

REGIONS OF TAIWAN

YUN LIN

TABLE OF CONTENT

- ▶ DATA COLLECT
 - ▶ CLUSTER THE REGION
 - ▶ OPTIMAL THE K-MEAN
 - ▶ GENERATE MAP
 - ▶ FIND 3 MOST COMMON VENUE AT EACH REGION
- 
- Several white lines of varying lengths and slopes are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.

SIMPLY SAY THERE IS ONLY 5 REGIONS AT TAIWAN

2.2.1 Select Area column

```
: df_tw.reset_index(drop=True, inplace=True)
df_tw = df_tw[['Present divisions']]
df_tw.rename(columns={'English': 'Area'}, inplace=True)
df_tw.head()
```

21]:

	Present divisions
0	Taipei, New Taipei, Keelung, Taoyuan, Hsinchu ...
1	Miaoli, Taichung, Changhua, Nantou
2	Yunlin, Chiayi City/County, Tainan
3	Kaohsiung, Pingtung, Penghu
4	Hualien, Taitung

PROVIDE THEIR LATITUDE AND LONGITUDE BY GOOGLE MAP API

```
3]: import requests

def get_coordinates(api_key, address, verbose=False):
    try:
        url = 'https://maps.googleapis.com/maps/api/geocode/json?key={}&address={}'.format(api_key, address)
        response = requests.get(url).json()
        if verbose:
            print('Google Maps API JSON result =>', response)
        results = response['results']
        geographical_data = results[0]['geometry']['location'] # get geographical coordinates
        lat = geographical_data['lat']
        lon = geographical_data['lng']
        return [lat, lon]
    except:
        return [None, None]
```

	Present divisions	Latitude	Longitude
0	Taipei, New Taipei, Keelung, Taoyuan, Hsinchu ...	24.813829	120.967480
1	Miaoli, Taichung, Changhua, Nantou	24.163165	120.674669
2	Yunlin, Chiayi City/County, Tainan	23.480075	120.449111
3	Kaohsiung, Pingtung, Penghu	22.627278	120.301435
4	Hualien, Taitung	23.991073	121.611195

MAP(TRIED BUT SHOULD LOOKS LIKE THAT)

4.1.2 Generate the Map

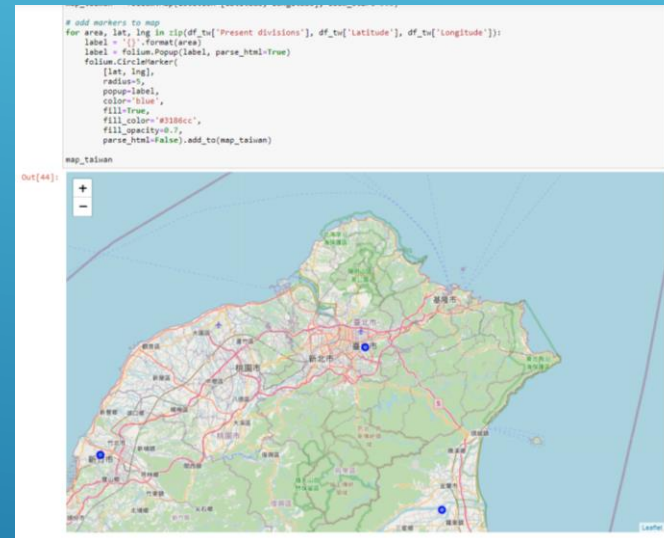
```
# create map of Taiwan using Latitude and Longitude values
map_taiwan = folium.Map(location=[latitude, longitude], zoom_start=9.5)

# add markers to map
for area, lat, lng in zip(df_kw['Present divisions'], df_tw['Latitude'], df_tw['Longitude']):
    label = '{}'.format(area)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_kuwait)
```

map_taiwan

```
-----
NameError                                Traceback (most recent call last)
<ipython-input-33-3041731c237a> in <module>
      1
      2 # create map of Kuwait using latitude and longitude values
----> 3 map_taiwan = folium.Map(location=[latitude, longitude], zoom_start=9.5)
      4
      5 # add markers to map

NameError: name 'folium' is not defined
```



NUMBERS OF VENUE AT EACH AREA AND TOTAL UNIQUE CATEGORIES

5.2 How many venues were returned for each neighborhood

```
: taiwan_venues.groupby('Area').count()
```

63]:

	Area Lat	Area Long	Venue	Venue Lat	Venue Long	Category
Area						
(Penghu), Kinmen, Matsu (Lienchiang)	8	8	8	8	8	8
(Yilan), Hualien, Taitung	46	46	46	46	46	46
Hualien, Taitung	60	60	60	60	60	60
Kaohsiung, Pingtung, (Penghu)	100	100	100	100	100	100
Kaohsiung, Pingtung, Penghu	100	100	100	100	100	100
Kinmen, Matsu (Lienchiang)	6	6	6	6	6	6
Miaoli, Taichung, Changhua, Nantou	100	100	100	100	100	100
Taichung, Changhua, Nantou	100	100	100	100	100	100
Taipei, New Taipei, Keelung, (Yilan)	100	100	100	100	100	100
Taipei, New Taipei, Keelung, Taoyuan, Hsinchu City/County, Yilan	95	95	95	95	95	95
Taoyuan, Hsinchu City/County, Miaoli	95	95	95	95	95	95
Yunlin, Chiayi City/County, Tainan	94	94	94	94	94	94

5.2.1 See how many unique Categories Taiwan have

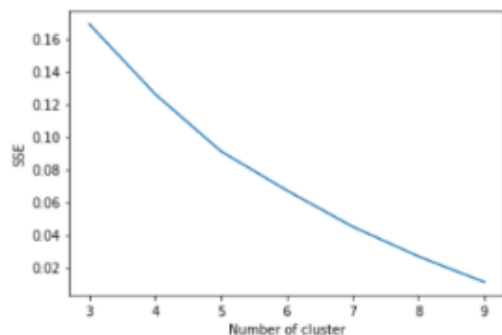
```
: print('There are {} uniques categories.'.format(len(taiwan_venues['Category'].unique())))
```

There are 131 uniques categories.

K-MEAN

```
In [76]: sse = {}  
list_1 = []  
list_2 = []  
sil = pd.DataFrame()  
  
for k in range(3,10):  
    kmeans = KMeans(n_clusters=k, max_iter=10000, random_state=0).fit(taiwan_grouped_clustering)  
    taiwan_grouped_clustering["Clusters"] = kmeans.labels_  
    #print(data["clusters"])  
    sse[k] = kmeans.inertia_ # Inertia: Sum of distances of samples to their closest cluster center  
    label = kmeans.labels_  
    sil_coeff = silhouette_score(taiwan_grouped_clustering, label, metric='euclidean')  
    list_1.append(k)  
    list_2.append(sil_coeff)  
  
sil['k'] = list_1  
sil['sil_coeff'] = list_2  
highest = sil.sort_values(['sil_coeff'], ascending=False).head(1)  
print('The best k is {} with a silhouette score of {}'.format(highest['k'].values, highest['sil_coeff'].values))  
plt.figure()  
plt.plot(list(sse.keys()), list(sse.values()))  
plt.xlabel("Number of cluster")  
plt.ylabel("SSE")  
plt.show()
```

The best k is [3] with a silhouette score of [0.68934177]



```
In [77]: kclusters = 4  
  
# run k-means clustering  
kmeans = KMeans(n_clusters=kclusters, random_state=0, max_iter=10000).fit(taiwan_grouped_clustering)
```


```
In [78]: kmeans.labels_[0:10]
```

```
Out[78]: array([2, 1, 3, 0, 0, 2, 2, 1, 2, 0])
```

TOP 3 COMMON AT ALL AREA

	Area	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue
0	(Penghu), Kinmen, Matsu (Lienchiang)	Hotel	Department Store	Tourist Information Center
1	(Yilan), Hualien, Taitung	Bakery	Asian Restaurant	Park
2	Hualien, Taitung	Hotel	Chinese Restaurant	Hostel
3	Kaohsiung, Pingtung, (Penghu)	Coffee Shop	Hotel	Taiwanese Restaurant
4	Kaohsiung, Pingtung, Penghu	Coffee Shop	Hotel	Taiwanese Restaurant
5	Kinmen, Matsu (Lienchiang)	Airport	Noodle House	Historic Site
6	Miaoli, Taichung, Changhua, Nantou	Hotel	Chinese Restaurant	Café
7	Taichung, Changhua, Nantou	Hotel	Chinese Restaurant	Café
8	Taipei, New Taipei, Keelung, (Yilan)	Hotel	Café	Bakery
9	Taipei, New Taipei, Keelung, Taoyuan, Hsinchu ...	Chinese Restaurant	Convenience Store	Coffee Shop
10	Taoyuan, Hsinchu City/County, Miaoli	Chinese Restaurant	Convenience Store	Coffee Shop
11	Yunlin, Chiayi City/County, Tainan	Taiwanese Restaurant	Hotel	Park

CONCLUSION

- ▶ Use google map api to find location then use foursquare to restore the venue of each area
 - ▶ Then find the top 3 common venue
 - ▶ K-mean gives better prediction and recommendation place
 - ▶ Clustering areas will give equally distance for the customer so they wouldn't search some venue that's way too far
- 
- Several white lines of varying lengths and slopes are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.