

## **METODY EKSPŁORACJI DANYCH**

*PROJEKT: "Możliwości lokalizacji obiektów na podstawie geolokalizacji wpisów  
na portalach społecznościowych na przykładzie lokalizacji lotnisk z serwisu  
społecznościowego Twitter"*

**Aleksandra Knapik**

**Julita Musiał**

GiK 1 rok MSU

geoinformatyka

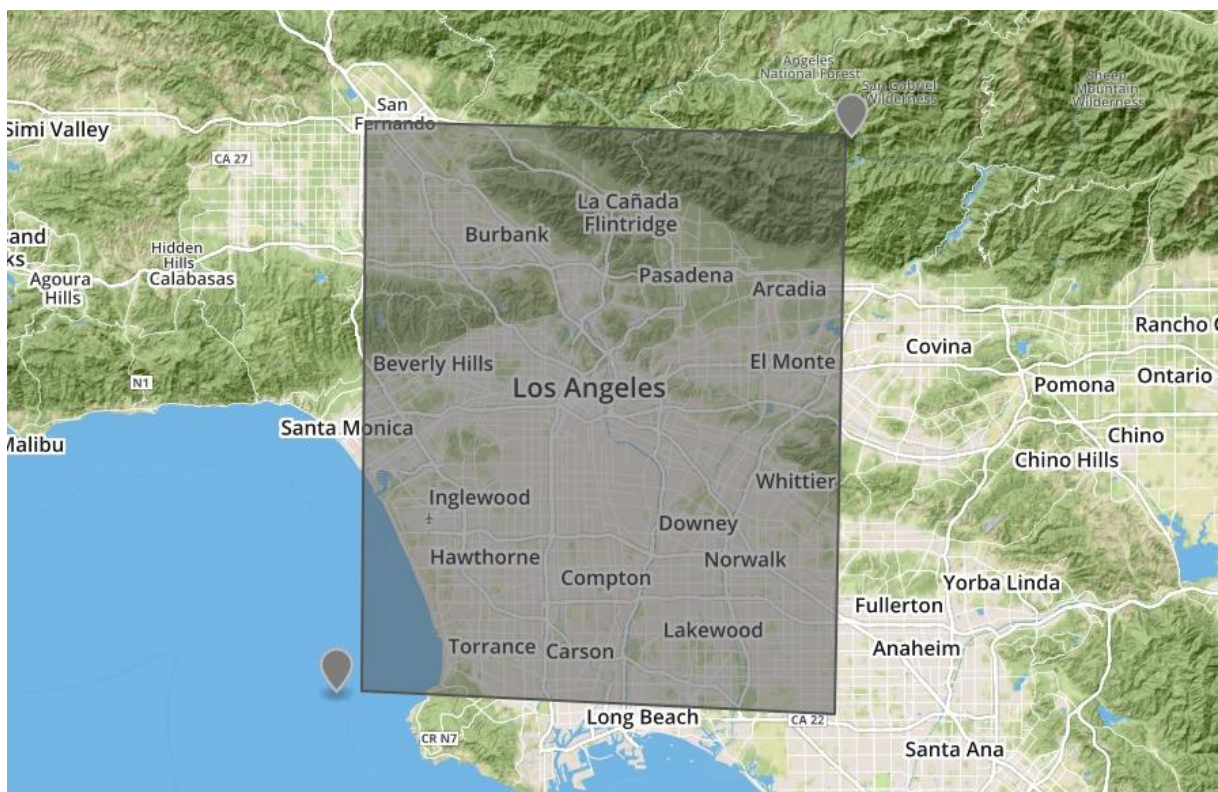
Wrocław, 01.02.2015r.

### 1. Opis i cel projektu:

Celem projektu jest sprawdzenie możliwości lokalizacji obiektów na podstawie geolokalizacji wpisów na portalach społecznościowych. W projekcie analizowano wpisy na portalu społecznościowym Twitter w celu określenia lokalizacji lotnisk.

### 2. Dane źródłowe:

Do wykonania analiz konieczne było pobranie wpisów z serwisu społecznościowego Twitter. Wykorzystano w tym celu Streaming API udostępniony przez serwis Twitter. Pobierane wiadomości były filtrowane w momencie zapisu ze względu na lokalizację. Obszar badawczy to miasto **Los Angeles** w Stanach Zjednoczonych Ameryki Północnej. Obszar został ograniczony BBoxem: -118.49853515625, 33.796267541909884, -117.97805786132811, 34.260621527140444.



Dane do sprawdzenia poprawności lokalizacji lotnisk wykorzystane będą dane z serwisu OpenStreetMap.

### 3. Format danych wejściowych:

Pobrane wpisy z serwisu społecznościowego Twitter zostały zapisane w postaci pliku tekstowego .txt w formacie JSON. Plik tekstowy ma rozmiar ok 400MB.

```

16 {"created_at":"Fri Jan 09 23:00:02 +0000 2015","id":553687671382765568,"id_str"
: "553687671382765568","text":"BONG! BONG! BONG! BONG! BONG!","source":"\u003ca
href=\"https://en.wikipedia.org/wiki/Big_Ben\" rel=\"nofollow\"\u003eFaultyBigB
en\u003c/a\u003e","truncated":false,"in_reply_to_status_id":null,"in_reply_to_stat
us_id_str":null,"in_reply_to_user_id":null,"in_reply_to_user_id_str":null,"in_reply
_to_screen_name":null,"user":{"id":1330109264,"id_str":"1330109264","name":"Faulty
Big Ben Clock","screen_name":"FaultyBigBen","location":"Westminster, London","url"
:null,"description":"I am wrong.","protected":false,"verified":false,"followers_cou
nt":126,"friends_count":0,"listed_count":1,"favourites_count":0,"statuses_count"
:13573,"created_at":"Fri Apr 05 21:22:52 +0000 2013","utc_offset":0,"time_zone"
:"London","geo_enabled":true,"lang":"en","contributors_enabled":false,"is_translato
r":false,"profile_background_color":"CODEED","profile_background_image_url":"http://
pbs.twimg.com/profile_background_images/378800000008520159/d6ace022ff4650c8cdf23df98369f861
.jpeg","profile_background_image_url_https":"https://pbs.twimg.com/
profile_background_images/378800000008520159/d6ace022ff4650c8cdf23df98369f861
.jpeg","profile_background_tile":true,"profile_link_color":"0084B4","profile_sideba
r_border_color":"000000","profile_sidebar_fill_color":"DDEEF6","profile_text_color"
:"333333","profile_use_background_image":true,"profile_image_url":"http://pbs
.twimg.com/profile_images/3709555224/8e90e8045390de6bd838b8f328eee2dd_normal
.jpeg","profile_image_url_https":"https://pbs.twimg.com/profile_images/37095552
24/8e90e8045390de6bd838b8f328eee2dd_normal.jpeg","default_profile":false,"default_
profile_image":false,"following":null,"follow_request_sent":null,"notifications"
:null},"geo":{"type":"Point","coordinates":[51.500753,-0.124680]},"coordinates"
:{"type":"Point","coordinates":[-0.124680,51.500753]},"place":{"id":"457b4814b4240d
87","url":"https://api.twitter.com/1.1/geo/id/457b4814b4240d87.json","place_t
ype":"city","name":"London","full_name":"London, England","country_code":"GB"
,"country":"United Kingdom","bounding_box":{"type":"Polygon","coordinates":[[[-0
.187894,51.483718],[-0.187894,51.516465],[-0.109978,51.516465],[-0.109978,51
.483718]]]},"attributes":{},"contributors":null,"retweet_count":0,"favorite_count"
:0,"entities":{"hashtags":[],"trends":[],"urls":[],"user_mentions":[],"symbols":[]
},"favorited":false,"retweeted":false,"possibly_sensitive":false,"filter_level"
:"medium","lang":"tl","timestamp_ms":"1420844402534"}

```

#### 4. Wykorzystane technologie:

- język: Python
- baza danych: mongoDB
- biblioteki: tweepy, pandas, matplotlib, NumPy, SciPy, PyMongo, PyQt4
- program QGIS Brighton 2.6.1

Do pobrania danych z serwisu Twitter i ich analizy został wykorzystany język **Python**. Aby możliwa była komunikacja z API Twittera konieczna jest również biblioteka tweepy. Dodatkowo wykorzystano również bibliotekę pandas (do analizy danych i obliczeń statystycznych) oraz matplotlib (do rysowania wykresów).

Ze względu na duży rozmiar pliku JSON z pobranymi tweetami konieczne było wykorzystanie bazy danych. Dla celów tego projektu zastosowano bazę danych **MongoDB**. MongoDB jest przykładem nierelacyjnego systemu zarządzania bazą danych. Dane składowane są jako dokumenty typu JSON w kolekcjach. Podobnie jak w relacyjnej bazie danych możliwe jest wykonywanie zapytań, co zostanie wykorzystane do wstępnej filtracji danych.



Pobrane wpisy z serwisu Twitter zostały zgromadzone w bazie danych MongoDB. Do połączenia z bazą danych MongoDB konieczna jest także biblioteka PyMongo, zawierająca zestaw narzędzi do pracy z bazą danych.

Do klasteryzacji obiektów zostały wykorzystane funkcje z biblioteki NumPy oraz SciPy.

Końcowy efekt lokalizacji lotnik został zwizualizowany w autorskiej wtyczce do programu **QGIS Brighton 2.6.1**.

## **5. Dostęp do danych:**

Dostęp do danych umożliwia Streaming API. Aby móc pobierać tweet'y należy posiadać konto na portalu Twitter, a następnie stworzyć swoją aplikację na stronie [www.apps.twitter.com](http://www.apps.twitter.com). Wygenerowane zostają wówczas kody dostępu i kody autoryzacyjne niezbędne podczas tworzenia aplikacji:

- consumer key
- consumer secret
- access token
- access secret

Za streaming tweet'ów odpowiadają odpowiednie metody i klasy z biblioteki tweepy.

Możliwa jest parametryzacja zapytań streamu. Podając odpowiedni filtr, można pobierać tylko wybrane tweet'y. Przykłady parametrów:

- język twitta (language)
- osoby, które się śledzi (follow)
- słowa kluczowe (track)
- lokalizacja (locations)
- odpowiedzi (replies)

W celu pobrania wpisów z serwisu Twitter został napisany skrypt w języku Python.

```
# -*- coding: cp1250 -*-  
#Stream do pobierania Tweetów dla celów lokalizacji lotnisk w LA  
  
#Import the necessary methods from tweepy library  
from tweepy.streaming import StreamListener  
from tweepy import OAuthHandler  
from tweepy import Stream  
  
#Variables that contains the user credentials to access Twitter API  
access_token = "  
access_token_secret = "  
consumer_key = "  
consumer_secret = "  
  
#This is a basic listener that just prints received tweets to stdout.  
class StdOutListener(StreamListener):  
  
    def on_data(self, data):  
        print data  
        return True  
  
    def on_error(self, status):  
        print status  
  
if __name__ == '__main__':  
  
    #This handles Twitter authentication and the connection to Twitter Streaming API  
    l = StdOutListener()  
    auth = OAuthHandler(consumer_key, consumer_secret)  
    auth.set_access_token(access_token, access_token_secret)  
    stream = Stream(auth, l)  
  
    #This line filter Twitter Streams to capture data by the location of LA  
    stream.filter(locations = [-118.49853515625, 33.796267541909884, -117.97805786132811, 34.260621527140444])
```

## 6. Etapy opracowania projektu:

Wykonanie projektu wymagało dzielenia go na kilka etapów. Do głównych etapów należy zaliczyć:

- pobranie danych
- dodanie danych do bazy danych MongoDB
- zapis współrzędnych lokalizujących tweety
- klasteryzacja danych
- wizualizacja danych w programie QGIS

## Szczegółowy opis etapów opracowania projektu:

### **1. Wybór obszaru badawczego**

Obszar badawczy to miasto **Los Angeles** w Stanach Zjednoczonych Ameryki Północnej.

### **2. Pobranie danych za pomocą Streamu do pliku JSON dla zadanego BBoxa**

Dane pobierane były w godzinach 15:00 - 10:00 czasu polskiego UTC+1 (8:00 - 24:00 czasu UTC-8) oraz 22:00 - 7:00 czasu polskiego (UTC+1) (14:00 - 23:00 czasu UTC-8). W tym czasie pobrane zostały pliki testowe o łącznym rozmiarze ok. 300 MB.

### **3. Dodanie danych do bazy danych MongoDB**

Zaimportowano dane do bazy MongoDB. Kolekcja zawiera 115 743 dokumenty.

### **4. Wstępna filtracja danych:**

#### **4.1. usunięcie tweet'ów nie posiadających geolokalizacji**

Z kolekcji usunięto 21 376 dokumentów nie posiadających geolokalizacji. Do dalszych analiz wykorzystano 94 367 dokumentów.

#### **4.2. filtracja danych ze względu na słowa kluczowe: "airport", "flight", "plane".**

##### Wystąpienie słów kluczowych:

airport:	225
plane:	95
flight :	83

Sprawdzono również wystąpienie innych słów kluczowych, ale nie zostały one znalezione. Ponadto zdecydowano się ograniczyć liczbę słów kluczowych do minimum, by uzyskać jak najbardziej trafione wpisy.

### **5. Zapisanie dokumentów, które spełniają zadane wyżej kryteria i zapisanie ich do nowej kolekcji**

Do nowej kolekcji zapisano łącznie 403 obiekty, zawierające słowa kluczowe.

### **6. Połączenie się z bazą danych MongoDB za pomocą języka Python i analizy statystyczne**

Połączenie z bazą danych zostało zrealizowane w nowym skrypcie.

```
#ustanowienie klienta bazy MongoDB
client = MongoClient('localhost', 27017)

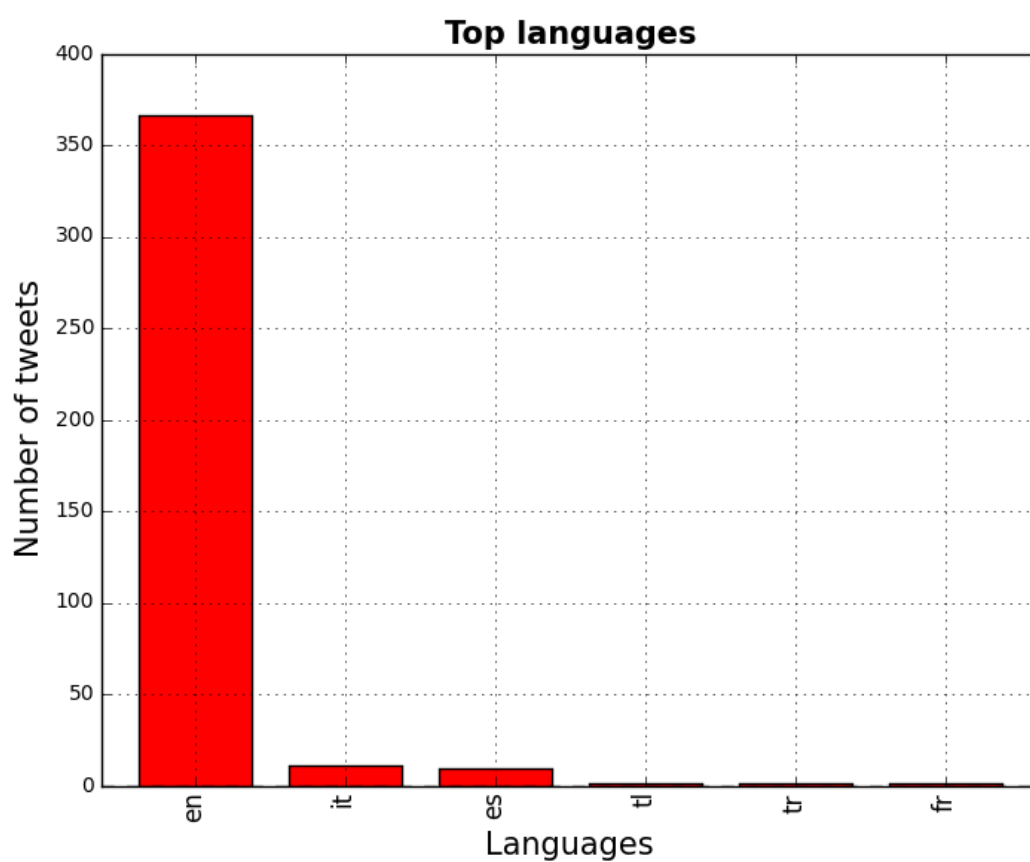
#dostęp do bazy danych
mydb = client.twitter_lot

#dostęp do kolekcji
myCollection = mydb.lotWszystkieKW
```

Zostały wykonane analizy statystyczne dotyczące uzyskanych wpisów.

```
airport: 225  
flight: 83  
plane: 95
```

```
Łączna liczba tweetów: 389  
Languages of tweets  
en      366  
it       11  
es        9  
tl         1  
tr         1  
fr         1
```



## 7. Zapis współrzędnych tweetów do pliku tekstowego

Zapisane współrzędne tweetów posiadających słowa kluczowe, będą danymi wejściowymi do procesu klasteryzacji lotnisk.

```
#dodaje kolekcję do tablicy
keyWords = []
for item in myCollection.find():
    keyWords.append(item)
print 'keyWords:', len(keyWords)

#zapisuje współrzędne do pliku txt
keyWords2 = []
lotFile = open('lotniskaWsp_OK2.txt', 'w')
for i in range(0, len(keyWords)):
    keyWords2 = str(keyWords[i]['geo']['coordinates']) + '\n'
    print keyWords2
    lotFile.writelines(keyWords2)
print 'keyWords2:', len(keyWords2)
lotFile.close()
```

## 8. Klasteryzacja danych

Po wyłuskaniu z bazy danych współrzędnych tweetów o odpowiedniej tematyce i zapisaniu ich do pliku, przystąpiono do wyszukania na ich podstawie współrzędnych lotnisk. W tym celu wykonano klasteryzację punktów wejściowych przy użyciu autorskiej aplikacji, napisanej w języku Python.

Klasteryzacja (grupowanie) jest jedną z metod nienadzorowanej (bez dostępnej a priori wiedzy) analizy danych. Nie wiadomo bowiem nic na temat tego ile lotnisk mamy odnaleźć ani w jakiej lokalizacji ich szukać. Możemy jedynie przypuszczać, że znajdują się one tam, gdzie występują największe skupiska tematycznych tweetów. Głównym celem klasteryzacji jest podział rozpatrywanego zbioru obiektów na grupy (klastry), w ten sposób, aby każda z grup była możliwie jednorodna, tzn. zawierała elementy najbliższe do siebie (w rozumieniu przestrzeni cech, u nas - współrzędne), a jednocześnie poszczególne klastry były jak najbardziej zróżnicowane między sobą.

Do zadania została wykorzystana **klasteryzacja hierarchiczna**. Jest to rodzaj klasteryzacji pozwalający na podział danego zbioru na nieokreśloną z góry liczbę podzbiorów. A my nie wiemy początkowo ile jest lotnisk w danym miejscu. Techniki klasteryzacji hierarchicznej można podzielić na dwie grupy:



- Techniki skupiające (gromadzące)

Stosując technikę skupiającą zaczynamy od pojedynczych obiektów tworzących klastry (w których same są jedynym elementem), a następnie w każdym kroku, łączymy dwa klastry, aż do momentu uzyskania jednej grupy skupiającej wszystkie obiekty.

- Techniki dzielące

Przy technice dzielącej rozpoczynamy od zdefiniowania jednego klastra, do którego należą wszystkie dane wejściowe. W kolejnych krokach dokonujemy podziału aż każdy element wejściowy (każdy obiekt) sam będzie stanowił klaster (będzie jedynym elementem należącym do tego klastra).

Wynik klasteryzacji hierarchicznej przedstawiany jest zazwyczaj w postaci drzewa zwanego dendrogramem. Dendrogram jest wykresem dendrytowym pokazującym jak poszczególne klastry są ze sobą związane. Liście dendrogramu są elementami wejściowymi, a korzeń jest końcowym wynikiem klasteryzacji łączącym wszystkie liście. Rozgałęzienie w tym drzewie występuje w punkcie, w którym są łączone dwa klastry (lub jeden klaster podzielony na dwa – przy klasteryzacji dzielącej). Przez „ucięcie” dendrogramu na określonym poziomie, otrzymujemy zbiór rozłącznych grup (klastrow).

### Algorytm i jego sposób implementacji

Zadanie rozpoczęto od wczytania danych, powstałych jako efekt filtracji przestrzennej i selekcji tematycznej, a następnie zbudowano macierz z tymi danymi.

```
inFile = open('daneLotniskaOK.txt','r')
for line in inFile:
    data = line.strip().split('\t')
    rowHeaders.append(data[0])
    dataMatrix.append([float(x) for x in line.strip().split()[1:]])
```

Klasteryzacja hierarchiczna przeprowadzana jest na podstawie pewnej miary podobieństwa obiektów. W następnym etapie nasz algorytm buduje macierz zawierającą odległości pomiędzy kolejnymi parami obiektów, stosując metrykę euklidesowa:

```
distanceMatrix = dist.pdist(dataMatrix)
```

Mała wartość w tabeli odległości pozwala przypuszczać, że te dwa klastry/obiekty są bardziej podobne do siebie niż inne klastry/obiekty z większą wartością miary odległości. Podczas wykonywania klasteryzacji techniką skupiania skanujemy macierz odległości w poszukiwaniu najmniejszej odległości. Element  $a(i,j)$ , będący najmniejszą wartością w macierzy odległości, wyznacza nam dwa klastry (pierwszy określony przez numer rzędu 'i', a drugi określony przez numer kolumny 'j'), które zostaną złączone. W ten sposób powstanie nowa grupa, składająca się z dwóch połączonych ze sobą klastrow. Kolejnym etapem jest uaktualnienie macierzy odległości. Rzędy i kolumny odpowiadające połączonym klastrom są usuwane a na ich miejsce wprowadzony jest jeden rząd i jedna kolumna odpowiadające nowo powstałej grupie. Konieczne jest w tym miejscu określenie w jaki sposób będziemy określać odległość pomiędzy dwoma klastrami (skupiskami obiektów). Najczęściej stosowane metody to:

- **Pojedyncze wiązanie (*single linkage*)**

W tej metodzie łączenie grup opiera się na odległości pomiędzy najbliższymi elementami należącymi do łączonych klastrow. Grupy z najmniejszą odległością pomiędzy ich najbliższymi elementami są łączone jako pierwsze. Rozpoczynamy od grup o rozmiarze 1 (każdy obiekt tworzy jeden klaster), a następnie, po każdym połączeniu zmniejszamy liczbę grup o 1. Znana także jako metoda najbliższego sąsiada.

- **Pełne wiązanie (*complete linkage*)**

W tej metodzie używamy odległości pomiędzy najbardziej odległymi elementami należącymi do grup, aby zdecydować które z dwóch klastrow połączyć jako pierwsze.

- **Wiązanie średnich (*average linkage*)**

W tym wypadku, odległość między klastrami definiujemy jako średnią odległość pomiędzy wszystkimi parami elementów należących do obu grup. Metodę tę określa się skrótem UPGMA (*unweighted pair group method with arithmetic mean*). Jeśli użyjemy wag w postaci wielkości klastrow (liczby znajdujących się w nich elementów) otrzymamy metodę WPGMA (*weighted pair group method with arithmetic mean*)

- **Metoda centroidów**

Dla każdej z grup obliczamy centroid – jako wartość średnią wszystkich obiektów (wektorów) należących do danej grupy. Odległość między klastrami jest definiowana jako odległość między centroidami tych klastrow. Metodę tę określa się skrótem UPGMC (*unweighted pair group*

*method centroid*). Istnieje również metoda uwzględniająca, w postaci odpowiednich wag, wielkość klastrow. Określa się ją skrótem WPGMC (*weighted pair group method centroid*).

Do procesu odpowiedniego łączenia danych wykorzystano metodę linkage, która zapewnia budowę planu klasteryzacji danych w kolejne grupy. Jako parametr podawano różne metody, następnie je wizualizując, by sprawdzić ich skuteczność.

```
linkage = hier.linkage(distanceSquareMatrix, method='centroid',  
                      metric='euclidean')
```

Ostatnim istotnym krokiem jest określenie kryterium łączenia oraz progu dla danej metody, a także wyprodukowanie macierzy końcowej, uwzględniającej te wymagania. Uczynić to można w następujący sposób:

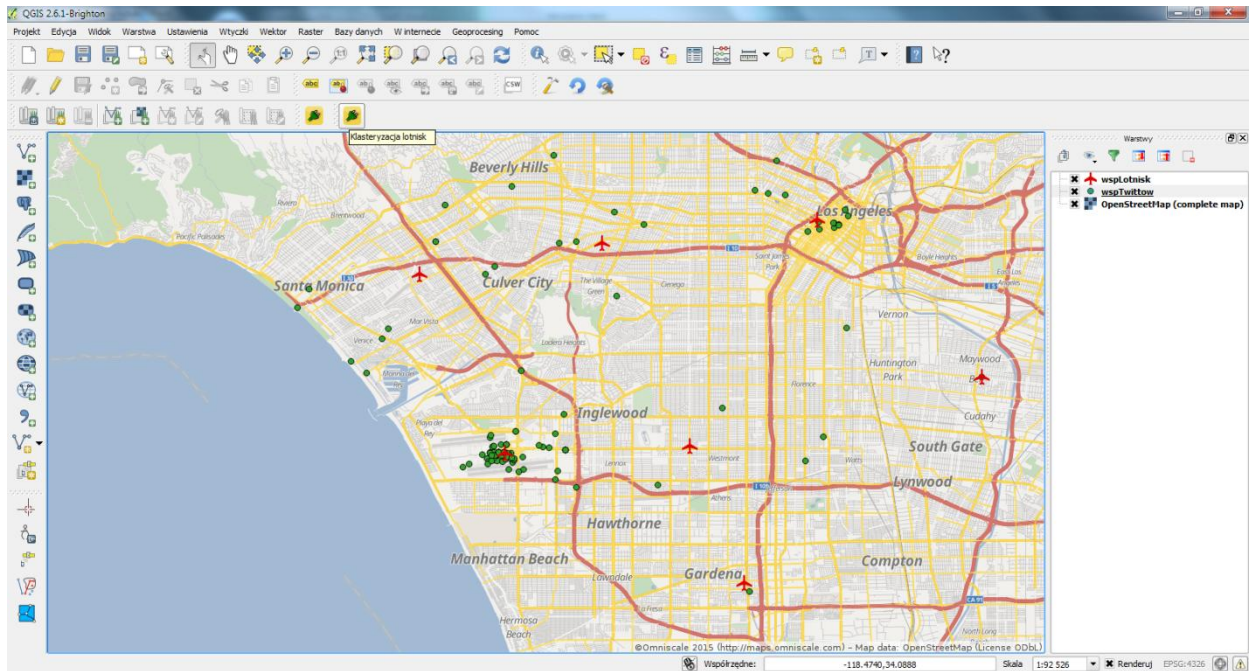
```
ccc = hier.fcluster(linkage, 2, criterion = 'distance')
```

W badaniu uwzględniono dwa rodzaje kryterium łączenia:

- *łączenie względem wartości niespójności (inconsistent)*, polegające na tym, że klastry klasyfikowane są do takiego klastra potomnego, by wartość niespójności w tym klastrze była mniejsza lub równa zadanemu progowi
- *odległości (distance)*, polegające na tym, że obserwacje w każdym klastrze potomnym nie mają większej odległości niż wartość progowa w określonej metryce.

Na końcu obliczane jest położenie centrum geograficznego dla każdego klastra, które reprezentuje miejsce, gdzie przypuszczalnie powinno znajdować się lotnisko.

## 9. Wizualizacja danych w programie QGIS Brighton 2.6.1 z wykorzystaniem autorskiej wtyczki Klasteryzacja



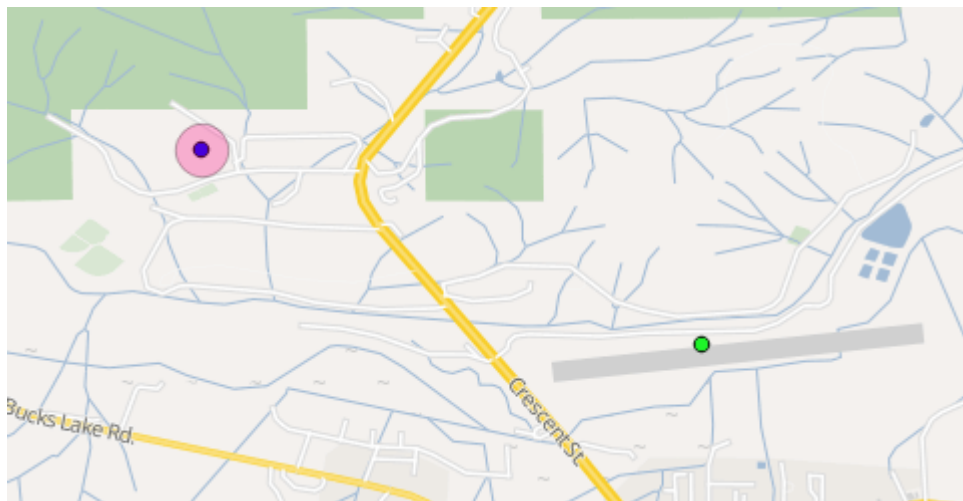
## 10. Analiza skuteczności działania algorytmów klasteryzacji

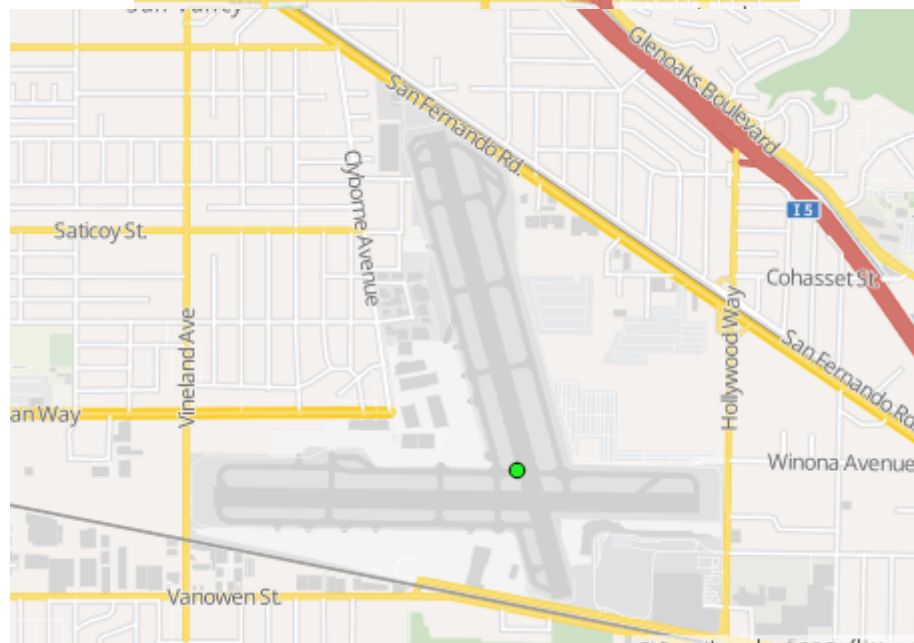
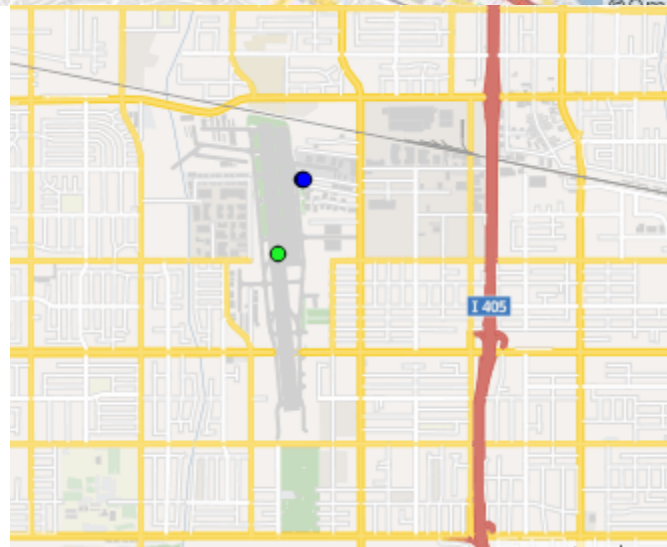
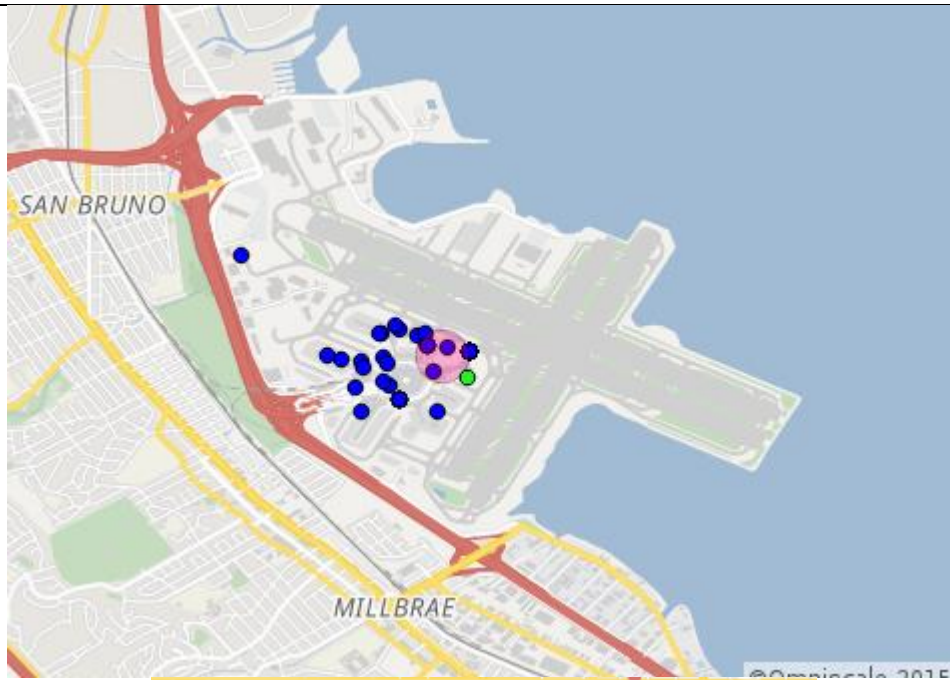
METODA	PRÓG	KRYTERIUM	KLASTRÓW WYNIKOWYCH	ILOŚĆ TRAFIONYCH	SKUTECZNOŚĆ
Pojedyncze wiązanie	1	distance	24	5/9	55%

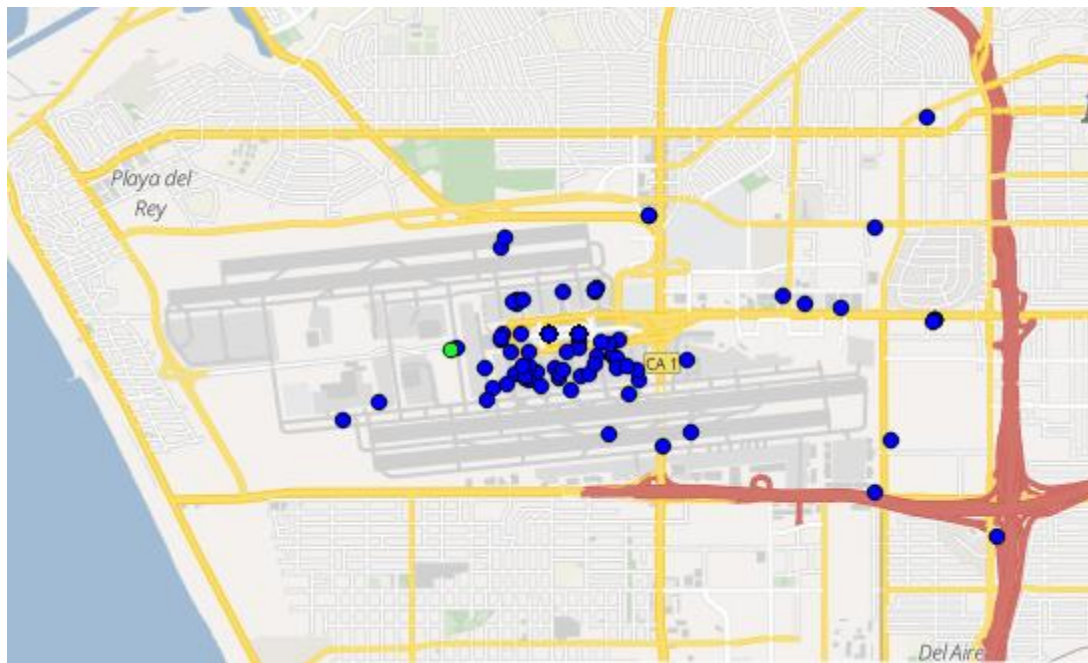
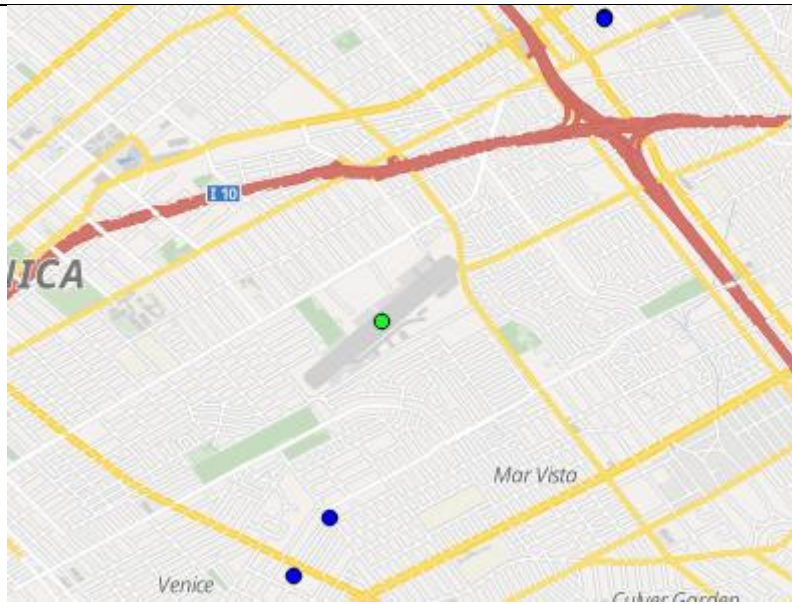
**Niebieski punkt** - wpis z Twittera

**Zielony punkt** - lotnisko

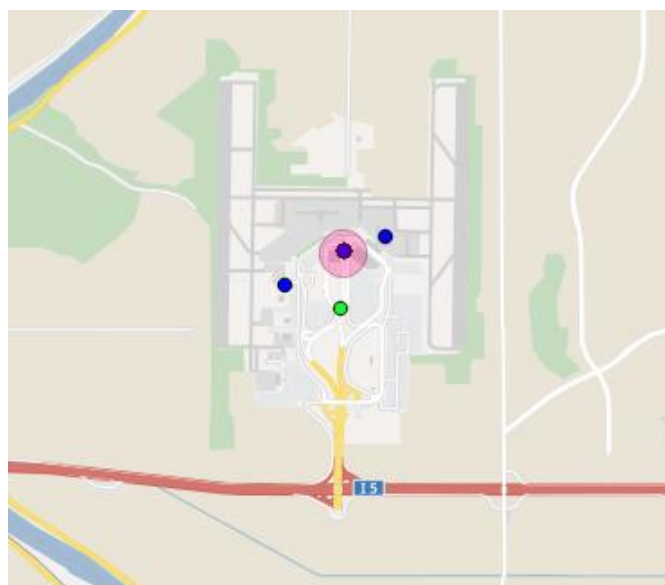
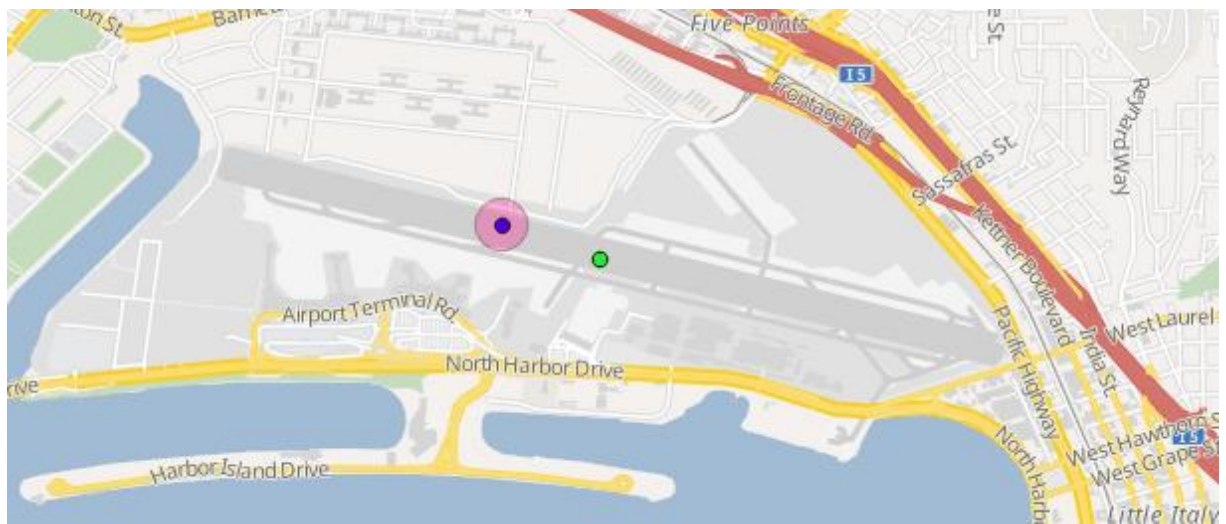
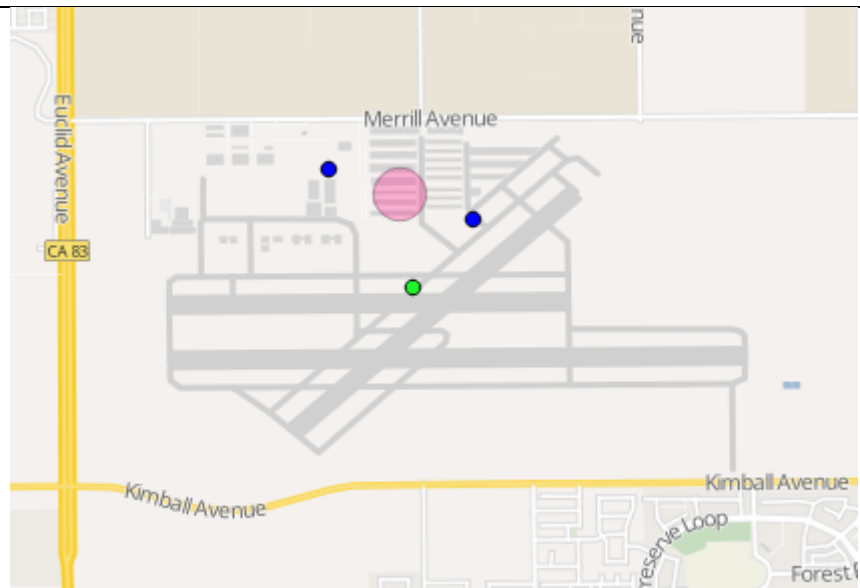
**Różowy punkt** - sklasteryzowane lotnisko


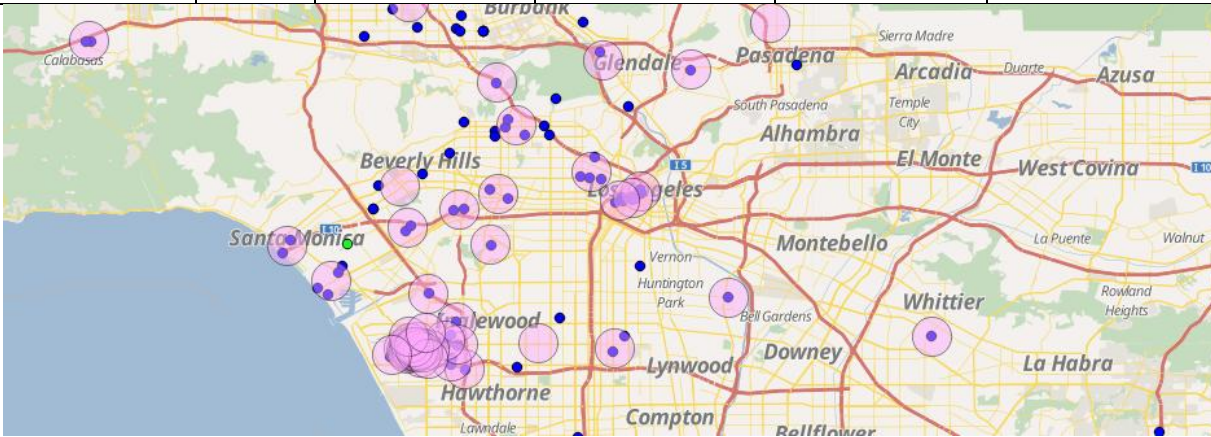
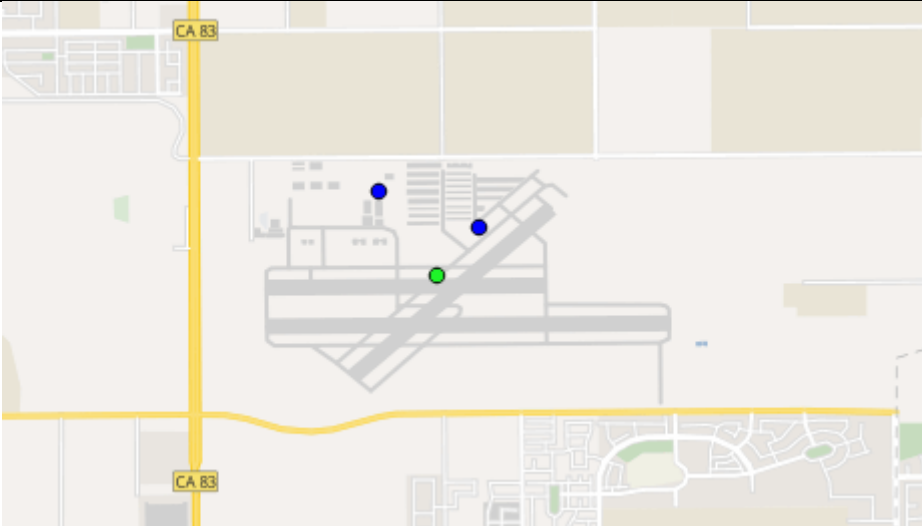




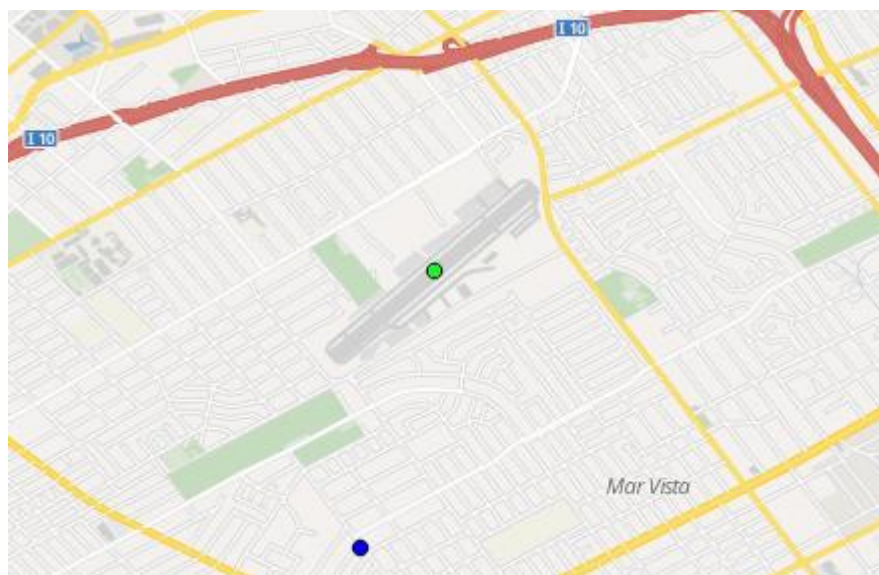
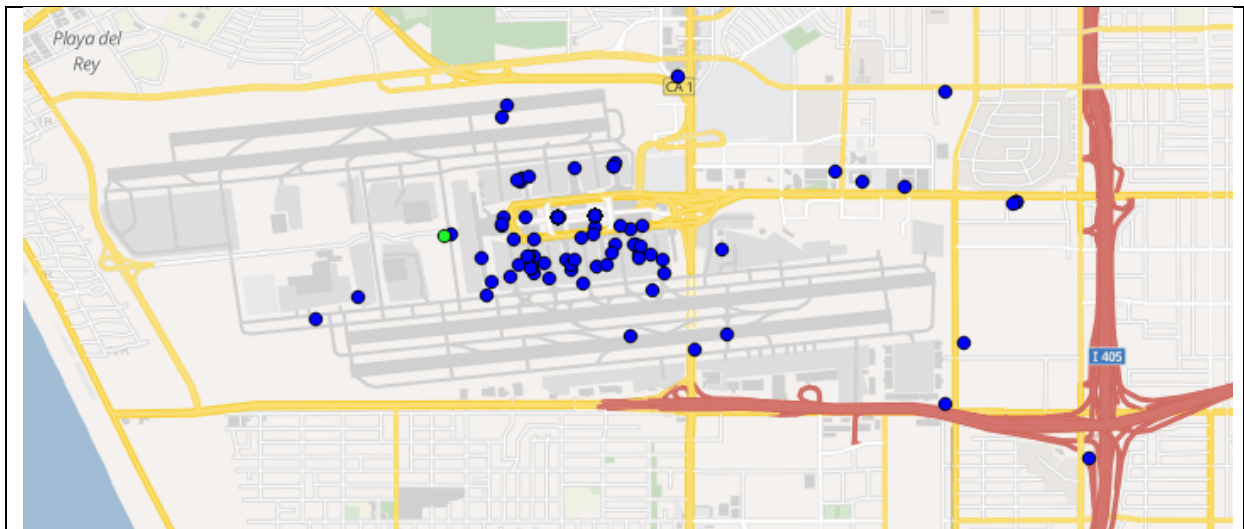


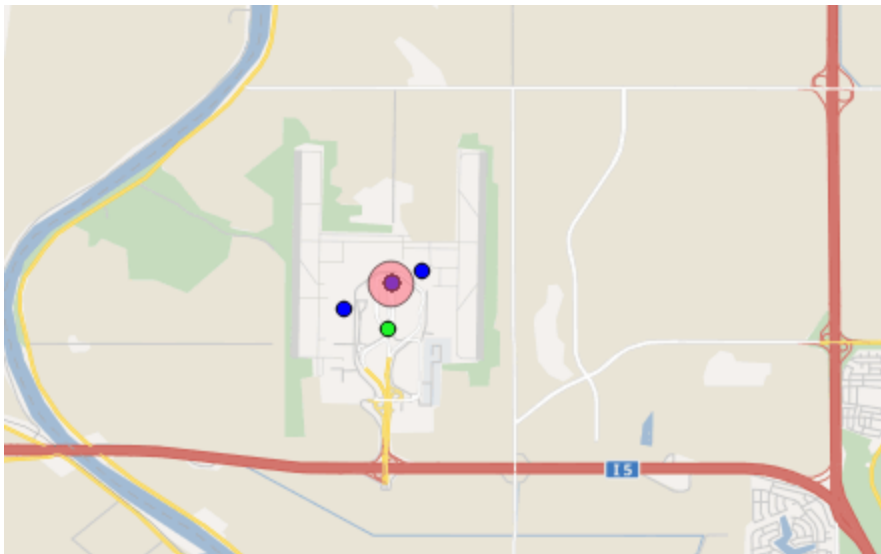
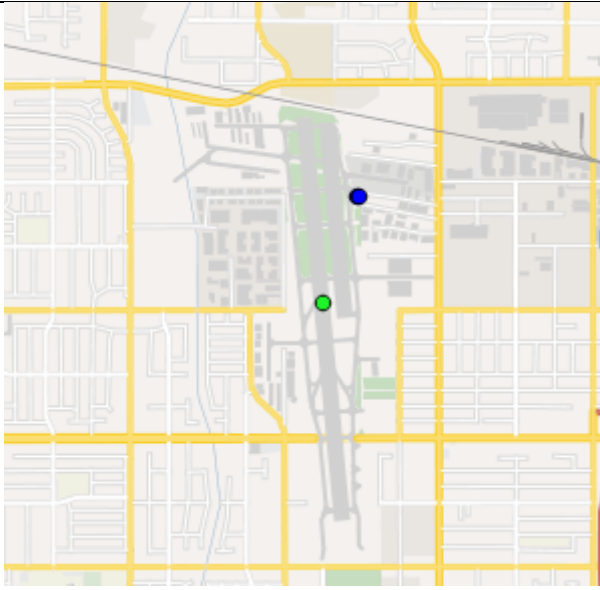


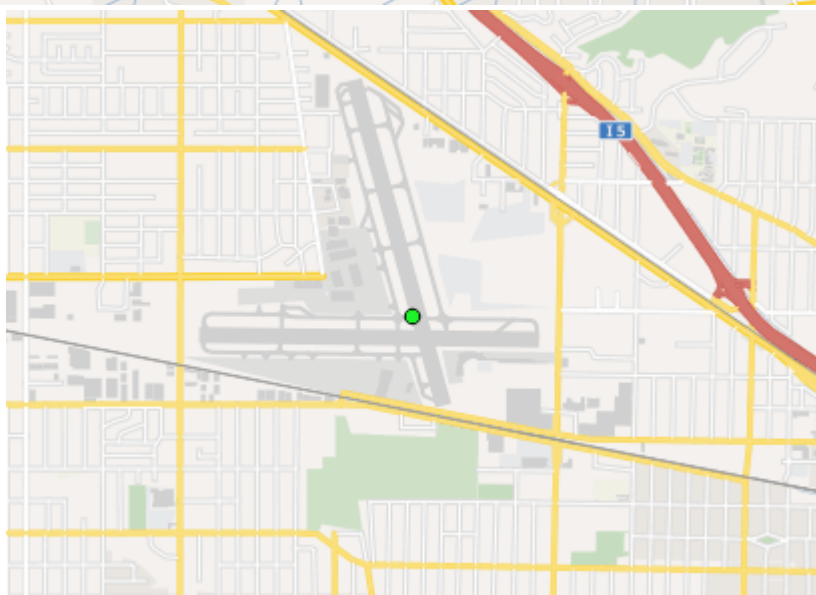
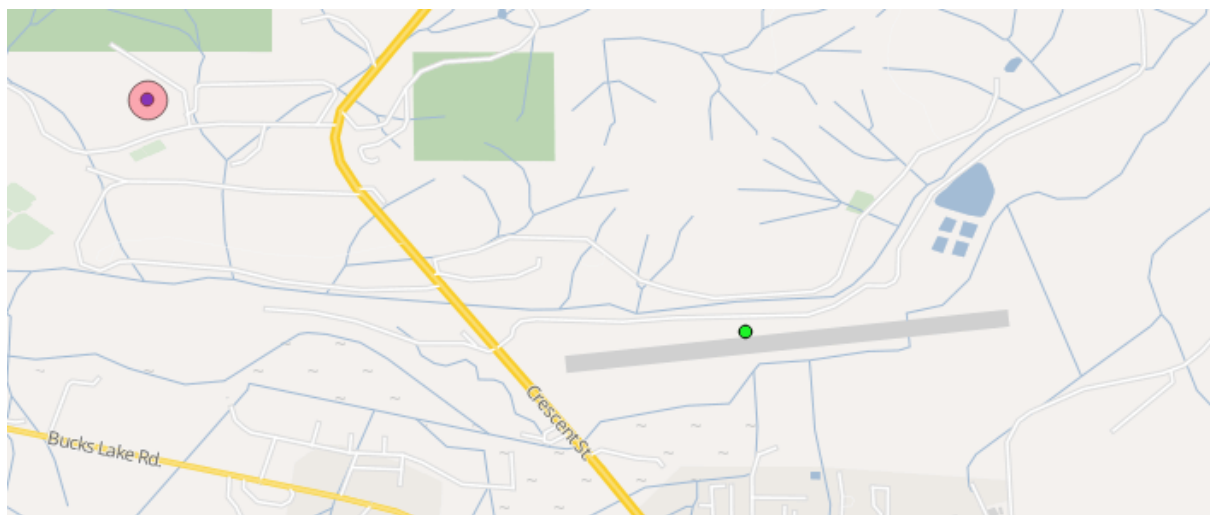


Pojedyncze wiązanie	2	inconsistent	1	0/9	0%
					
Pojedyncze wiązanie	1	inconsistent	87	-	Nie spełnia oczekiwań
					
Pełne wiązanie	5	distance	19	3/9	
					

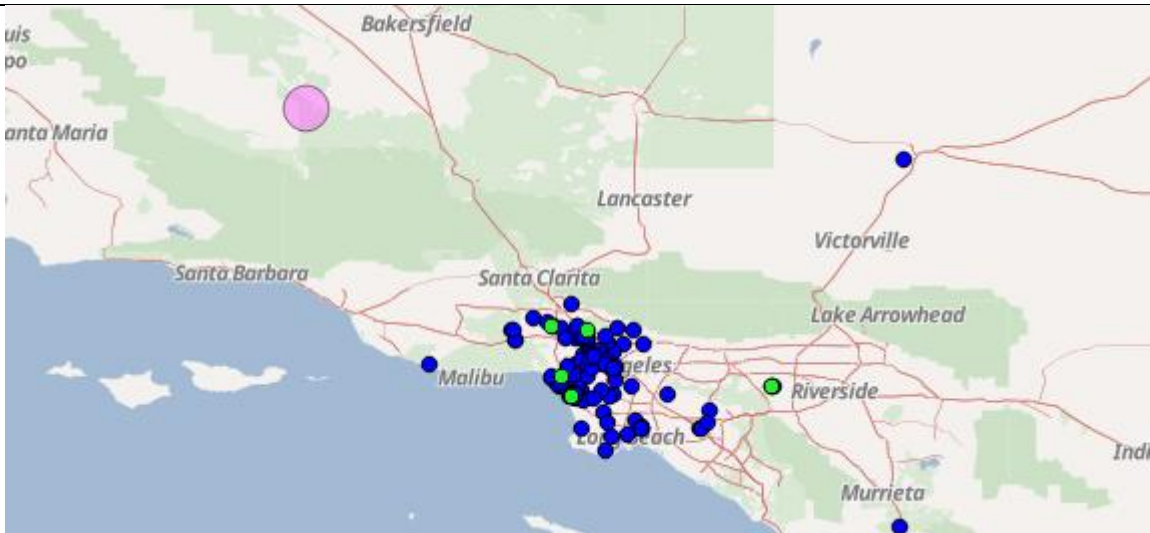








<b>Pełne wiązanie</b>	1	inconsistent	79	-	Nie spełnia oczekiwań
<b>Pełne wiązanie</b>	2	inconsistent	1	-	-



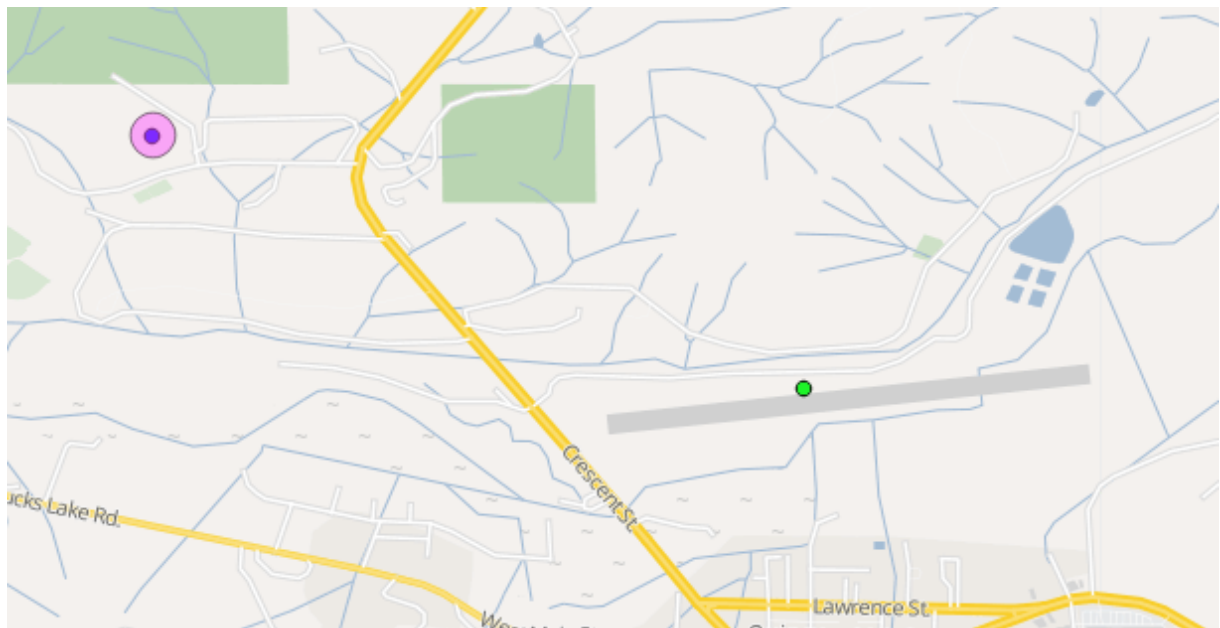
Wiązanie  
średnich

4

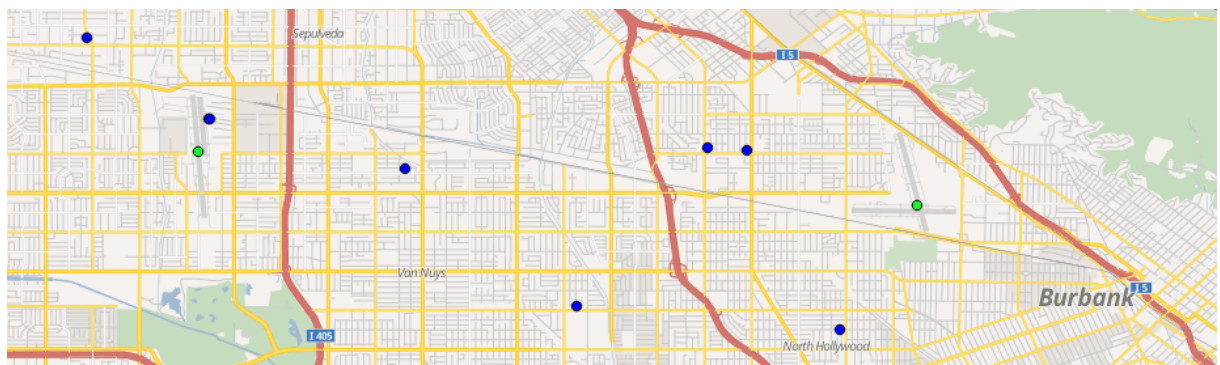
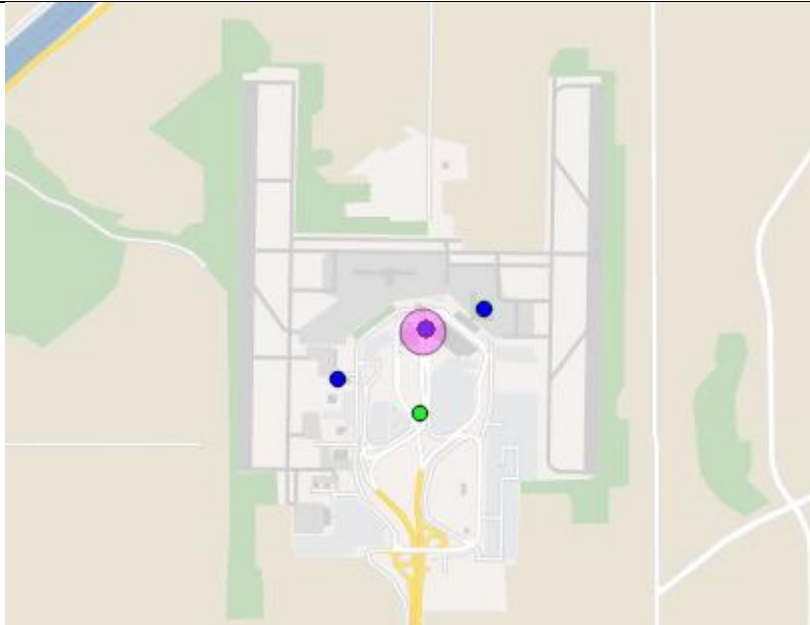
distance

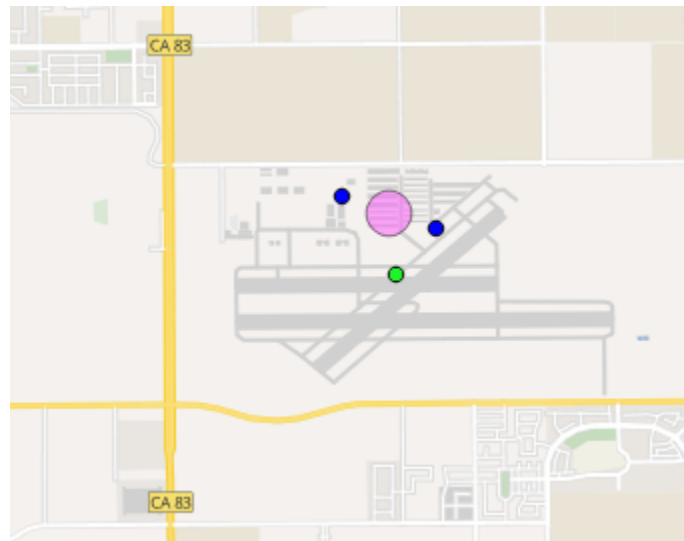
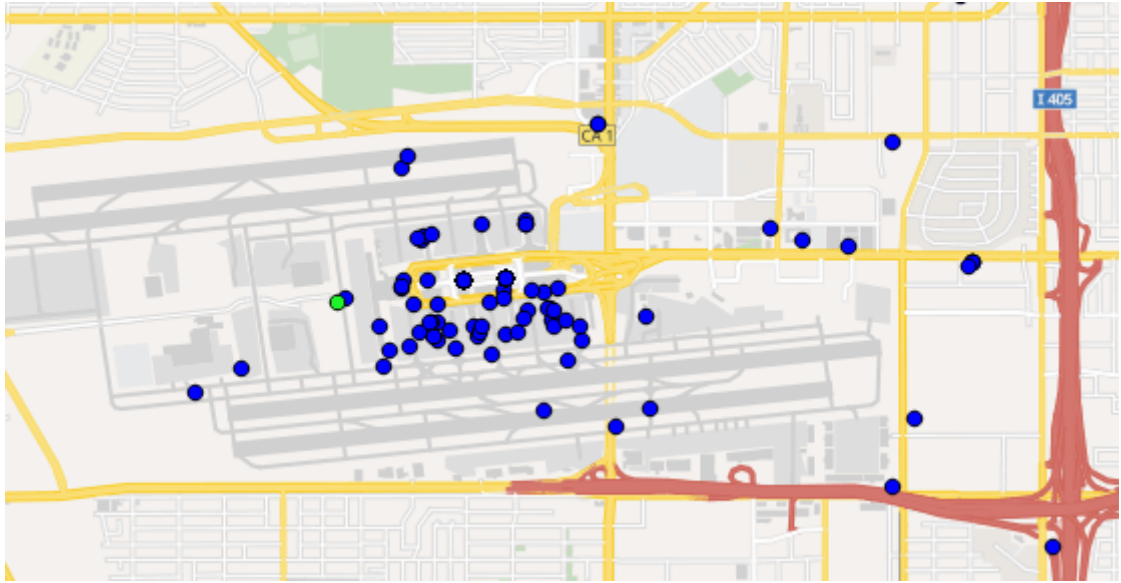
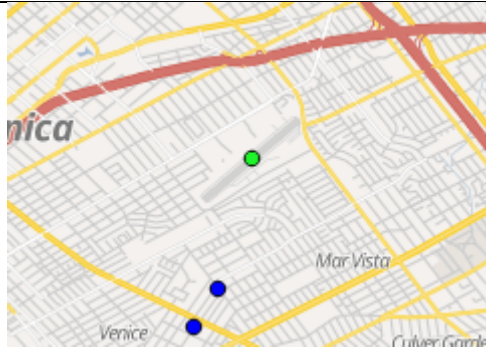
20

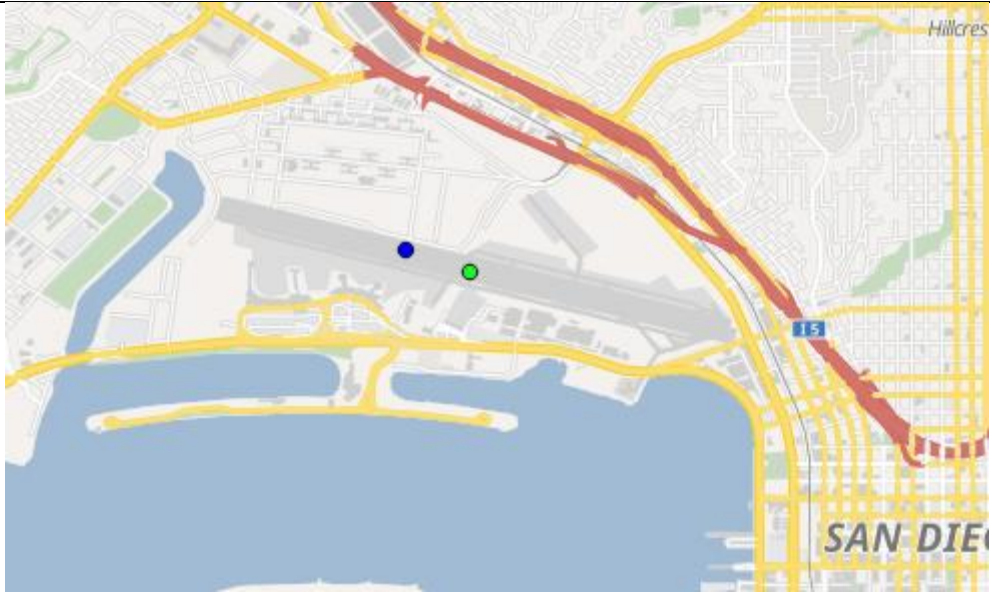
4/9



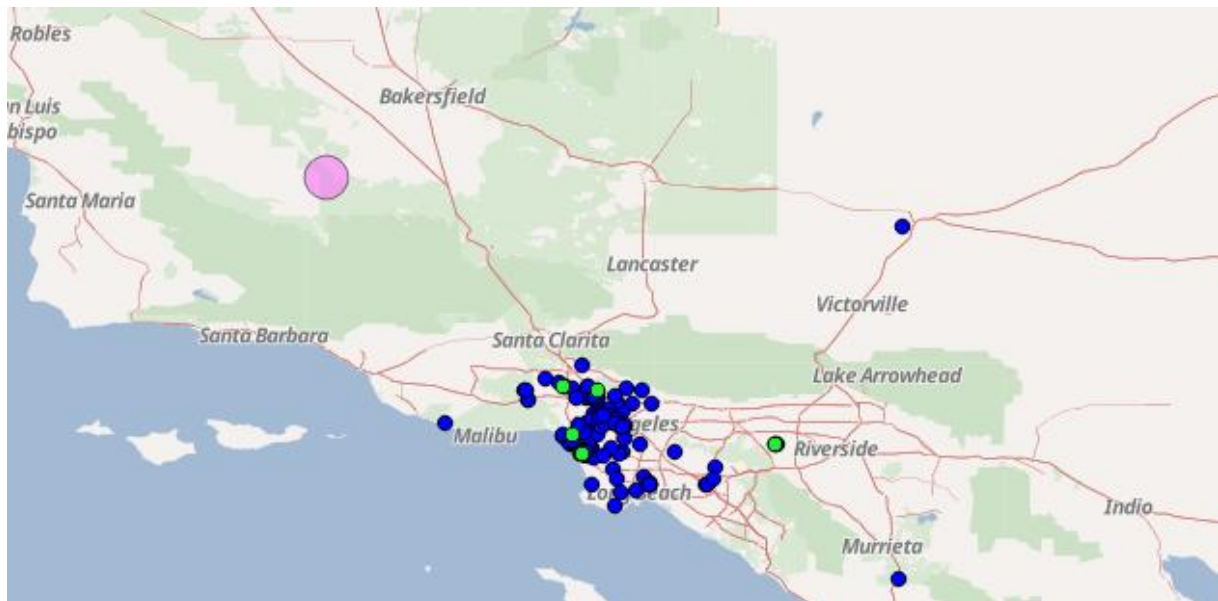


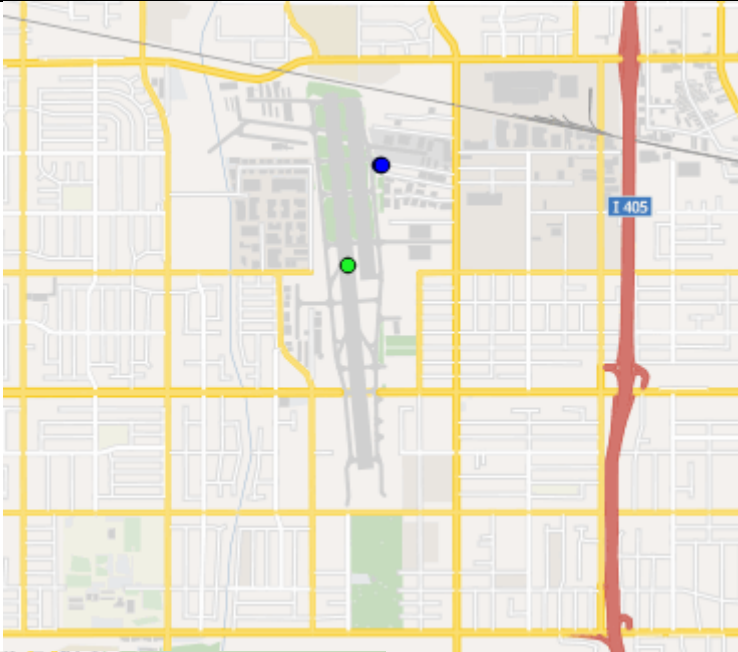
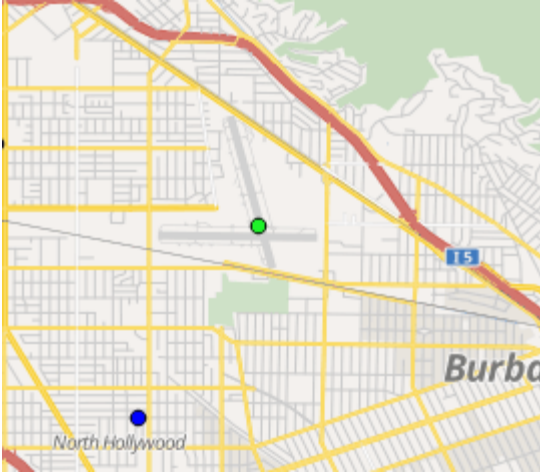
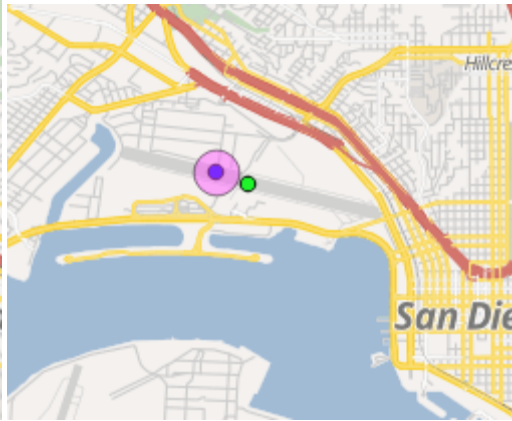
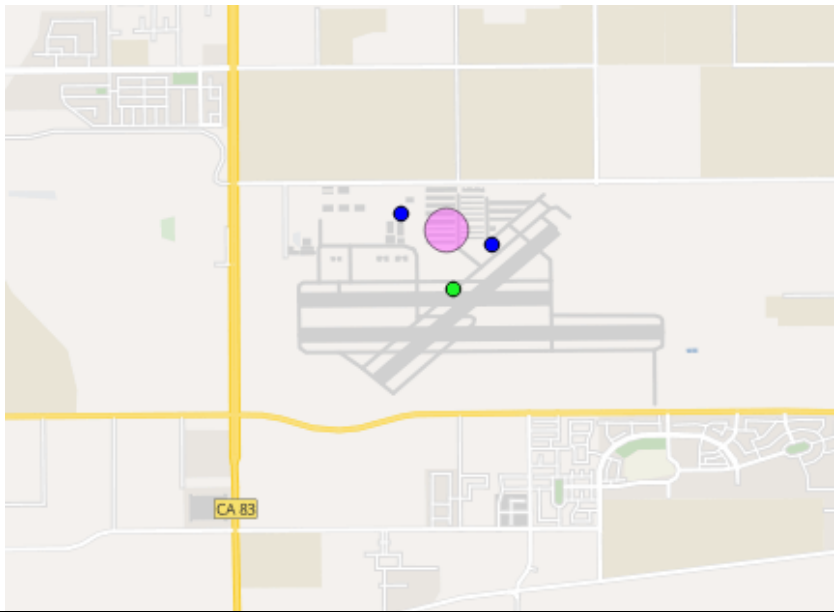




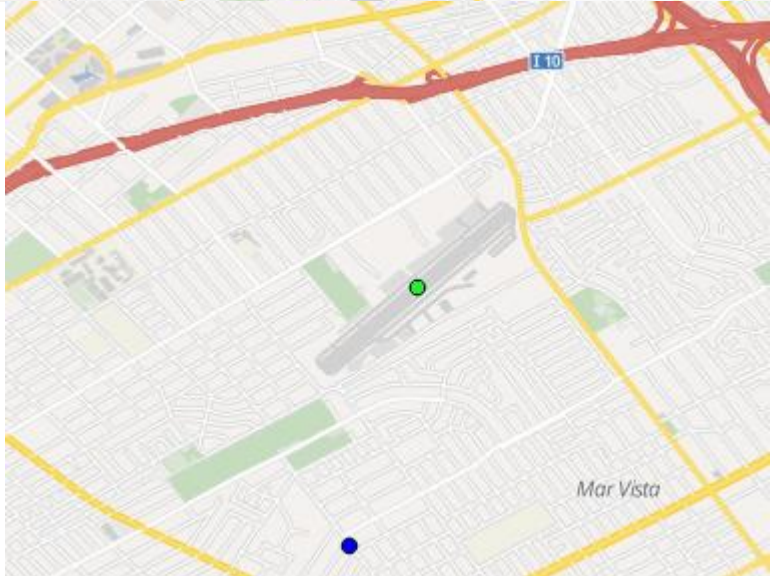
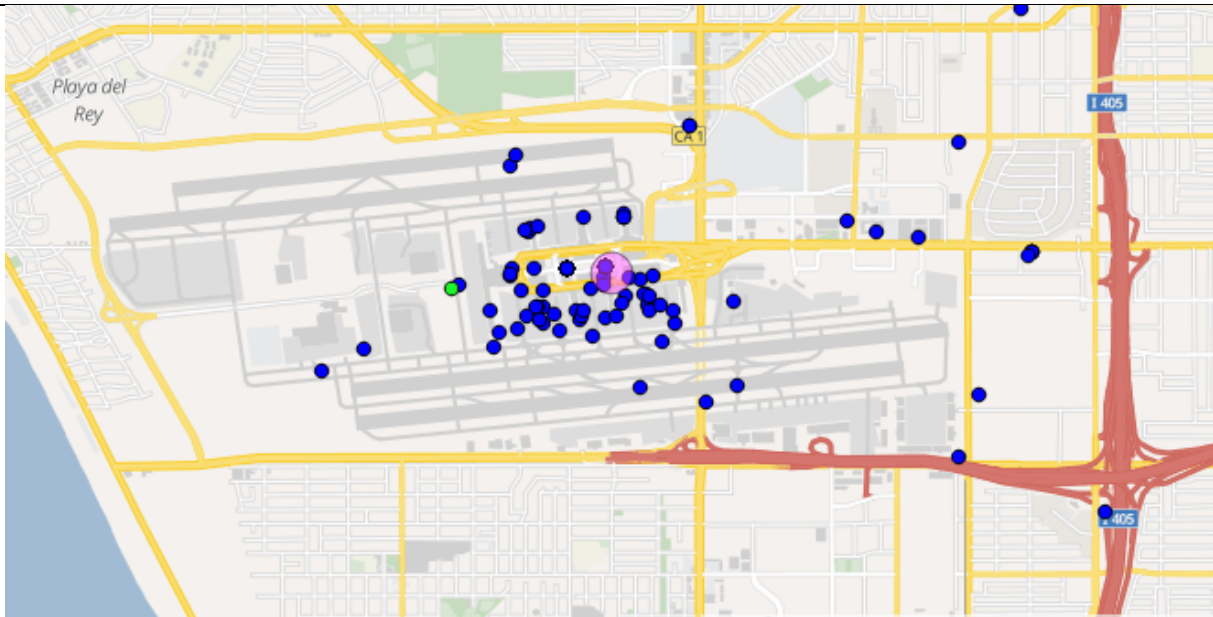


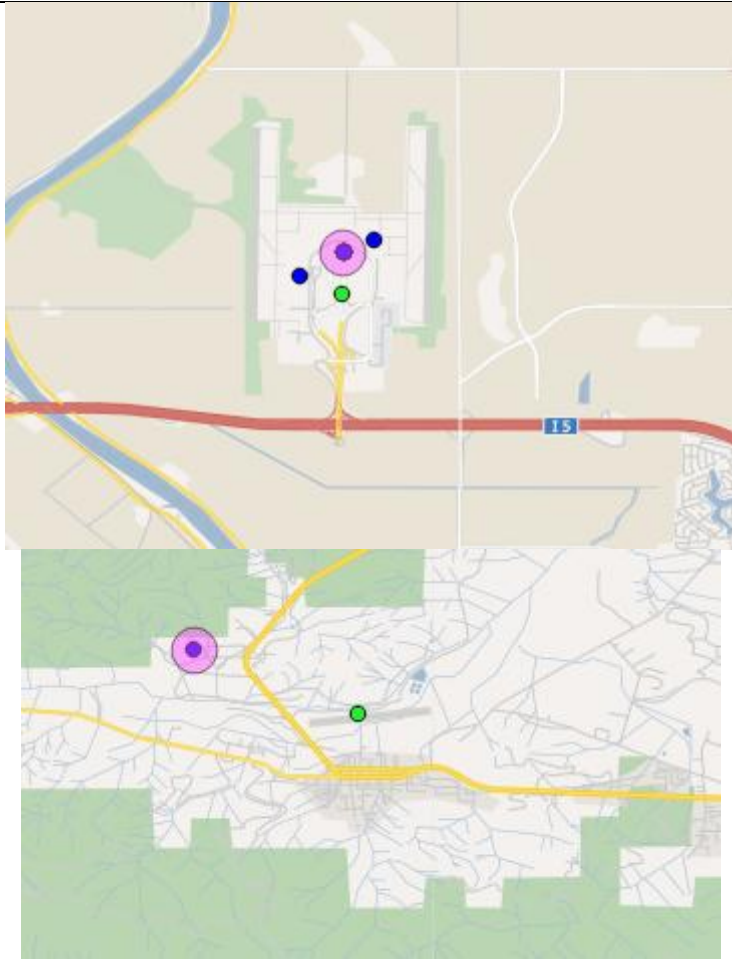
Wiązanie średnich	1	inconsistent	87	-	Nie spełnia oczekiwań
Wiązanie średnich	2	inconsistent	1	-	-



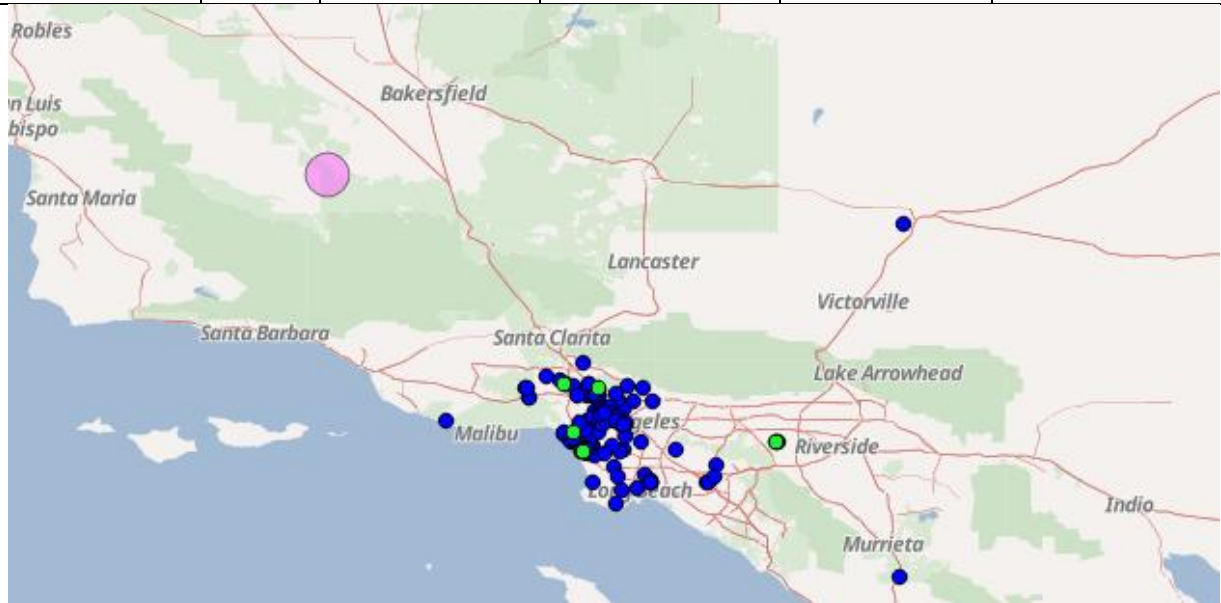
Metoda centroidów	2	distance	23	6/9	
   					





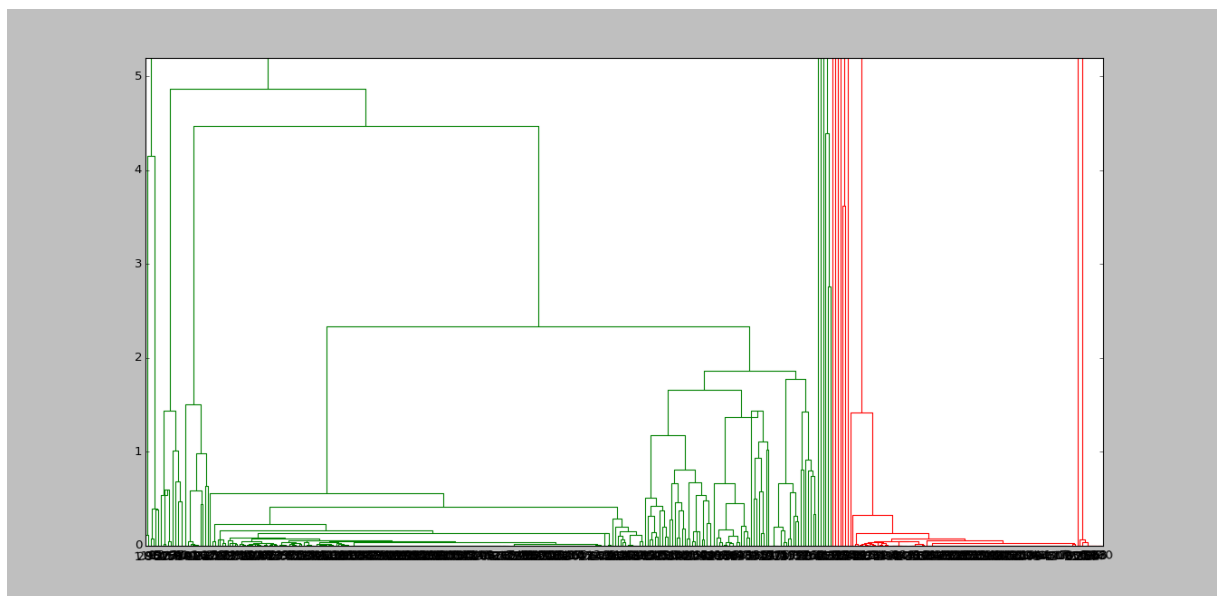
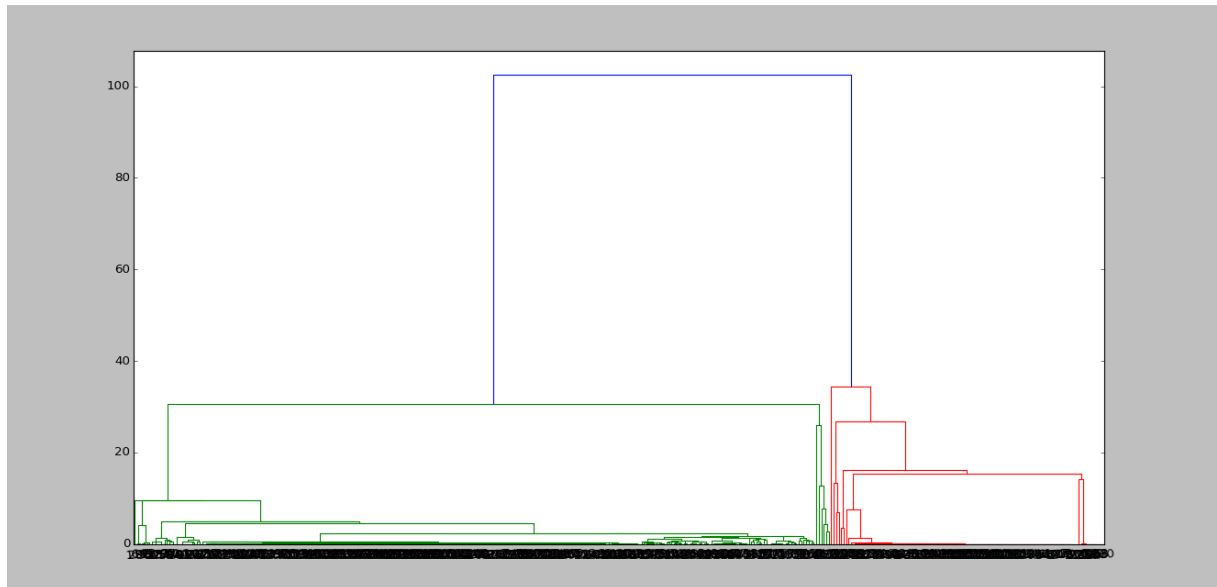


<b>Metoda centroidów</b>	1	inconsistent	84		Nie spełnia
<b>Metoda centroidów</b>	2	inconsistent	1		



## 7. Dendrogram dla najskuteczniejsze metody klasteryzacji danych:

Poniżej znajduje się dendrogram dla najskuteczniejszej metody klasteryzacji - metoda centroidów z kryterium odległościowym. Pierwszy wykres ilustruje cały zakres przeprowadzonej klasteryzacji od wszystkich klas do jednej. Drugi wykres pokazuje jedynie fragment (zbliżenie), obrazujący, który został wykonany w programie.

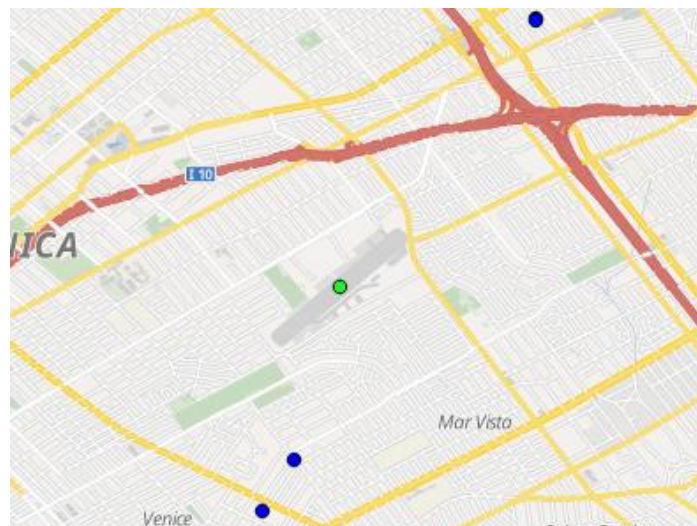


## 8. Komentarz otrzymanych wyników:

- W kolumnie SKUTECZNOŚĆ oznaczono trafność działania danej metody, wyrażoną w procentach. Wskaźnik ten oznacza stosunek ilości poprawnie wyznaczonych położeń lotnisk do wszystkich obliczonych klastrow. W niektórych przypadkach wartość ta nie została obliczona, ponieważ metoda nie dawała poprawnych i satysfakcjonujących rezultatów bez względu na testowanie różnych parametrów progów i kryteriów. Została wyznaczana albo zbyt liczna liczba klastrow lub tylko jeden obiekt;
- Wyznaczanych jest dużo klas, które nie są lotniskami;
- Kryterium Inconsistent – jest domyślne w klasteryzacji, ale generalnie gorsze niż distance;
- Widać powtarzalność w znajdowaniu/nieznajdowaniu tych samych lotnisk. Niektóre są po dobrze otagowane (posiadają wiele tweetów w obrębie lotniska, a jednocześnie jest niewiele punktów mogących zostać mylnie zaklasyfikowanych do klastra);



- Inne mają wokół dużo obserwacji „rozpraszających klasę”, nie więcej niż obserwacji skupionych na obszarze lub w bezpośrednim sąsiedztwie lotniska;



- W większości przypadków wyznaczonych pozycji jest albo:
  - o wiele więcej niż lotnisk, przez co i tak trzeba jeszcze dokonać ręcznej eliminacji. Nie ma więc możliwości całkowitej automatyzacji.
  - jedna klasa, w dodatku nie tam gdzie powinna, przy zmianie progu o jeden w stosunku do tej, gdzie wychodziło za dużo klas
- W wyniku przeprowadzonych testów, najlepszą metodą do klasteryzacji okazała się metoda centroidów z kryterium odległościowym (*distance*).

## 9. Wnioski końcowe z projektu:

Celem projektu było sprawdzenie możliwości lokalizacji obiektów na podstawie wpisów z serwisów społecznościowych, na przykładzie lokalizacji lotnisk z serwisu Twitter. Po pobraniu danych okazało się, iż niektóre wpisy znajdują się poza określonym BBoxem. Przypuszcza się, iż punkty znajdujące się w lokalizacji poza zdefiniowanym obszarem przez BBox, zostały pobrane ponieważ posiadają źle zdefiniowaną geolokalizację. Wynikać to może z faktu, iż czasami użytkownik sam definiuje swoje położenie, niekoniecznie umiejętnie. Inną przyczyną może być to, że podczas pobierania danych, zapisywane są te wpisy, których współrzędne znajdują się w określonym BBoxie lub gdy miasto znajduje się w BBoxie. Niestety zdarza się, iż tweet pobrany ze względu na poprawne miejsce nie zgadza się z jego faktycznymi współrzędnymi.

Analiza uzyskanych danych oraz przeprowadzony proces klasteryzacji pokazały, iż lokalizacja obiektów na podstawie wpisów z serwisów społecznościowych daje przeciętne rezultaty. Duża część lotnisk została poprawnie znaleziona, jednakże w zbiorze wyjściowym są również obiekty niepożądane. Wynika to z faktu występowania wpisów o zadanych słowach kluczowych, znajdujących się poza żądanym obszarem badawczym oraz wpisów których położenie nie sąsiaduje z lotniskami.

Po przeprowadzeniu testowania różnych algorytmów klasteryzacji można stwierdzić, iż najlepszą metodą do lokalizacji obiektów jest metoda centroidów z kryterium odległościowym.

Można przyjąć, iż na podstawie geolokalizacji wpisów na serwisach społecznościowych, jesteśmy w stanie wskazać lokalizację poszukiwanych obiektów. Należy jednak dobrać odpowiednie i jednoznaczne słowa kluczowe, dostatecznie duży obszar testowy oraz długi czas kolekcjonowania danych, jak również odpowiednią metodę i technikę klasteryzacji. Niestety pośród wyselekcjonowanych obiektów zawsze znajdą się obiekty niepożądane. Pełna automatyzacja tego procesu nie jest możliwa..