
5  0.66-0.79  10938
6  0.79-0.92   8458
7  0.92-1.05   6928
8  1.05-1.18   3818
9  1.18-1.32   2660
```

และสามารถทำเป็นกราฟได้ดังนี้

```
import matplotlib.pyplot as plt


plt.figure(figsize=(10, 6))
plt.bar(gym_interval_counts_df['Interval'], gym_interval_counts_df['Count'], color='skyblue')
plt.xlabel('Distance Interval (Km)')
plt.ylabel('Number of encounter')
plt.title('Histogram of Gym Distance Intervals')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

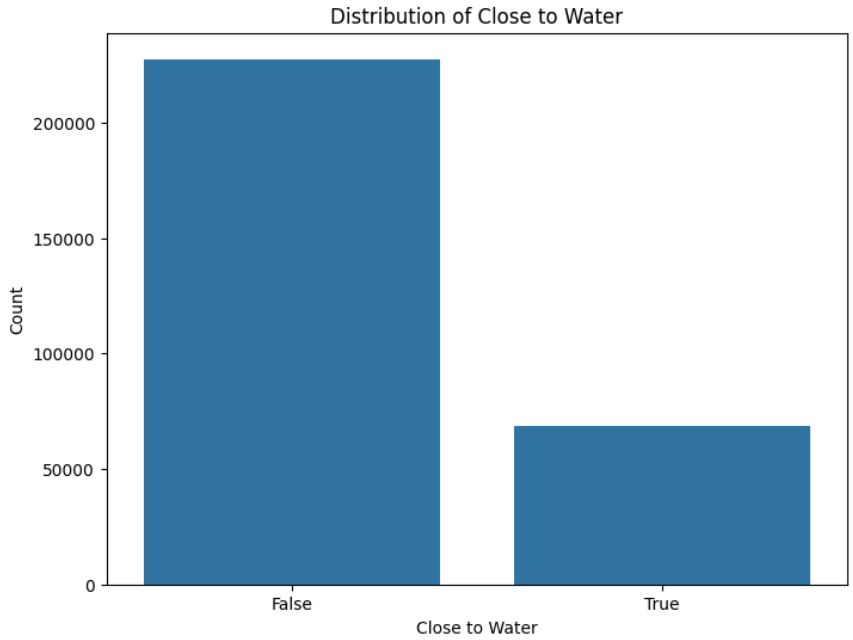


จากการนำข้อมูลที่ได้มาทำเป็นกราฟพบว่า ระยะห่างระหว่างโปเกมอนที่พบกับ Pokemon Gym ว่ามีผลต่อความถี่การปรากฏตัวของโปเกมอนในชุดข้อมูลนี้ ซึ่งยิ่งความห่างระหว่างตัวโปเกมอนที่พบกับ Pokemon Gym มีค่าน้อยก็จะมีการเจอที่มากขึ้นโดยเริ่มตั้งแต่ช่วงที่มีระยะห่าง 0.13 - 0.26 กิโลเมตร ที่มีการพบเจอโปกมอนเยอะมากที่สุดและจะพบน้อยลงมาจนไปถึงช่วงที่มีระยะห่าง 1.18-1.32 กิโลเมตร เราสามารถคาดการณ์อีกได้ว่าผู้เล่นมักไม่ได้จับโปเกมอนโดยอยู่ใกล้ชิดกับ Pokemon Gym โดยตรง เนื่องจาก Gym มีกระจายอยู่ไม่ทั่วถึงเท่า Pokestop แต่ในระยะห่างที่ห่างจาก Pokemon Gym เล็กน้อยก็ทำให้สามารถพบโปเกมอนอยู่ได้มากเช่นกัน

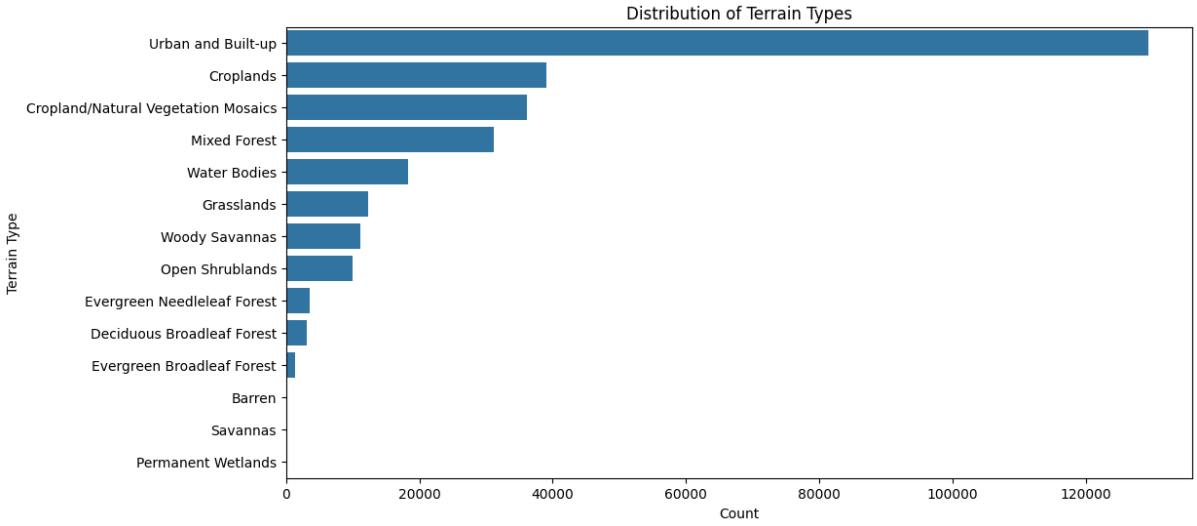
8. Terrain type and closeToWater by pokemon encounter

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Plot distribution of terrainType with horizontal bars
plt.figure(figsize=(12, 6))
sns.countplot(data=pokemon_df, y='terrainType', order=pokemon_df['terrainType'].value_counts().index)
plt.title('Distribution of Terrain Types')
plt.xlabel('Count')
plt.ylabel('Terrain Type')
plt.show()

# Plot distribution of closeToWater
plt.figure(figsize=(8, 6))
sns.countplot(data=pokemon_df, x='closeToWater')
plt.title('Distribution of Close to Water')
plt.xlabel('Close to Water')
plt.ylabel('Count')
plt.xticks([0, 1], ['False', 'True'])
plt.show()

# Plot relationship between terrainType and closeToWater with horizontal bars
plt.figure(figsize=(12, 6))
sns.countplot(data=pokemon_df, y='terrainType', hue='closeToWater', order=pokemon_df['terrainType'].value_counts().index)
plt.title('Distribution of Terrain Types by Close to Water')
plt.xlabel('Count')
plt.ylabel('Terrain Type')
plt.legend(title='Close to Water', loc='upper right')
plt.show()
```

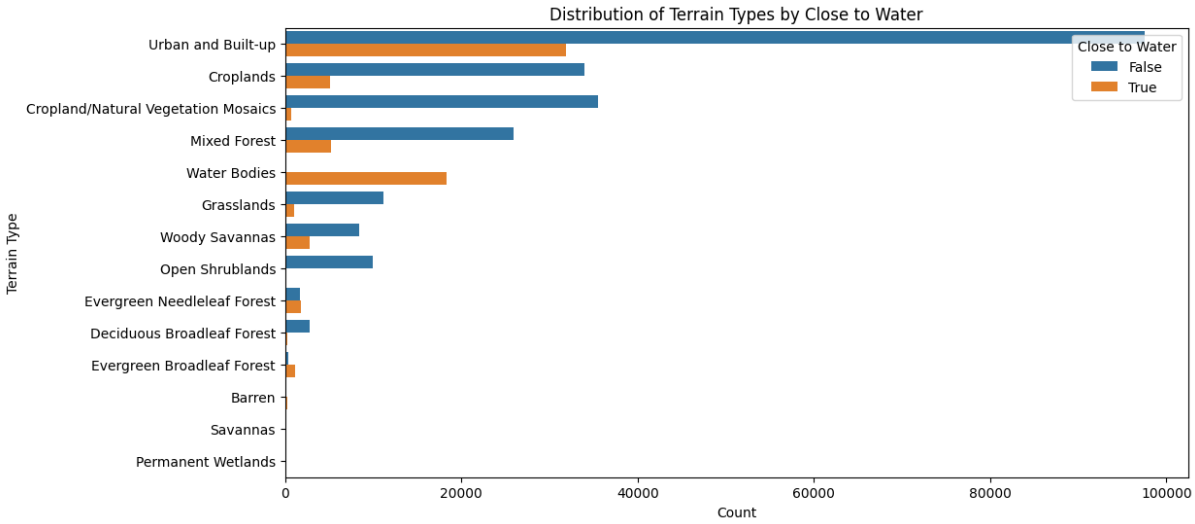
กราฟแสดงจำนวนโปเกมอนที่พบและพบอยู่ใกล้แหล่งน้ำหรือไม่



กราฟแสดงจำนวนโปเกมอนที่พบและรูปแบบของภูมิประเทศ



กราฟแสดงจำนวนโปเกมอนที่พบ รูปแบบของภูมิประเทศ และอยู่ใกล้แหล่งน้ำหรือไม่

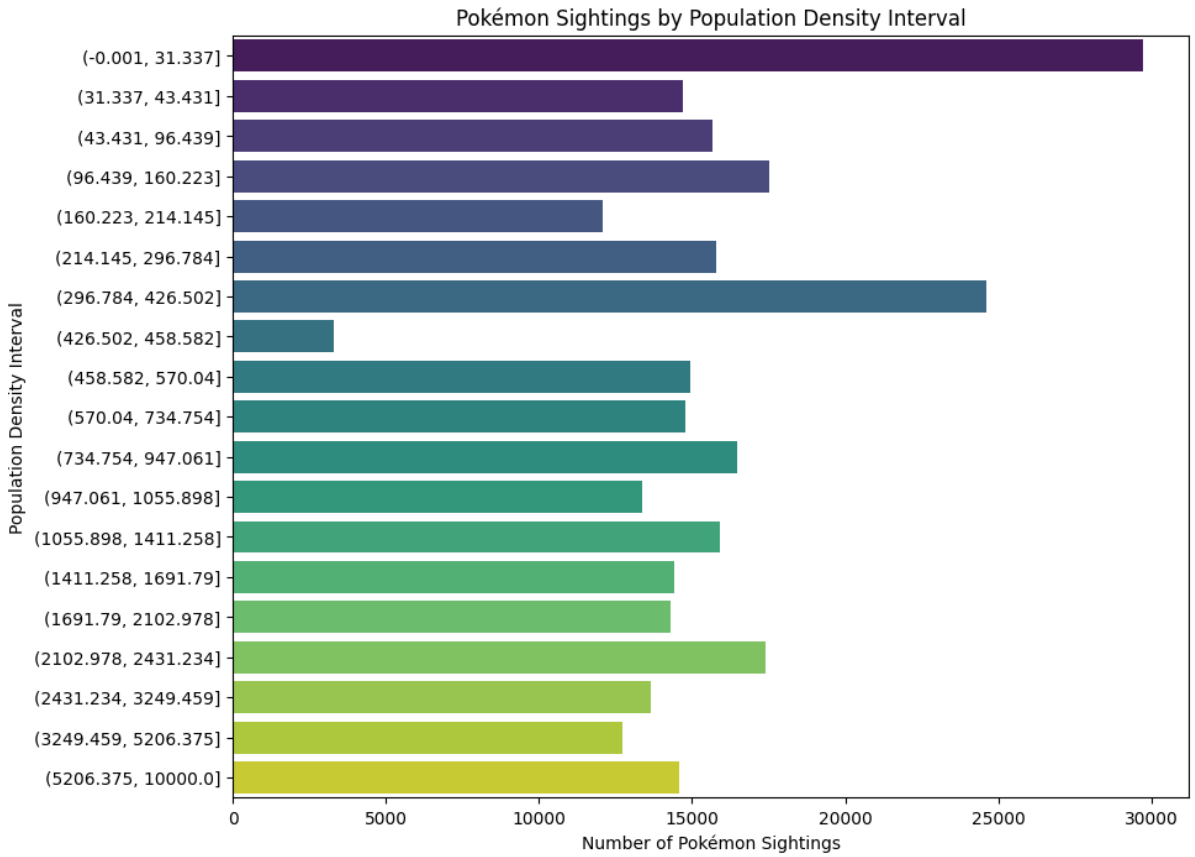


จากกราฟแสดงจำนวนโปเกมอนที่พบ รูปแบบของภูมิประเทศ และอยู่ใแหล่งน้ำหรือไม่พบว่า รูปแบบภูมิประเทศที่มีการพบเจอโปเกมอนเยอะมากที่สุดคือ Urban and Built-up ซึ่งก็คือพื้นที่ปกคลุมด้วยสิ่งก็สร้างหรือตัวเมือง รองมาจะเป็น Croplands ซึ่งคือพื้นที่ใกล้ที่เพาะปลูก และรองมาเป็นพื้นที่เพาะปลูกที่ขึ้นตามธรรมชาติ หรือทุ่งหญ้าเขียวชะอุ่มในภูมิประเทศที่ไม่ใกล้แหล่งน้ำมักจะพบโปเกมอนในพื้นที่เพาะปลูกที่ขึ้นมากกว่าพื้นที่เพาะปลูกโดยมนุษย์เล็กน้อยแต่ก็จะพบโปเกมอนที่อยู่ในป่าผสมเช่นกันเป็นลำดับถัดมา ในภูมิประเทศส่วนใหญ่พื้นที่ที่ไม่ใกล้แหล่งน้ำจะมีการพบเจอโปเกมอนมากกว่าพื้นที่ที่ใกล้แหล่งน้ำ ยกเว้นในตัวแหล่งน้ำเองกับป่าที่มีต้นไม้ไม่ผลัดใบและใบกว้าง ส่วนป่าที่มีต้นไม้ไม่ผลัดใบและใบแหลมจะพบโปเกมอนใกล้น้ำและไม่ใกล้น้ำในจำนวนที่ใกล้เคียงกัน

9. Population_density by pokemon encounter

กราฟแสดงจำนวนโปเกมอนที่พบ กับความหนาแน่นของประชากร

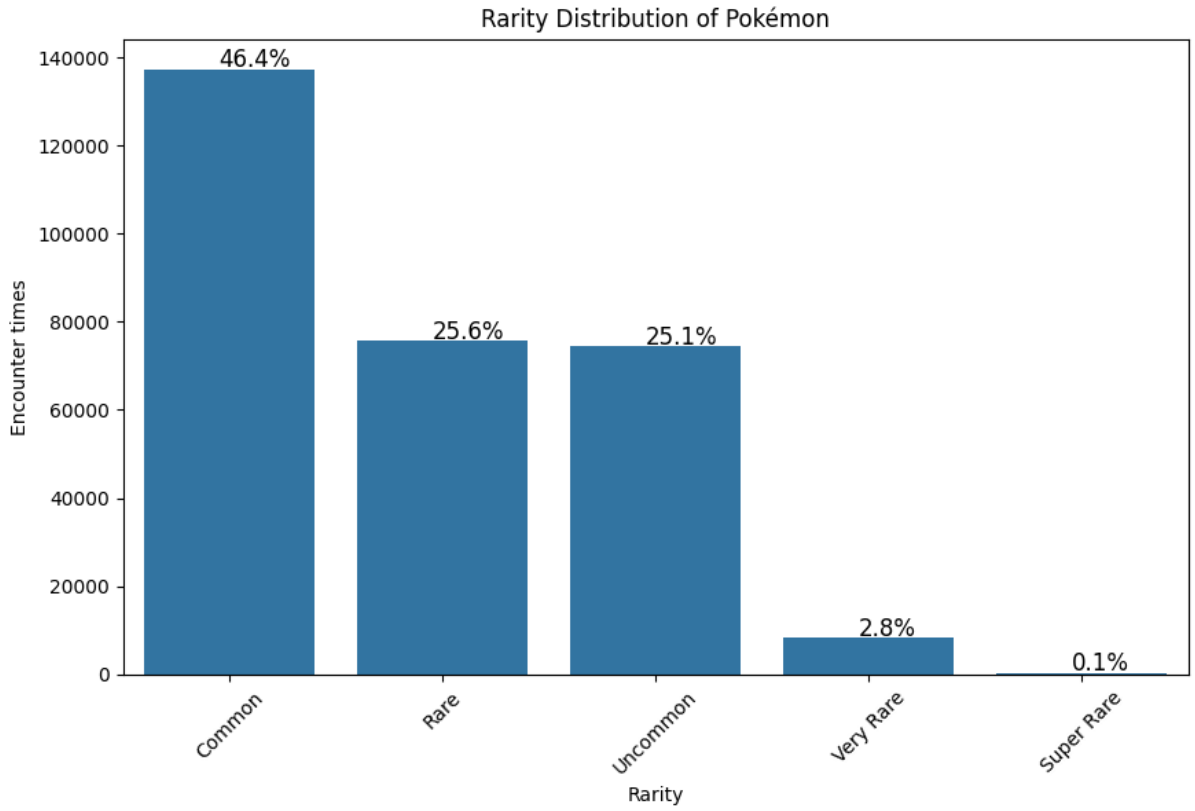
```
import matplotlib.pyplot as plt
import seaborn as sns

num_bins = 20  # Specify the number of bins
intervals = pd.qcut(pokemon_df['population_density'], q=num_bins, duplicates='drop')

# Step 4: Create bins
pokemon_df['population_density_interval'] = intervals

# Step 5: Analyze and Visualize
plt.figure(figsize=(10, 8))  # Adjust figsize if needed
sns.countplot(y='population_density_interval', data=pokemon_df, palette='viridis')
plt.title('Pokémon Sightings by Population Density Interval')
plt.xlabel('Number of Pokémon Sightings')
plt.ylabel('Population Density Interval')
plt.show()
```



จากกราฟแสดงจำนวนโปเกมอนที่พบ กับความหนาแน่นของประชากรจะพบว่าจะพบโปเกม่อนเยอะสุดที่ช่วงความหนาแน่นประชากร -0.001 - 31.337 และน้อยสุดที่ช่วง 426.502 - 458.582 จากการแจกแจง จะพบได้ว่านอกจากช่วงที่มากที่สุดและช่วงน้อยที่สุดแล้วการพบโปเกมอนก็ไม่ได้มีความสัมพันธ์มากนักกับความหนาแน่นของประชาการ โอกาสพบจะอยู่ที่ไล่เลี่ยกัน

10. Rarity by pokemon encounter

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Plot the rarity distribution
plt.figure(figsize=(10, 6))
ax = sns.countplot(data=pokemon_df, x='rarity', order=pokemon_df['rarity'].value_counts().index)
plt.title('Rarity Distribution of Pokémon')
plt.xlabel('Rarity')
plt.ylabel('Encounter times')
plt.xticks(rotation=45)

# Add labels with percentage
total = len(pokemon_df['rarity'])
for p in ax.patches:
    percentage = '{:.1f}%'.format(100 * p.get_height() / total)
    x = p.get_x() + p.get_width() / 2 - 0.05
    y = p.get_height() + 500
    ax.annotate(percentage, (x, y), fontsize=12)

plt.show()
```



จากกราฟแสดงระดับความหายากของโปเกมอนกับจำนวนโปเกมอนที่พบเจอจะพบว่าระดับ Common หาง่ายสุดที่ 46.4% และยากสุดที่ระดับ Super Rare 0.1% โดยที่โปเกมอนระดับความหายาก Rare และ Uncommon มีความหายากที่ใกล้เคียงกันที่ค่าประมาณ 25%

Data Modeling Method

เนื่องด้วยการหาเงื่อนไขในการจับโปเกมอนหายากนั้นมีหลากหลายวิธีเนื่องด้วยหลายปัจจัย
ทางผู้จัดทำได้ตัดสินใจใช้วิธีการสร้างแบบจำลอง 2 แบบ
เพื่อเป็นการตอบปัญหาในสองด้านที่เกี่ยวกับการหาเงื่อนไขในการจับโปเกมอนหายาก

1. Classification แบบจำลองในการจำแนก โดยจะใช้ในการแก้ปัญหาว่า
   “ถ้าจับโปเกมอนในเงื่อนไขสภาพอากาศ ภูมิประเทศ แรงลม ฯลฯ จะได้โปเกมอนที่หายากหรือไม่”
   ซึ่งจะจำแนกได้ True (หายาก) กับ False (หาไม่ยาก)
   โดยจะได้คำตอบที่ออกมาจากแบบจำลองอย่างชัดเจนว่าถ้าจับโปเกมอนที่เงื่อนไขที่กำหนดไว้จะมีโอกาสเจอ Rare Pokemon หรือไม่
2. Clustoring แบบจำลองในการจัดกลุ่ม โดยจะใช้ในการแก้ปัญหาว่า “โปเกมอนที่หายากจะมีเงื่อนไขสภาพอากาศ ภูมิประเทศ แรงลม ฯลฯ ในแต่ละกลุ่มเป็นแต่ละเงื่อนไข”
   โดยจะได้คำตอบเป็นกลุ่มโปเกมอนหายาก พร้อมเงื่อนไขการจับที่คล้ายกันของโปเกมอนในกลุ่มนั้น ๆ

- **Sampling แบ่งประเภทความความหายากจาก Rarity**

ทำการเปลี่ยนระดับความหายากให้เหลือเพียง Common และ Rare เพื่อใช้ในการจัดทำแบบจำลองทั้งคู่
และสร้าง column boolean ที่ระบุว่าโปเกมอนที่พบนั้นมีความหายากระดับ Rare หรือไม่

```
[ ] pokemon_df['rarity'] = pokemon_df['rarity'].replace({'Uncommon': 'Common',
                                                         'Very Rare': 'Rare',
                                                         'Super Rare': 'Rare'})

[ ] rare_counts = pokemon_df['rarity'].value_counts()
    print(rare_counts)

    rarity
    Common    211618
    Rare       84364
    Name: count, dtype: int64
```

```
pokemon_df['is_rare'] = pokemon_df['rarity'].apply(lambda x: x == 'Rare')
```

เลือก Column ที่เหมาะสมในการ Classification

```
[ ] selected_columns = [ 'appearedTimeOfDay', 'terrainType', 'closeToWater', 'temperature', 'windSpeed', 'windBearing', 'pressure', 'population_density', 'gymDistanceKm', 'pokestopDistanceKm','is_rare']

    pokemon_df_classification = pokemon_df.loc[:, selected_columns]
    pokemon_df_classification.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 295982 entries, 0 to 296020
Data columns (total 11 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   appearedTimeOfDay   295982 non-null  object
 1   terrainType         295982 non-null  int64
 2   closeToWater        295982 non-null  bool
 3   temperature         295982 non-null  float64
 4   windSpeed           295982 non-null  float64
 5   windBearing         295982 non-null  int64
 6   pressure            295982 non-null  float64
 7   population_density  295982 non-null  float64
 8   gymDistanceKm       295982 non-null  float64
 9   pokestopDistanceKm  295982 non-null  float64
 10  is_rare             295982 non-null  bool
dtypes: bool(2), float64(6), int64(2), object(1)
memory usage: 23.1+ MB
```

Sampling For Classification : Undersampling

เนื่องด้วยข้อมูลของ Rare Pokemon มีน้อยกว่า Not Rare ทำให้เมื่อนำข้อมูลไป Train แล้วจะเกิดค่า Recall และ Accuracy ในส่วน Rare อยู่ที่น้อยมาก เราจึงต้องทำการ Undersampling ตัวที่ Not Rare ให้มีขนาดน้อยกว่าตัว Rare อยู่เล็กน้อยโดยการตัดข้อมูลของ Dataset

โดยนำข้อมูลโปเกมอนที่ไม่ได้อยู่ในระดับ Rare ออกไป 70%

```
# Filter the rows where is_rare is False
false_rare_rows = pokemon_df_classification[pokemon_df_classification['is_rare'] == False]

# Randomly select 50% of the rows where is_rare is False
rows_to_drop = false_rare_rows.sample(frac=0.70, random_state=1)

# Drop the selected rows from the original DataFrame
pokemon_df_classification = pokemon_df_classification.drop(rows_to_drop.index)

# Reset the index of the resulting DataFrame
pokemon_df_classification = pokemon_df_classification.reset_index(drop=True)

rare_counts = pokemon_df_classification['is_rare'].value_counts()
print(rare_counts)
```

```
is_rare
True     84364
False    63485
Name: count, dtype: int64
```

ทำการเปลี่ยนข้อมูลประเภท object ให้เป็น boolean

```
[ ]  pokemon_df_1 = pokemon_df_classification.select_dtypes('object')
     pokemon_df_cat = pd.get_dummies(pokemon_df_1, drop_first = True)
     pokemon_df_num = pokemon_df_classification.select_dtypes('number')
     pokemon_df_prep = pd.concat([pokemon_df_num,pokemon_df_cat,pokemon_df_classification['is_rare']], axis = 1)
     pokemon_df_prep.head()
```

ทำการแบ่งข้อมูลเป็นส่วนที่นำไปใช้ทำ classification และ ใช้ ทดสอบที่ 30% จะได้โอกาส Train กับ Test อยู่ที่ค่าใกล้เคียงกัน

```
from sklearn.model_selection import train_test_split
pokemon_df_train, pokemon_df_test = train_test_split(pokemon_df_prep, test_size = 0.3)
```

```
[ ]  pd.crosstab(pokemon_df_train['is_rare'],columns = 'p',normalize = True)
```

is_rare	col_0 p
False	0.429571
True	0.570429

```
[ ]  pd.crosstab(pokemon_df_test['is_rare'],columns = 'p',normalize = True)
```

is_rare	col_0 p
False	0.428971
True	0.571029

Sampling for Clustering

เนื่องด้วยเราต้องการที่จะรู้เงื่อนไขของการจับโปเกมอนแบบ Rare เท่านั้น เราจึงเลือก Data ที่เป็นการพบโปเกมอนที่มีระดับความหายาก Rare และเลือก column ที่เหมาะสมเพื่อใช้ในการ Clustering

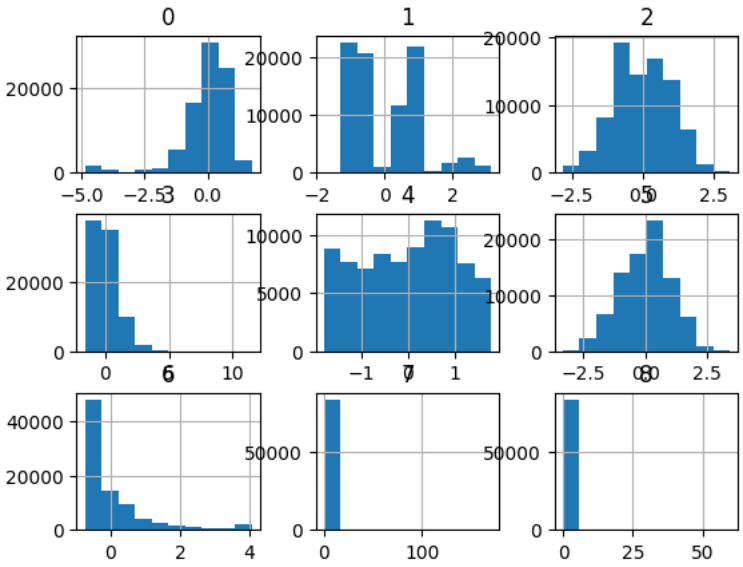
```
rare_pokemon_df = pokemon_df[pokemon_df['is_rare'] == True]
rare_counts = rare_pokemon_df['is_rare'].value_counts()
print(rare_counts)
```

```
is_rare
True    84364
Name: count, dtype: int64
```

```
selected_columns = ['latitude','longitude', 'temperature', 'windSpeed', 'windBearing', 'pressure', 'population_density', 'gymDistanceKm', 'pokestopDistanceKm']

pokemon_df_selected= rare_pokemon_df.loc[:, selected_columns]
pokemon_df_selected.info()
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df_prep = pd.DataFrame(scaler.fit_transform(pokemon_df_selected),index=pokemon_df_selected.index)
```

```
df_prep.hist()
```



จาก histogram จะได้เห็นการกระจายตัวของข้อมูลตามแต่ละคอลัมน์ดังนี้

- 0 : latitude คาดว่าค่ากลางส่วนใหญ่จะเป็นเส้นที่คาดไปในช่วงยุโรป-อเมริกา และช่วงล่าง ๆ จะเป็นเส้นคาดไปช่วงออสเตเรีย-นิวซีแลนด์-อเมริกาใต้
- 1 : longtitude จะพบในช่วงเลยค่า 0 กับต่ำกว่าค่า 0 คาดว่าค่า 0 คือตรงช่วงใกล้กับเส้นคาดมกาสมุทรแอตแลนติกจึงทำให้ค่าที่เจอพบได้น้อย แต่ช่วงมากกว่า 0 จะเป็นยุโรป และน้อยกว่า 0 จะเป็นสหรัฐอเมริกา
- 2 : temperature ค่าอุณหภูมิ มีการแจกแจงที่ค่อนข้างโค้งปกติ
- 3 : windspeed ความเร็วลมส่วนใหญ่จะอยู่ในช่วง 0 หรือก็คือลมนิ่งปกติ ทำให้กราฟเบ้ขวา

- 4 : windbearing ทิศทางลมจะคละกันไป คาดว่าค่ากลางคือ 180 (ใต้)
- 5 : pressure ความกดอากาศส่วนใหญ่แล้วจะพบโปเกมอนได้ในช่วงปกติ ทำให้มีกราฟเป็นโค้งปกติ
- 6 : population density โปเกมอนส่วนใหญ่พบในที่ที่มีความหนาแน่นคนไม่ได้มากนัก แต่ว่าก็มีบางส่วนที่พบในความหนาแน่นมากเช่นกัน
- 7 : gym distance โปเกมอนพบในจุดที่ไม่ได้หาจากยิมเกือบจะทั้งหมด
- 8 : pokestop distance โปเกมอนพบในจุดที่ไม่ได้ห่างจาก pokestop เกือบจะทั้งหมด

- **Classification**

สร้าง decision tree ที่มี min_samples_leaf = 10 และ max_depth=8

```
tree = DecisionTreeClassifier(min_samples_leaf=10, max_depth=8)
tree.fit(pokemon_df_train.drop(columns='is_rare'),
         pokemon_df_train['is_rare'])
```

โดยเมื่อนำไปสร้าง Decision tree ได้ดังนี้

```
from six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(tree, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=False,
                rotate=True,
                feature_names=pokemon_df_train.columns[:-1])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('tree.png')
Image(graph.create_png())
```

- **Clustering**

กำหนดให้ใช้จำนวน cluster ของกลุ่มโปเกมอน Rare ตามเงื่อนไขมีค่าเท่ากับ 9

โดยค่านี้เป็นค่าที่จะได้กลุ่มก้อนที่แตกต่างกันมากที่สุดโดยที่ไม่มีกลุ่มที่คล้ายคลึงกันจนซ้ำกัน

```
from sklearn.cluster import KMeans

cls = KMeans(n_clusters=9)
cls.fit(df_prep)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
  warnings.warn(
```

```
KMeans(n_clusters=9)
```

```
cls.labels_
```

```
array([2, 1, 7, ..., 6, 4, 5], dtype=int32)
```

```
cls.cluster_centers_
```

```
array([[-4.01568804e-01,  1.10395032e+00,  5.98518160e-01,
         1.22087893e-01,  1.31835691e-01, -3.97287934e-01,
         2.94284926e+00, -8.00830114e-02, -7.04456969e-02],
       [ 1.38774666e-01, -1.02768524e+00, -9.89363463e-01,
        -5.67187088e-01,  4.83145702e-01, -3.48929454e-01,
        -1.13857429e-01, -5.94215976e-02, -4.05205025e-02],
       [-2.72439182e-01, -8.66135530e-01,  1.01208014e+00,
         4.50515666e-01,  5.78869003e-01, -8.01681626e-01,
        -1.67659756e-01, -2.34758859e-02,  5.23567941e-02],
       [-1.64699687e+00,  1.53516253e+00,  7.23462893e-01,
         1.53969845e-01,  2.98807986e-01,  4.31511785e-02,
        -6.92386399e-01,  2.00095460e+01,  5.96900351e+01],
       [ 7.69581331e-01,  6.97926178e-01, -5.34596214e-01,
         8.73431153e-01,  7.26305321e-01, -3.02028953e-01,
        -2.36967993e-01, -5.14342097e-02, -2.28713547e-02],
       [ 5.90549473e-01,  7.04691086e-01, -3.82920703e-01,
        -5.88359252e-01, -3.18887983e-01,  5.46959918e-01,
        -3.07106500e-01, -5.93688207e-02, -3.31033663e-02],
       [-1.71876381e-01, -5.67751304e-01,  4.85363595e-01,
        -1.03766129e-01, -9.38902638e-01,  5.37339770e-01,
        -1.77540860e-01, -3.21841319e-02, -4.64688687e-03],
       [-3.47494882e+00,  1.75033054e+00, -3.33092867e-01,
         1.62848752e-01,  1.32777011e-01,  1.81623885e-01,
         6.21703409e-02, -3.77747965e-02, -3.58389544e-02],
       [-1.86638495e+00,  1.88163394e+00,  1.00651245e+00,
        -2.82524178e-01,  3.79196837e-01, -6.58508277e-01,
        -7.02311036e-01,  1.36974398e+01,  6.82836110e-01]])
```

Modeling Result and Discussion

- Classification

ได้ค่า Variable importance ดังนี้

```
pd.DataFrame(dict(Feature=pokemon_df_train.columns[:-1],
                  Value=tree.feature_importances_))\
  .sort_values(by='Value', ascending=False)
```

	Feature	Value
4	population_density	0.470823
6	pokestopDistanceKm	0.105784
17	terrainType_Open Shrublands	0.085333
0	temperature	0.067272
3	pressure	0.051767
5	gymDistanceKm	0.044297
21	terrainType_Water Bodies	0.044114
1	windSpeed	0.035371
2	windBearing	0.029319
15	terrainType_Grasslands	0.020329
20	terrainType_Urban and Built-up	0.015923
16	terrainType_Mixed Forest	0.013275
11	terrainType_Croplands	0.010645
9	appearedTimeOfDay_night	0.003185
22	terrainType_Woody Savannas	0.001384
8	appearedTimeOfDay_morning	0.001180

และผลการทดสอบดังนี้

```
[ ]  res = tree.predict(pokemon_df_test.drop(columns='is_rare'))
     print(classification_report(y_true=pokemon_df_test['is_rare'].values, y_pred=res))
```

```
              precision    recall  f1-score   support

       False       0.56      0.40      0.47     19147
        True       0.63      0.77      0.69     25208

    accuracy                           0.61     44355
   macro avg       0.59      0.58      0.58     44355
weighted avg       0.60      0.61      0.59     44355
```

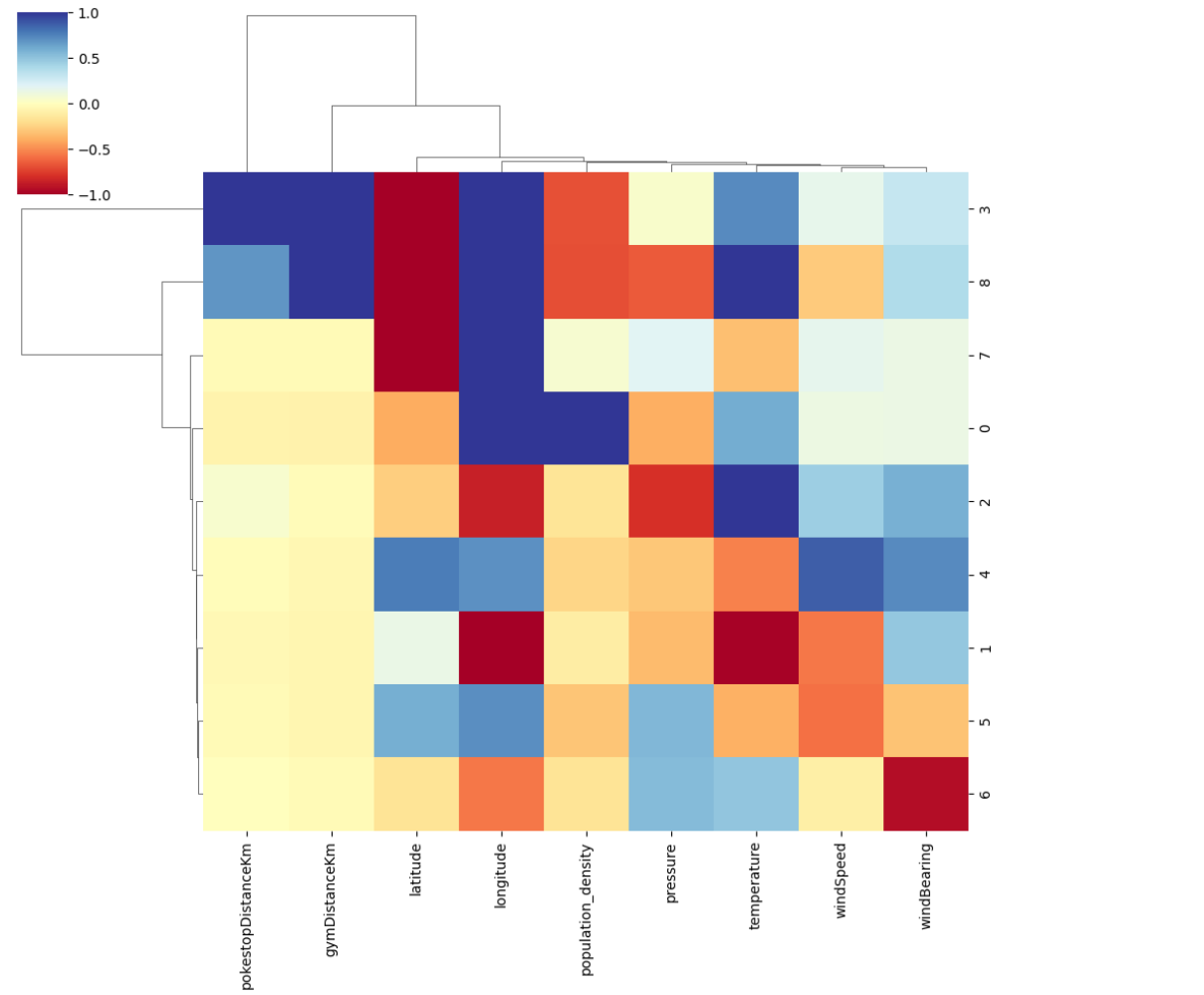
จากผลลัพธ์จะพบว่าแบบจำลองแบบจำแนกอาจจะไม่ได้ทำนายได้ Accurate ที่สุดในการทำนายโดยรวม เนื่องด้วยปัจจัยการเกิดของโปเกมอนหายากอาจแตกต่างกันไปตามแต่ละกลุ่มจึงทำให้แบบจำลองมีความแม่นยำอยู่ที่ประมาณ 0.61 แต่ว่าค่า Recall กับ Precision จะมีค่าสูงขึ้นมาเล็กน้อยเป็น 0.63 กับ 0.77 เพราะการเลือกใช้ Undersampling จึงทำให้สามารถทำนายโปเกมอนประเภท Rare ได้แม่นยำยิ่งขึ้น ซึ่งสอดคล้องกับจุดประสงค์ของการสร้างแบบจำลองนี้ซึ่งคือ

“ต้องการหาเงื่อนไขที่จะทำให้ได้เจอโปเกมอนแบบ Rare”

- **Clustering**

นำ cluster ที่สร้างได้ทำออกมาเป็น Heatmap ได้ดังนี้

```
import seaborn as sns

sns.clustermap(pd.DataFrame(cls.cluster_centers_,columns=['latitude','longitude', 'temperature', 'windSpeed', 'windBearing', 'pressure', 'population_density', 'gymDistanceKm', 'pokestopDistanceKm']), cmap='RdYlBu',
               vmin=-1, vmax=1)
```



จะสามารถตีความในแต่ละกลุ่มได้ว่า

- กลุ่มที่ 0 จะจับได้ในระยะที่ใกล้ Pokestop และ Gym ในช่วง latitude ใกล้ยุโรปและ longtitude ห่างจากมหาสมุทรแอตแลนติกไปทางขวา (คาดว่าเป็นโซนยุโรปที่เริ่มค่อนตะวันออก ใกล้ทะเลเมดิเตอร์เรเนียน) จับได้ในที่คนอยู่เยอะ ในความดันอากาศน้อยกว่าทั่วไปเล็กน้อย อุณหภูมิร้อนนิดหน่อย ความเร็วลมกลางๆ ทิศลมเข้าใกล้ 180 (ใต้)
- กลุ่มที่ 1 จะจับได้ในระยะที่ใกล้ Pokestop และ Gym ในช่วง latitude ใกล้ยุโรปและ longtitude ห่างจากมหาสมุทรแอตแลนติกไปทางซ้าย (คาดว่าเป็นโซนใกล้แคนาดา) จับได้ในพื้นที่ที่คนอยู่บ้าง ในความดันอากาศน้อยกว่าทั่วไปเล็กน้อย อุณหภูมิต่ำ ความเร็วลมต่ำเล็กน้อย ทิศลมเข้าใกล้ 270 (ตะวันตก)

- กลุ่มที่ 2 จะจับได้ในระยะที่ใกล้ Pokestop และ Gym ในช่วง latitude ต่ำกว่ายุโรปเล็กน้อยและ longtitude ห่างจากมหาสมุทรแอตแลนติกไปทางซ้าย (คาดว่าเป็นโซนแถบซ้ายของสหรัฐอเมริกา แถวแคลิฟอร์เนีย) จับได้ในพื้นที่ที่คนอยู่บ้าง ในความดันอากาศต่ำ อุณหภูมิสูง ความเร็วลมสูงเล็กน้อย ทิศลมเข้าใกล้ 315 (ตะวันตกเฉียงเหนือ)
- กลุ่มที่ 3 จะจับได้ในระยะที่ไกล Pokestop และ Gym ในช่วง latitude ต่ำกว่ายุโรปมากและ longtitude ห่างจากมหาสมุทรแอตแลนติกไปทางขวา (คาดว่าเป็นโซนโอเชียเนีย/อาเซียน) จับได้ในพื้นที่ที่คนไม่ได้อยู่มากในความดันอากาศกลาง อุณหภูมิค่อนข้างสูง ความเร็วลมกลาง ๆ ทิศลมเข้าใกล้ 225 (ตะวันตกเฉียงใต้)
- กลุ่มที่ 4 จะจับได้ในระยะที่ใกล้ Pokestop และ Gym ในช่วง latitude สูงกว่ายุโรปเล็กน้อยและ longtitude ห่างจากมหาสมุทรแอตแลนติกไปทางขวาเล็กน้อย (คาดว่าเป็นโซนยุโรปตะวันตก แถบสหราชอาณาจักรและนอร์ดิกตอนใต้) จับได้ในพื้นที่ที่มีคนอยู่ทั่วไปบ้าง ในความดันอากาศกลาง อุณหภูมิค่อนข้างต่ำ ความเร็วลมค่อนข้างมาก ทิศลมเข้าใกล้ 315 (ตะวันตกเฉียงเหนือ)
- กลุ่มที่ 5 จะจับได้ในระยะที่ใกล้ Pokestop และ Gym ในช่วง latitude สูงกว่ายุโรปเล็กน้อยและ longtitude ห่างจากมหาสมุทรแอตแลนติกไปทางขวาเล็กน้อย (คาดว่าเป็นโซนยุโรปตะวันตก แถบสหราชอาณาจักรและนอร์ดิกตอนใต้) จับได้ในพื้นที่ที่มีคนอยู่ทั่วไปบ้าง ในความดันอากาศสูงเล็กน้อย อุณหภูมิกลาง ๆ ความเร็วลมค่อนข้างต่ำ ทิศลมเข้าใกล้ 135 (ตะวันออกเฉียงใต้)
- กลุ่มที่ 6 จะจับได้ในระยะที่ใกล้ Pokestop และ Gym ในช่วง latitude ใกล้ยุโรปและ longtitude ห่างจากมหาสมุทรแอตแลนติกไปทางซ้ายเล็กน้อย (คาดว่าเป็นโซนทางขวาของสหรัฐอเมริกา แถบนิวยอร์ก) จับได้ในพื้นที่ที่มีคนอยู่ทั่วไปบ้าง ในความดันอากาศสูงเล็กน้อย อุณหภูมิสูงเล็กน้อย ความเร็วลมกลาง ๆ ทิศลมเข้าใกล้ 0 (เหนือ)
- กลุ่มที่ 7 จะจับได้ในระยะที่ใกล้ Pokestop และ Gym ในช่วง latitude ใต้ยุโรปและ longtitude ห่างจากมหาสมุทรแอตแลนติกไปทางขวา (คาดว่าเป็นโซนโอเชียเนีย/อาเซียน) จับได้ในพื้นที่ที่มีคนอยู่ทั่วไปบ้าง ในความดันอากาศกลาง ๆ อุณหภูมิต่ำเล็กน้อย ความเร็วลมกลาง ๆ ทิศลมเข้าใกล้ 180 (ใต้)
- กลุ่มที่ 8 จะจับได้ในระยะที่ไกล Pokestop และ Gym ในช่วง latitude ใต้ยุโรปและ longtitude ห่างจากมหาสมุทรแอตแลนติกไปทางขวา (คาดว่าเป็นโซนโอเชียเนีย/อาเซียน) จับได้ในพื้นที่ที่มีคนอยู่ค่อนข้างน้อย ในความดันอากาศต่ำ อุณหภูมิสูงมาก ความเร็วลมกลาง ๆ ทิศลมเข้าใกล้ 180 (ใต้)

จาก Heatmap clustering สามารถพบได้ว่า

- latitude และ longtitude มีค่าในทิศตรงข้ามกันเป็นส่วนใหญ่ในแต่ละ cluster คือถ้า latitude มีค่ามาก longtitude จะมีค่าน้อย หรือ latitude มีค่าน้อย longtitude จะมีค่ามาก
- ระยะห่างระหว่างโปเกมอนที่พบกับ Pokestop
  จะมีค่าใกล้เคียงกับระยะห่างระหว่างโปเกมอนที่พบกับ Pokemon gym เสมอในแต่ละ cluster
- ค่าความหนาแน่นของประชากรมักไปในทิศทางเดียวกันในทุกกลุ่ม cluster ยกเว้นกลุ่มที่ 0
- ใน Cluster ส่วนใหญ่แล้วจะเจอโปเกมอนใกล้กับ Pokestop และ Pokemon Gym
  แต่ว่ามีข้อยกเว้นอยู่ที่ Cluster ในช่วงโอเชียเนีย/อาเซียนที่อาจจะไกลกว่า คาดเดาสาเหตุว่าเกิดจาก Pokestop มีน้อยในแถบนอกเมือง แล้วพบโปเกมอนหายากในแถบนั้น

ดังนั้นถ้าอยากจับโปเกมอนหายากในไทย ควรนำเงื่อนไขใน Cluster 7 8 3 มาพิจารณาเพราะเป็น Cluster ที่มี longtitude กับ latitude ใกล้ไทย โดยเฉพาะเงื่อนไข 8 ที่ใกล้เคียงกับไทยมากที่สุด
เพราะประเทศไทยเป็นประเทศที่อุณหภูมิสูงอยู่ตลอดทั้งปี ถ้าต้องการจับในที่ที่มีความหนาแน่นของคนน้อย สามารถลองไปจับโปเกมอนแถวนอกเมือง
อาจจะเป็นสถานที่ท่องเที่ยวทางธรรมชาติที่ไม่ได้มีนักท่องเที่ยวหนาแน่นมากนัก

Conclusion

จากกการสร้างโมเดลจำลองทั้งสองแบบแล้วทำให้ได้วิธีในการตามหาโปเกมอนหายากที่ง่ายขึ้น โดยการจัดทำแบบจำลอง Classification ทำให้ได้รับแบบจำลองที่สามารถนำเงื่อนไขในการจับ (เช่น ความเร็วลม สภาพอากาศ ความหนาแน่นของประชากร ภูมิประเทศ) ในสถานที่ที่เราวางแผนจะไปจับ หลังจากนั้นนำไป Test เพื่อให้รู้ว่าการจับในเงื่อนไขแบบนี้จะมีโอกาสเจอ Rare Pokemon ได้หรือไม่ โดยแบบจำลองนี้จะเป็นแบบจำลองที่อาจจะไม่ได้มีความแม่นยำสูงมาก แต่มีความแม่นยำในระดับหนึ่งที่สามารถจำแนกหา Rare Pokemon จากเงื่อนไขได้ดี หลังจากนั้นจึงนำเงื่อนไขที่ผ่านการตรวจโดยแบบจำลองนั้นมาใช้จับในชีวิตจริง สำหรับแบบจำลองแบบ Clustering ทำให้ได้เห็นกลุ่มก้อนของ Rare Pokemon เช่นกัน โดยเงื่อนไขที่สามารถทำได้ง่ายที่สุดเพราะใกล้เคียงกับประเทศไทย อาจเป็นการจับโปเกมอนห่างจาก Pokestop เล็กน้อย ห่างจาก Pokemon gym และจับในที่ที่มีคนไม่ได้หนาแน่นมาก อาจจะเป็นแหล่งท่องเที่ยวตามธรรมชาติในต่างจังหวัด เป็นต้น การทำแบบจำลองทั้งคู่ทำให้ได้รับผลลัพธ์ที่ต่างกัน มีข้อดีข้อเสียต่างกัน แต่ว่าหากสามารถนำผลลัพธ์ที่ได้จากทั้งคู่มาใช้ร่วมกันอาจทำให้สามารถตามหาโปเกมอนหายากได้ง่ายขึ้นกว่าเดิม เช่น สร้างเงื่อนไขที่คล้ายจากการ Clustering เพื่อให้แบบจำลอง Classification ตรวจ แต่อย่างไรก็ตามนอกเหนือจากการพยายามตามหาโปเกมอนหายากแล้ว Pokemon Trainer ทุกท่านที่ได้ทดสอบเงื่อนไขควรระมัดระวังตัวในการจับโปเกมอน ให้จับในที่ที่ปลอดภัย และไม่ควรลืมที่จะหาความสนุกไปกับการจับโปเกมอนระหว่างเส้นทางการเป็น Pokemon Trainer อันดับหนึ่งต่อไป

Reference

Dataset การปรากฏตัวของโปเกมอนในเกม Pokemon Go ในปี 2016

https://www.kaggle.com/datasets/semioniy/predictemall

Dataset ข้อมูล Pokemon ตาม pokemonid

https://www.kaggle.com/datasets/maca11/all-pokemon-dataset/data/

Data Pokemon ระดับความหายากของโปเกมอนแต่ล่ะตัวในเกม Pokemon GO

https://web.archive.org/web/20240203201627/https://www.pokego.org/rare-pokemon-list/