ΤΕΧΝΙΚΕΣ ΕΞΟΡΥΞΗΣ ΔΕΔΟΜΕΝΩΝ

ΑΣΚΗΣΗ 2

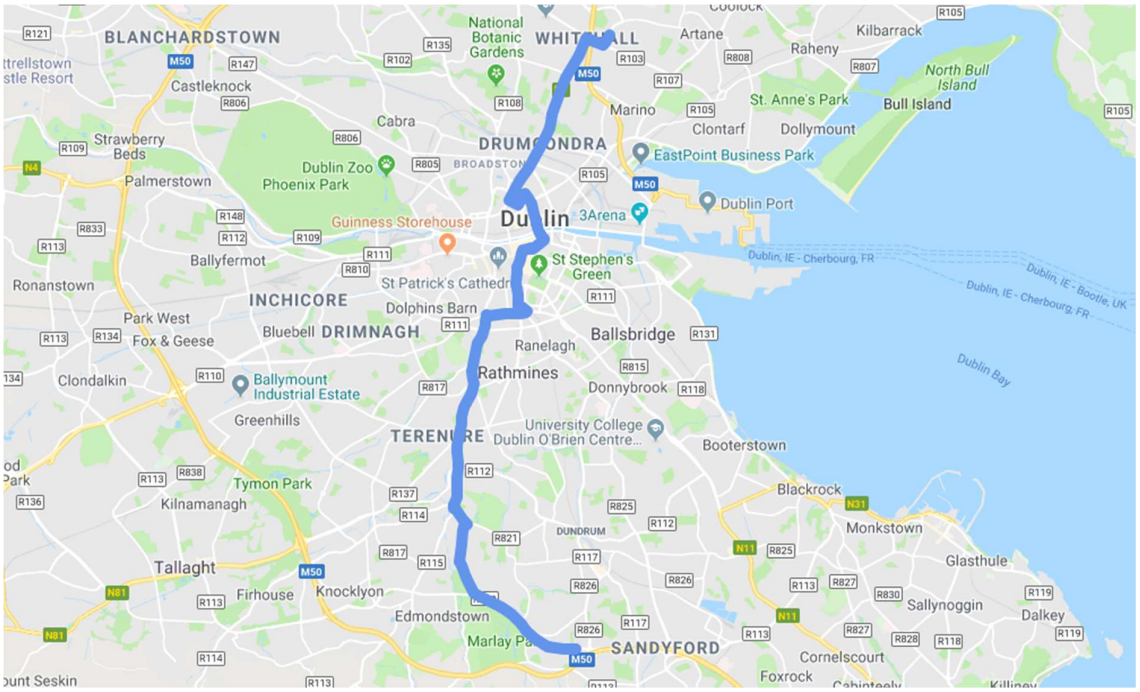
ΣΤΑΜΑΤΟΠΟΥΛΟΣ ΒΑΣΙΛΕΙΟΣ
(1115201400188)

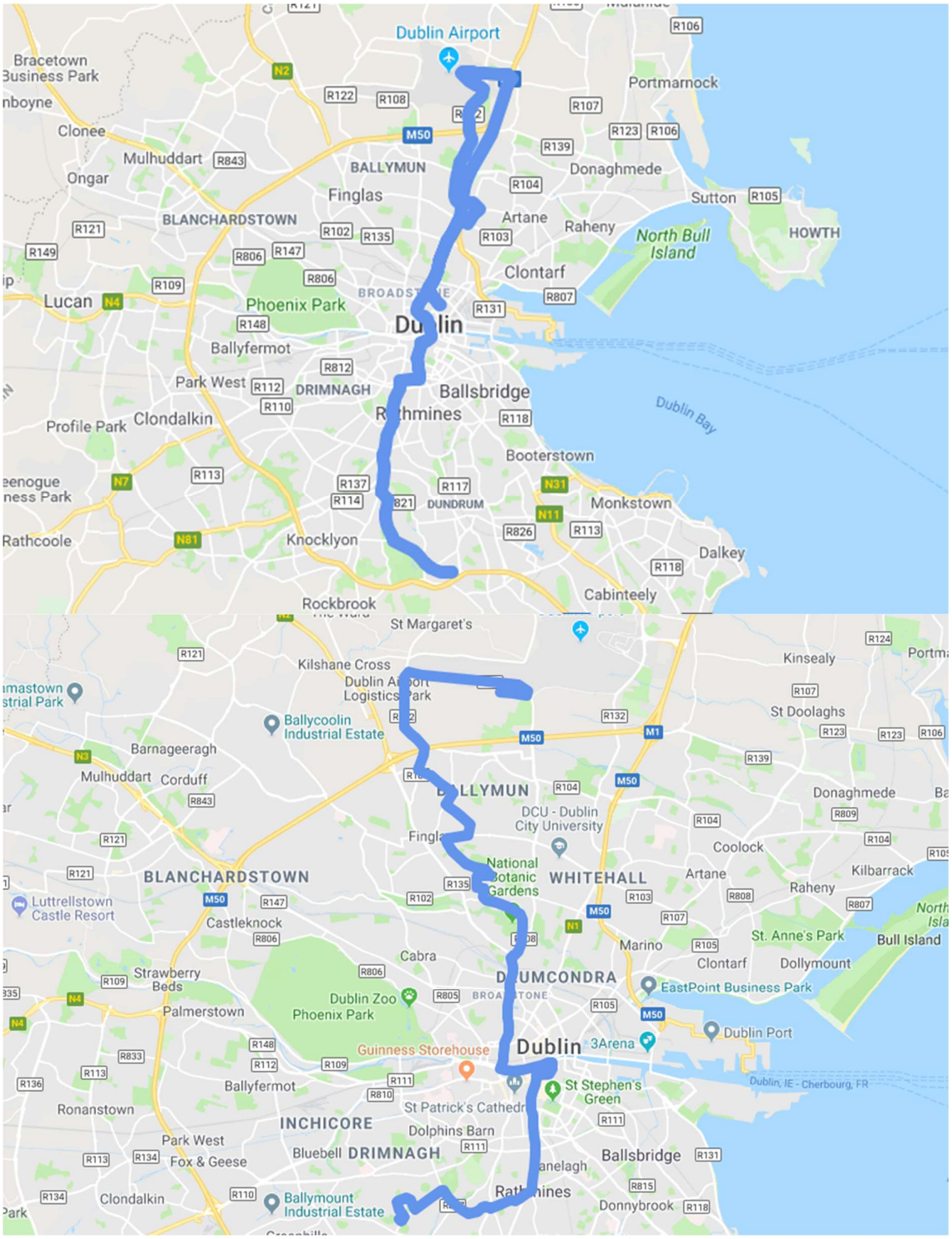
–

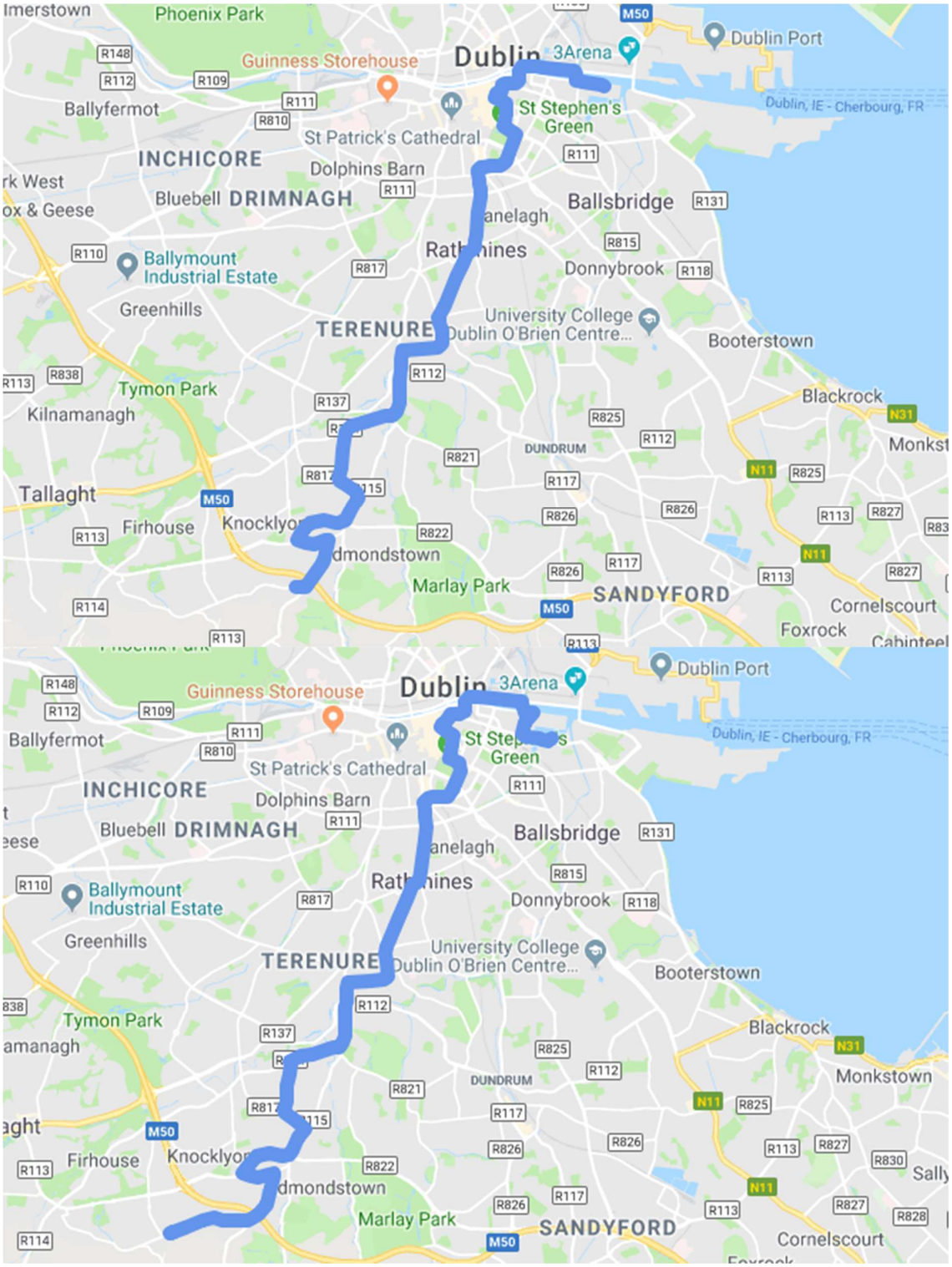
ΚΟΥΚΑΚΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ
(1115201400289)

Ερώτημα 1

Το ερώτημα αυτό απαντήθηκε στο αρχείο mapping.py με τα αποτελέσματα του να βρίσκονται στο φάκελο maps. Πήραμε 5 τυχαία ids από το dataset και βρήκαμε την πρώτη εμφάνιση του καθενός στο σύνολο και χαρτογραφήσαμε την καθεμιά από αυτές. Έχουμε 5 δυάδες λιστών lats και lons, και αναλόγως τις γεμίζουμε με τα δεδομένα των συντεταγμένων.
Τέλος, χαρτογραφούμε τις διαδρομές με τη χρήση του **gmplot** και παίρνουμε τα παρακάτω αποτελέσματα.




Dublin Airport
Bracetown Business Park
Clonee
Mulhuddart
Ongar
BALLYMUN
Finglas
BLANCHARDSTOWN
Portmarnock
Donaghmede
Sutton
Artane
Raheny
North Bull Island
HOWTH
Clontarf
Lucan
Phoenix Park
Dublin
Ballyfermot
Park West
DRIMNAGH
Ballsbridge
Rathmines
Profile Park
Clondalkin
Dublin Bay
Booterstown
Monkstown
DUNDRUM
Rathcoole
Knocklyon
Dalkey
Cabinteely
Rockbrook
St Margaret's
Kilshane Cross
Dublin Airport Logistics Park
Ballycoolin Industrial Estate
Barnageeragh
Mulhuddart
Corduff
Kinsealy
St Doolaghs
DCU - Dublin City University
Coolock
Kilbarrack
National Botanic Gardens
WHITEHALL
Luttrellstown Castle Resort
Castleknock
Cabra
Marino
St. Anne's Park
Bull Island
Clontarf
Dollymount
Strawberry Beds
DRUMCONDRA
EastPoint Business Park
Dublin Zoo
Phoenix Park
Palmerstown
Guinness Storehouse
Dublin
3Arena
Dublin Port
St Stephen's Green
Dublin, IE - Cherbourg, FR
Ballyfermot
Ronanstown
St Patrick's Cathedral
INCHICORE
Dolphins Barn
Park West
Bluebell
DRIMNAGH
Fox & Geese
Ballsbridge
Clondalkin
Ballymount Industrial Estate
Donnybrook


Phoenix Park
Guinness Storehouse
Dublin
3Arena
Dublin Port
St Stephen's Green
Dublin, IE - Cherbourg, FR
St Patrick's Cathedral
INCHICORE
Dolphins Barn
Bluebell
DRIMNAGH
Ballsbridge
Ballymount Industrial Estate
Donnybrook
Greenhills
TERENURE
University College Dublin O'Brien Centre...
Booterstown
Tymon Park
Kilnamanagh
Blackrock
DUNDRUM
Tallaght
Firhouse
Marlay Park
SANDYFORD
Cornelscourt
Foxrock
Monkstown
M50
N11
N31
R110
R111
R112
R113
R114
R117
R118
R131
R137
R148
R810
R815
R817
R821
R822
R825
R826
R827
R838

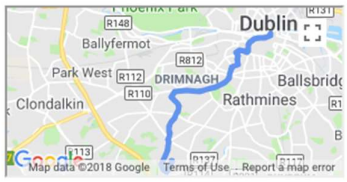
Ερώτημα 2

Α.

Για το ερώτημα αυτό ο κώδικας βρίσκεται στο αρχείο dtw_nn.py. Εκεί, υπάρχει μια συνάρτηση η οποία υπολογίζει την απόοσταση haversine μεταξύ δύο σημείων συντεταγμένων. Για κάθε trajectory στο σύνολο ελέγχου, παίρνουμε τις συντεταγμένες και τις αποθηκεύουμε σε δύο λίστες. Έπειτα, δημιουργούμε ένα .html αρχείο το οποίο αναπαριστά τη διαδρομή και θα το χρησιμοποιήσουμε αργότερα στα αποτελέσματα. Υπολογίζουμε την dtw της διαδρομής με κάθε διαδρομή στο σύνολο εκπαίδευσης, και για αυτή τη διαδικασία χρησιμοποιούμε τον αλγόριθμο της βιβλιοθήκης dtw.

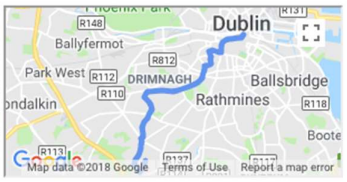
Στη συνέχεια, τοποθετούμε όλες τις αποστάσεις σε μια λίστα από tuples {διάσταση, index}, ταξινομούμε την λίστα βάση της απόστασης και βρίσκουμε τις 5 μικρότερες. Τέλος, αφού χαρτογραφήσουμε τις γειτονικές αποστάσεις, με τη βοήθεια του iframe, φτιάχνουμε έναν πίνακα html για να δώσουμε τα αποτελέσματα, τα οποία δίνονται στη συνέχεια.

Χρόνος Εκτέλεσης: 2428.12000012



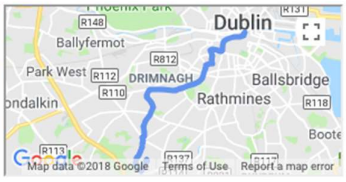
Test Trip 1
Dt= 1223.76438594sec

Neighbor 1
JP_ID: 01501001
DTW: 0.0

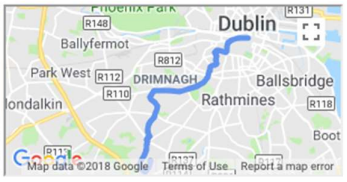


Neighbor 2
JP_ID: 01501001
DTW: 0.013874606415368754

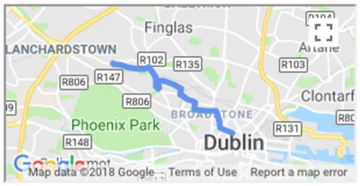
Neighbor 3
JP_ID: 01501001
DTW: 0.01522404812947076



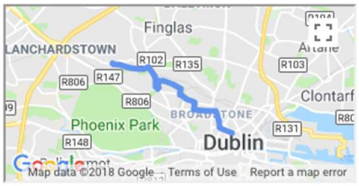
Neighbor 4
JP_ID: 01501001
DTW: 0.015736379836091052



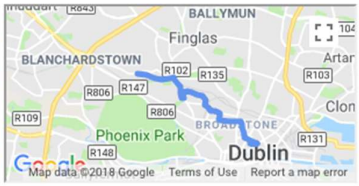
Neighbor 5
JP_ID: 01501001
DTW: 0.016013091332028724



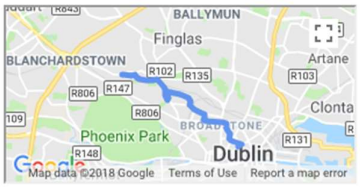
Test Trip 2
Dt= 623.550071955sec



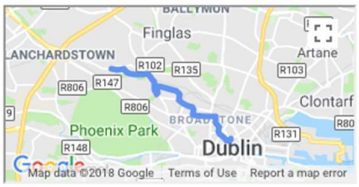
Neighbor 1
JP_ID: 01200001
DTW: 0.0



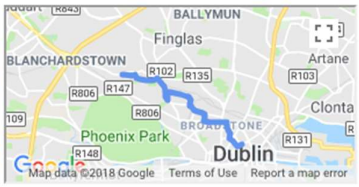
Neighbor 2
JP_ID: 01200001
DTW: 0.020685469224294074



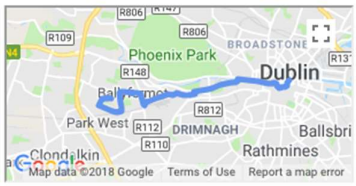
Neighbor 3
JP_ID: 01200001
DTW: 0.022780762378723198



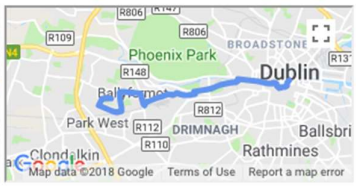
Neighbor 4
JP_ID: 01200001
DTW: 0.02432501949003578



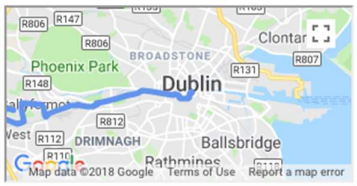
Neighbor 5
JP_ID: 01200001
DTW: 0.02438856487725342



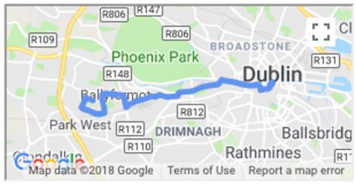
Test Trip 3
Dt= 1173.08323908sec



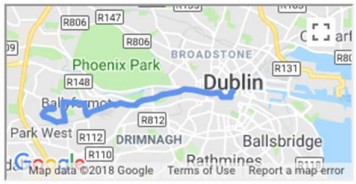
Neighbor 1
JP_ID: 00791001
DTW: 0.0



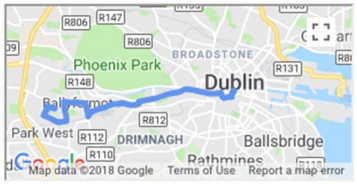
Neighbor 2
JP_ID: 00791001
DTW: 0.015536994044971993



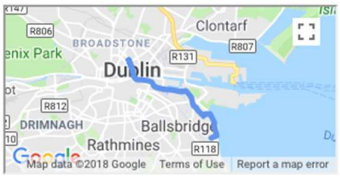
Neighbor 3
JP_ID: 00791001
DTW: 0.016580769101374924



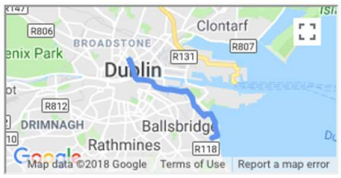
Neighbor 4
JP_ID: 00791001
DTW: 0.01710115456587192



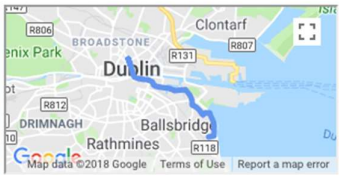
Neighbor 5
JP_ID: 00791001
DTW: 0.017658909723781403



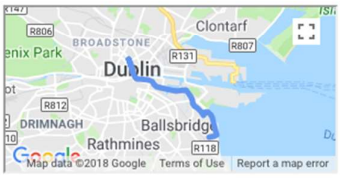
Test Trip 4
Dt= 755.005935907sec



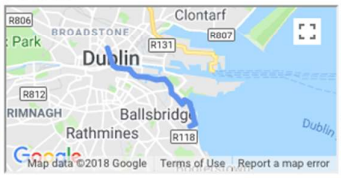
Neighbor 1
JP_ID: 00010002
DTW: 0.0



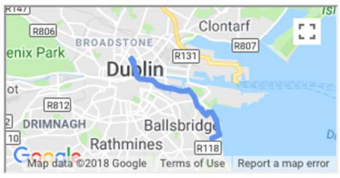
Neighbor 2
JP_ID: 00010002
DTW: 0.014882175877381278



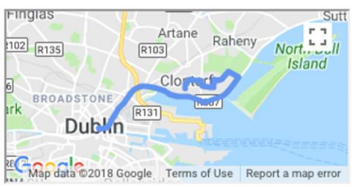
Neighbor 3
JP_ID: 00010002
DTW: 0.017247889261746016



Neighbor 4
JP_ID: 00010002
DTW: 0.019696952751058337

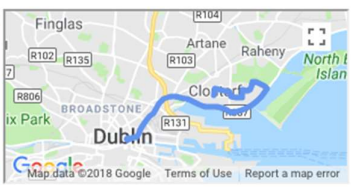


Neighbor 5
JP_ID: 00010002
DTW: 0.019894708089114727

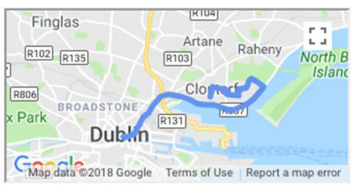


Test Trip 5
Dt= 760.965547085sec

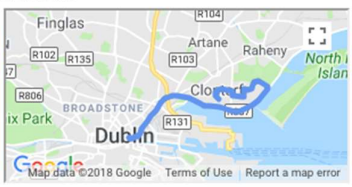
Neighbor 1
JP_ID: 01300001
DTW: 0.0



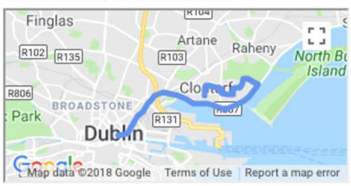
Neighbor 2
JP_ID: 01300001
DTW: 0.024293949698498574



Neighbor 3
JP_ID: 01300001
DTW: 0.024818117863589624



Neighbor 4
JP_ID: 01300001
DTW: 0.02561600605820375



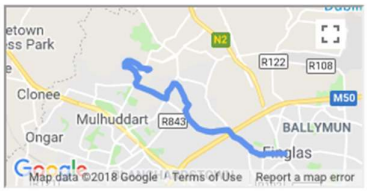
Neighbor 5
JP_ID: 01300001
DTW: 0.025947464263656815

B.

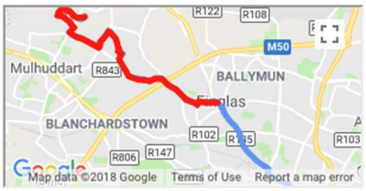
Για το ερώτημα αυτό ο κώδικας βρίσκεται στο αρχείο lcss.py. Η διαδικασία μοιάζει αρκετά με αυτή του προηγούμενου ερωτήματος μόνο που εδώ αντί για μικρότερες αποστάσεις, ψάχνουμε τα περισσότερα κοινά σημεία. Τα σημεία αυτά τα βρίσκουμε με τον αλγόριθμο lcs (longest common subsequence), ο οποίος επιστρέφει δύο πράγματα. Τον αριθμό των ταιριαστών σημείων και το μεγαλύτερο κοινό μονοπάτι. Δύο σημεία ταιριάζουν όταν απέχουν λιγότερο από 200m. Για το κοινό μονοπάτι χρησιμοποιούμε την συνάρτηση common_path οι οποία πέρνει των πίνακα που δημιουργεί η lcs και το test path και ελέγχει τα ταιριαστά διαγώνια σημεία και αναλόγως μειώνει το αντίστοιχο index (I,j). Στην ουσία κάνει traverse των πίνακα όπως θα τον έκανε και η lcs. Τέλος, αφού έχουμε το κοινό μονοπάτι και τα κοινά σημεία τα τοποθετούμε σε μια λίστα και δημιουργούμε ένα tuple του τύπου: matching = [[#no of points, [common_path]], index].

Σε αυτή τη δομή βρίσκονται όλα τα στοιχεία που χρειαζόμαστε για να καταγράψουμε τα αποτελέσματα μας.

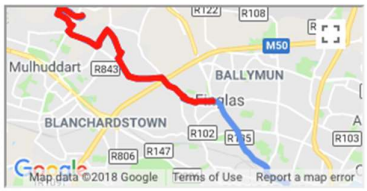
Τα αποτελέσματα δίνονται παρακάτω:



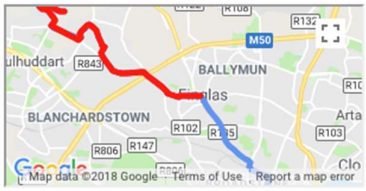
Test Trip 1
Dt= 711.824347019sec



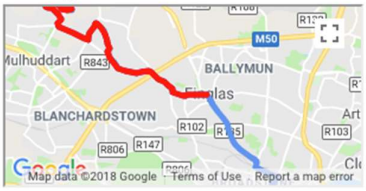
Neighbor 1
JP_ID: 040D1002
Matching Points: 78



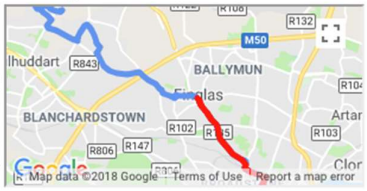
Neighbor 2
JP_ID: 040D1002
Matching Points: 78



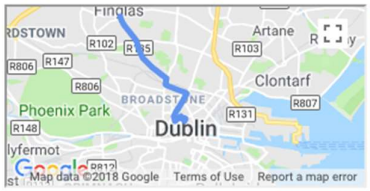
Neighbor 3
JP_ID: 040D1002
Matching Points: 76



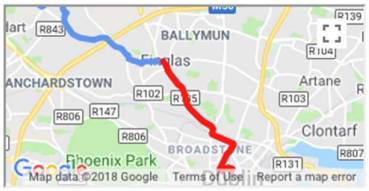
Neighbor 4
JP_ID: 040D1002
Matching Points: 76



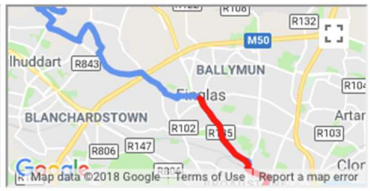
Neighbor 5
JP_ID: 040D1002
Matching Points: 75



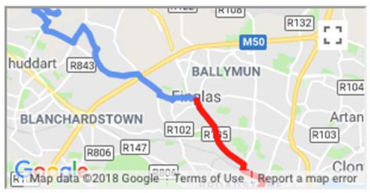
Test Trip 2
Dt= 747.889531851sec



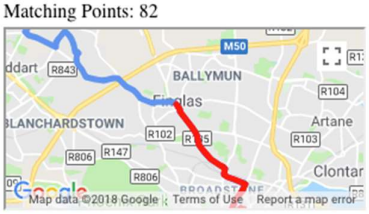
Neighbor 1
JP_ID: 040D1002
Matching Points: 82



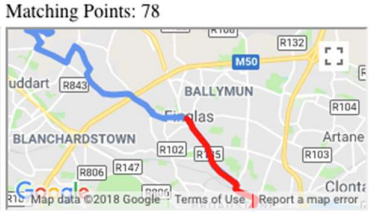
Neighbor 2
JP_ID: 040D1002
Matching Points: 78



Neighbor 3
JP_ID: 040D1002
Matching Points: 75

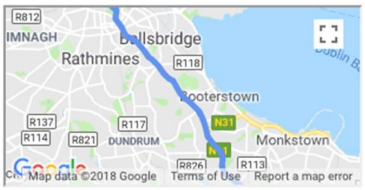


Neighbor 4
JP_ID: 040D1002
Matching Points: 74

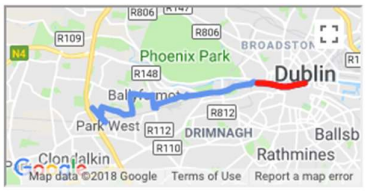


Neighbor 5
JP_ID: 040D1002
Matching Points: 73

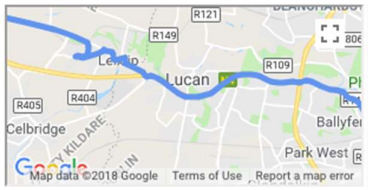
Test Trip 3
Dt= 362.457656145sec



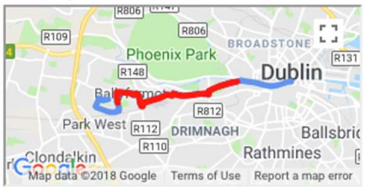
Neighbor 1
JP_ID: 01451001
Matching Points: 40



Neighbor 2
JP_ID: 079A0001
Matching Points: 40

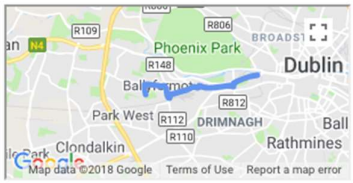


Neighbor 3
JP_ID: 066X0004
Matching Points: 40

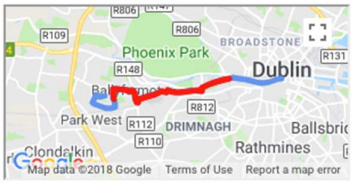


Neighbor 4
JP_ID: 00790001
Matching Points: 40

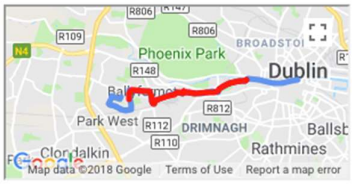
Neighbor 5
JP_ID: 01451008
Matching Points: 40



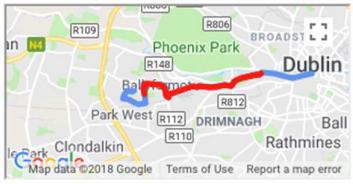
Test Trip 4
Dt= 516.552026033sec



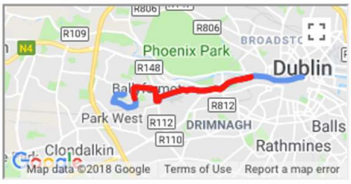
Neighbor 1
JP_ID: 00790001
Matching Points: 59



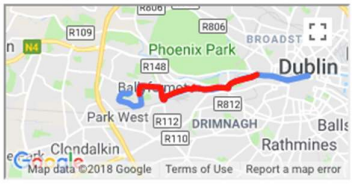
Neighbor 2
JP_ID: 00790001
Matching Points: 59



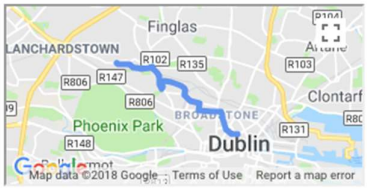
Neighbor 3
JP_ID: 00790001
Matching Points: 59



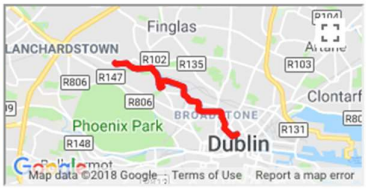
Neighbor 4
JP_ID: 00790001
Matching Points: 59



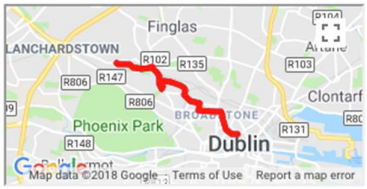
Neighbor 5
JP_ID: 00790001
Matching Points: 59



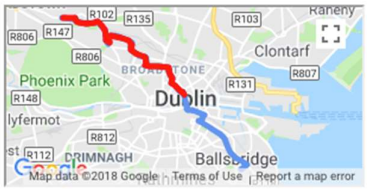
Test Trip 5
Dt= 640.398183107sec



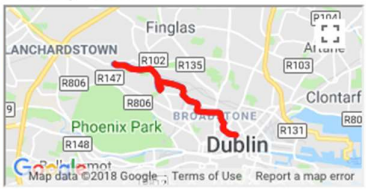
Neighbor 1
JP_ID: 01200001
Matching Points: 73



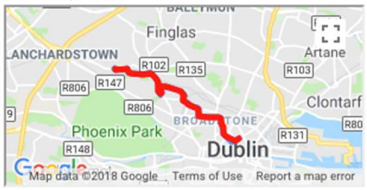
Neighbor 2
JP_ID: 01200001
Matching Points: 73



Neighbor 3
JP_ID: 01200002
Matching Points: 72



Neighbor 4
JP_ID: 01200001
Matching Points: 71



Neighbor 5
JP_ID: 01200001
Matching Points: 71

ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ: 1344.80699992 sec

Ερώτημα 3

Στο ερώτημα αυτό χρησιμοποιήσαμε τον αλγόριθμο που φτιάξαμε στην προηγούμενη εργασία για να ταξινομήσουμε τις διαδρομές του test_set.

Τα αποτελέσματα της απλής ταξινόμησης ήταν τα εξής:

1	040D1002
2	01201001
3	00371001
4	00790001
5	01200001

Για την διασταυρούμενη επικύρωση χρησιμοποιήσαμε τον ίδιο αλγόριθμο με το StratifiedKFold(), αλλά η διαδικασία δεν τερμάτιζε ποτέ. Σπάσαμε το σύνολο και χρησιμοποιήσαμε μονάχα το 10% του συνόλου για το οποίο πήραμε accuracy = 0.90920