

1^η Εργασία μαθήματος Τεχνικές Εξόρυξης Δεδομένων

Ιούνιος 2018

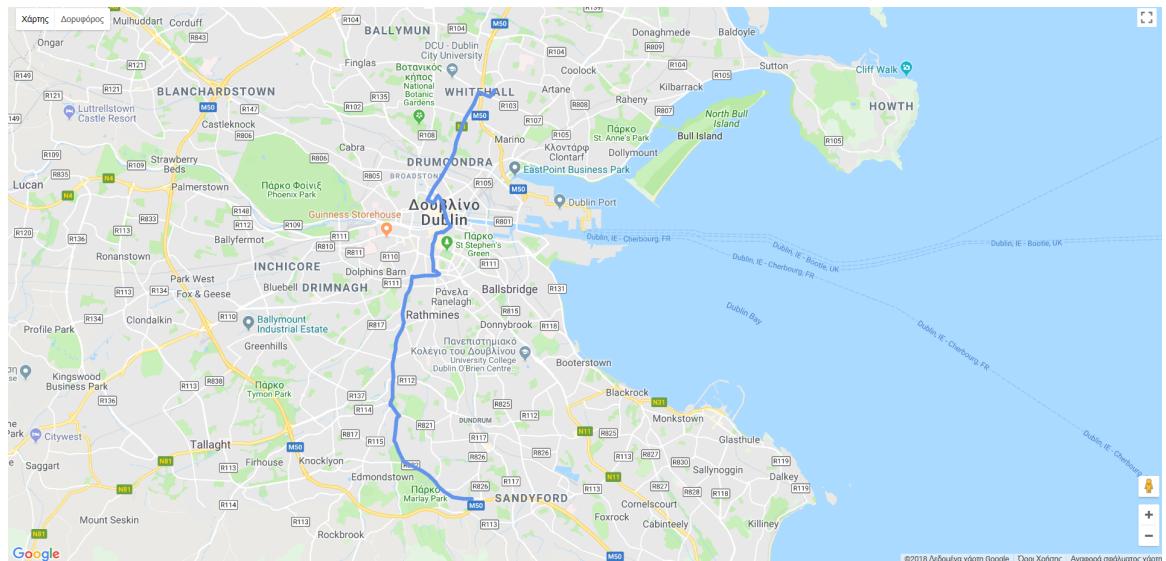
Ομάδα

- Ζέρντεβ Αλέξανδρος ΑΜ: 1115201600283
- Ανδρινούλου Χριστίνα ΑΜ: 1115201500006

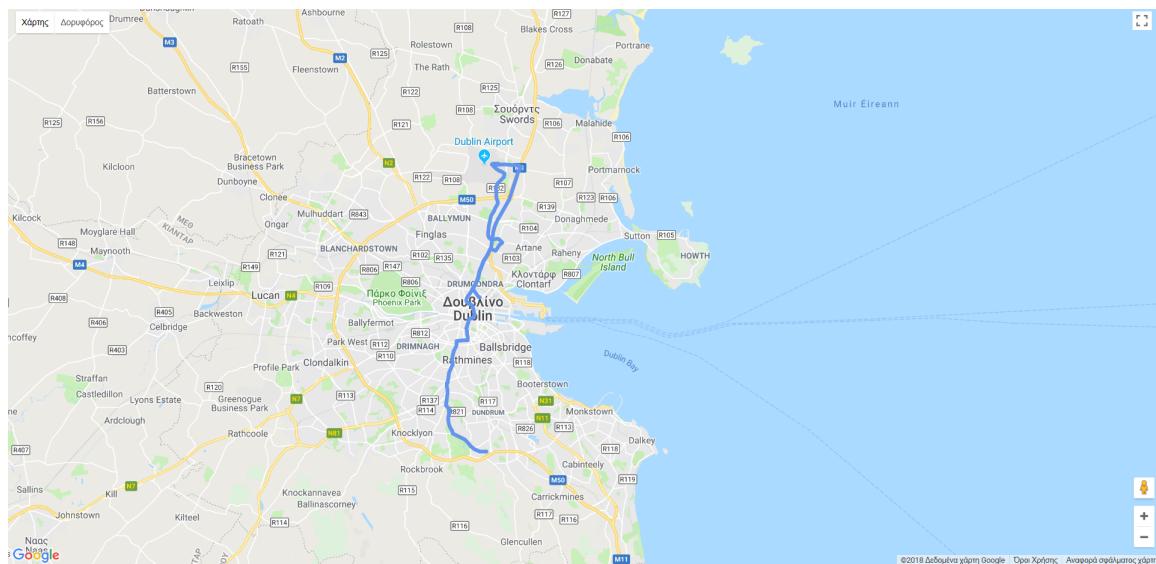
1. Οπτικοποίηση δεδομένων

Με χρήση της βιβλιοθήκης της *python* οπτικοποιήσαμε 5 διαφορετικές γραμμές λεωφορείων. Η επιλογή των γραμμών έγινε τυχαία. Έπειτα από πειραματισμούς, χρησιμοποιήσαμε ως γεωγραφικές παραμέτρους στο *gmpplot.GoogleMapPlotter* την πρώτη τιμή για το γεωγραφικό πλάτος και μήκος από το κάθε *Trajectory*. Παραθέτουμε τους 5 χάρτες που προέκυψαν.

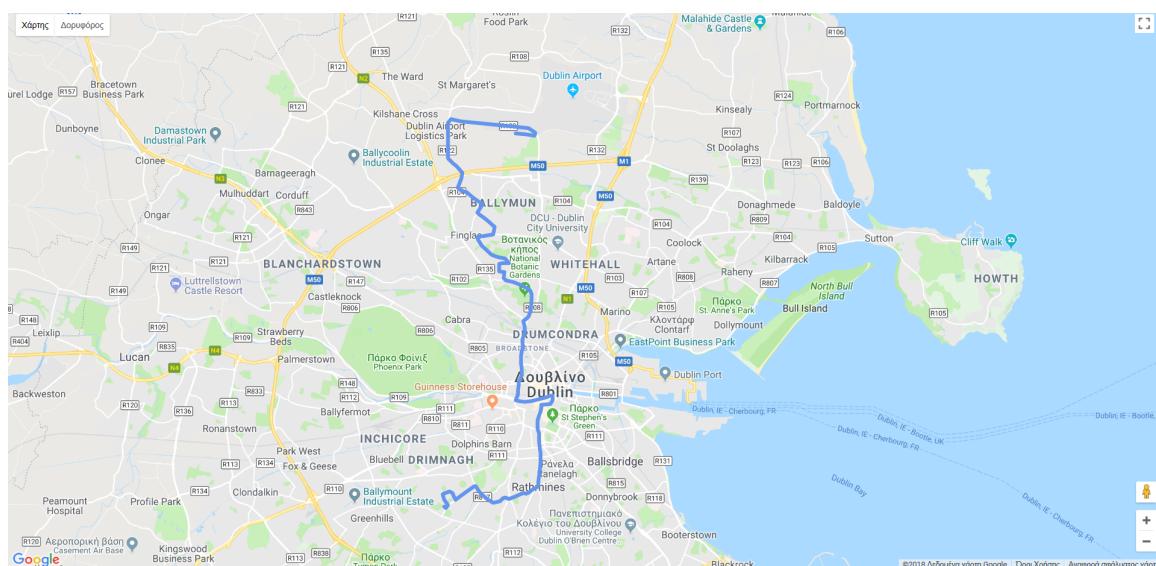
Map 1



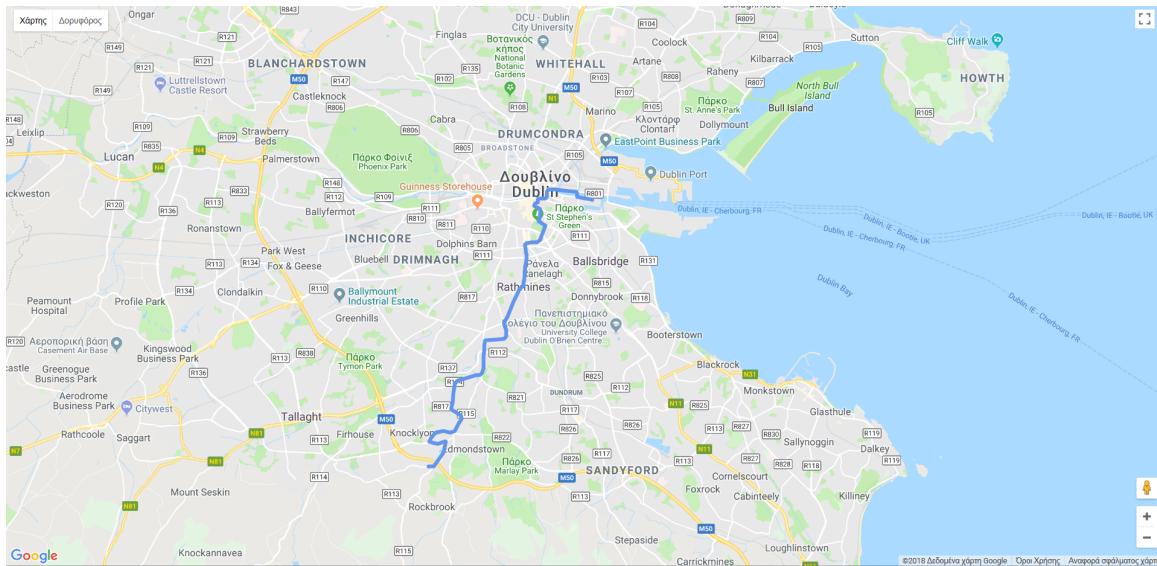
Map 2



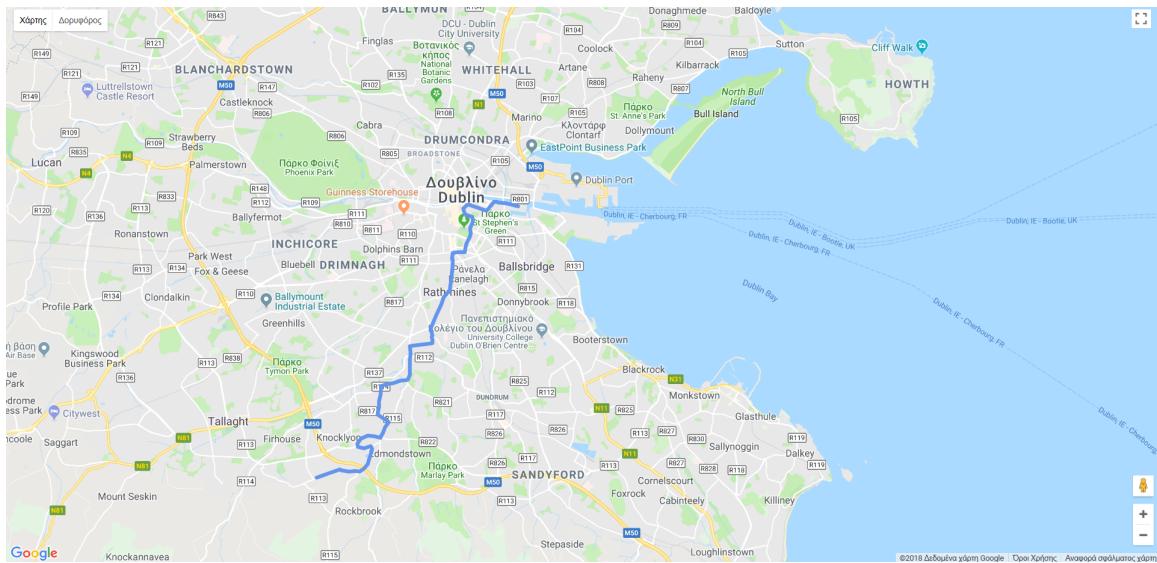
Map 3



Map 4



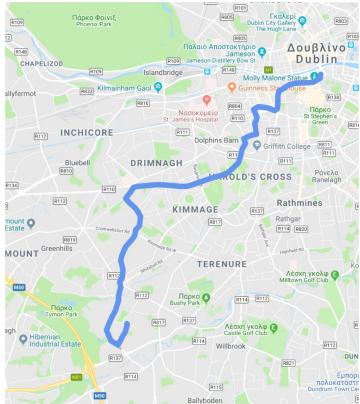
Map 5



2. Εύρεση κοντινότερων γειτόνων και υποδιαδρομών

Ερώτημα α1: Εύρεση κοντινότερων γειτόνων

Στο ερώτημα αυτό εντοπίσαμε τις πέντε κοντινότερες διαδρομές για κάθε μία διαδρομή του αρχείου *test set a1*. Για το σκοπό αυτό χρησιμοποιήθηκαν έτοιμα εργαλεία (*fastdtw* και *haversine* απόσταση). Η επιλογή αυτή έγινε για να επιτευχθούν καλύτεροι χρόνοι. Παραθέτουμε τα αποτελέσματά μας:



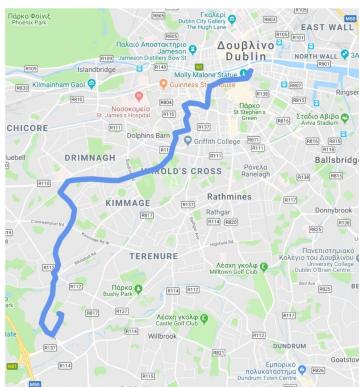
Test Trip 1
 $\Delta t = 141.3659999937$



Neighbor 1
JP_ID 01501001
DTW 0.0



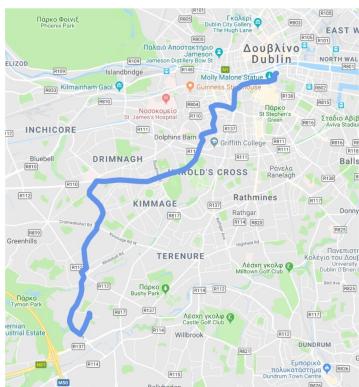
Neighbor 2
JP_ID 01501001
DTW 4.75969430159



Neighbor 3
JP_ID 01501001
DTW 5.14039600844



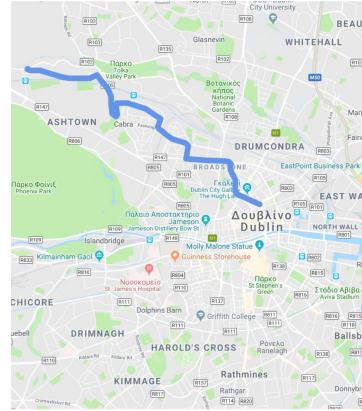
Neighbor 4
JP_ID 01501001
DTW 5.28965426169



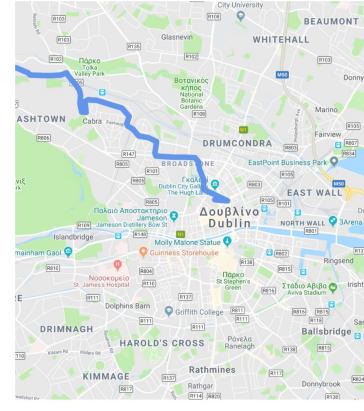
Neighbor 5
JP_ID 01501001
DTW 5.42564005929



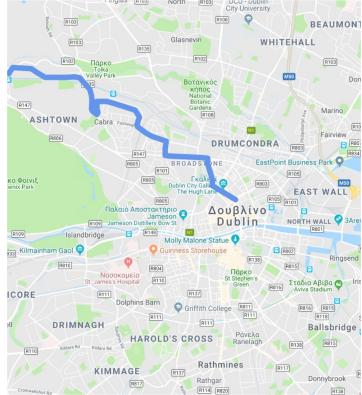
Test Trip 2
 $\Delta t = 113.030999899$



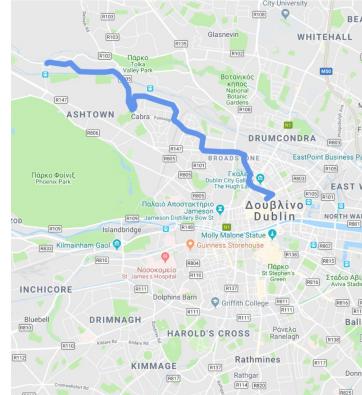
Neighbor 1
JP_ID 01200001
DTW 0.0



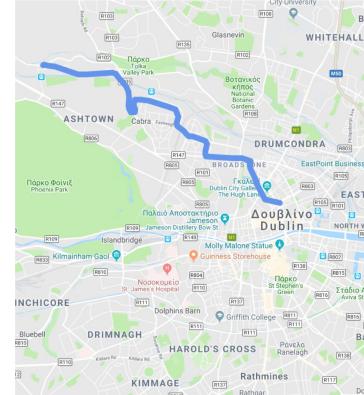
Neighbor 2
JP_ID 01200001
DTW 4.04876501568



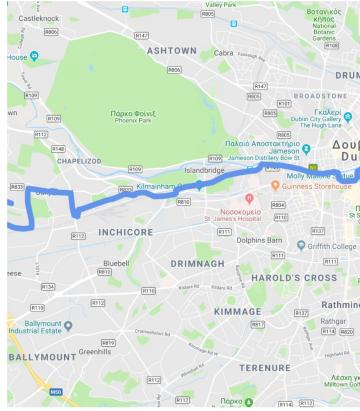
Neighbor 3
JP_ID 01200001
DTW 4.61949680884



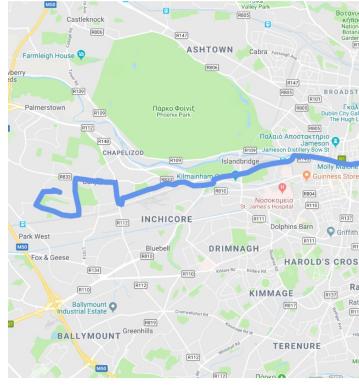
Neighbor 4
JP_ID 01200001
DTW 4.75077338023



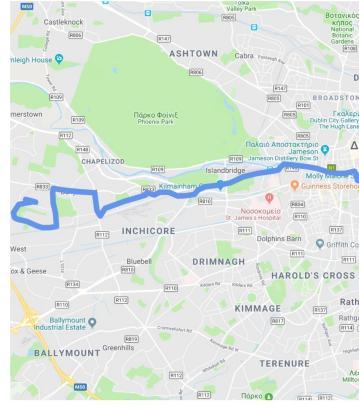
Neighbor 5
JP_ID 01200001
DTW 4.937631730



Test Trip 3
 $\Delta t = 143.839999914$



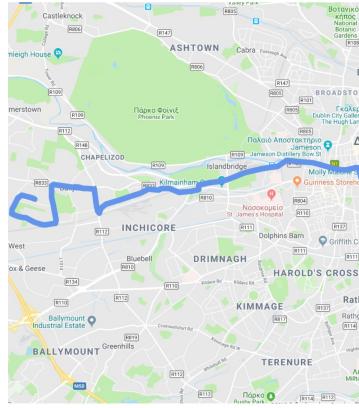
Neighbor 1
JP_ID 00791001
DTW 0.0



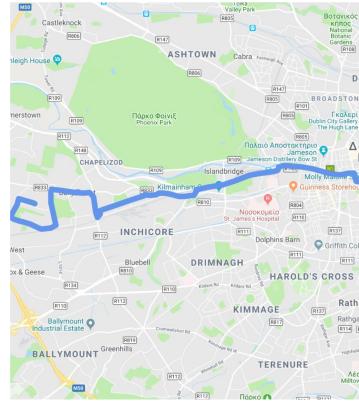
Neighbor 2
JP_ID 00791001
DTW 6.9054618011



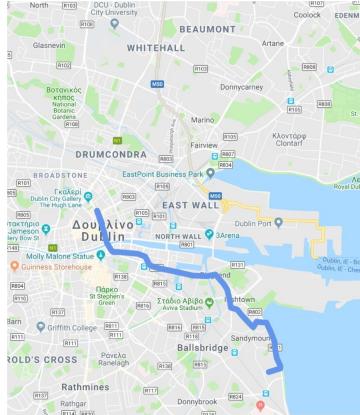
Neighbor 3
JP_ID 00791001
DTW 7.06420700122



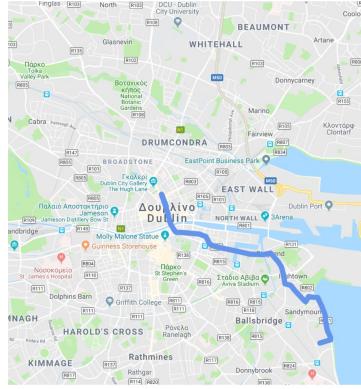
Neighbor 4
JP_ID 00791001
DTW 7.13563586752



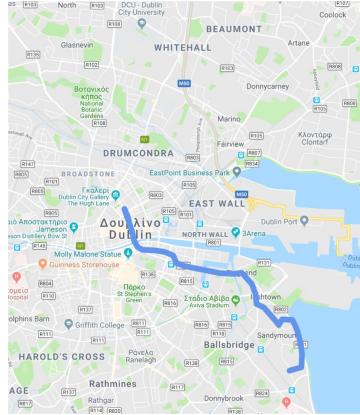
Neighbor 5
JP_ID 00791001
DTW 7.1457382969



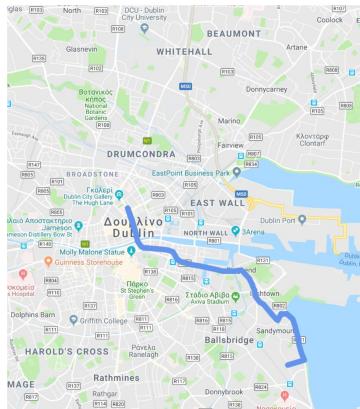
Test Trip 4
Δt = 121.111000061



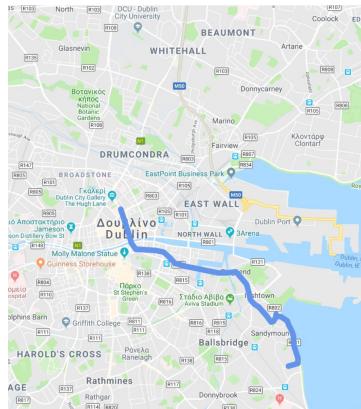
Neighbor 1
JP_ID 00010002
DTW 0.0



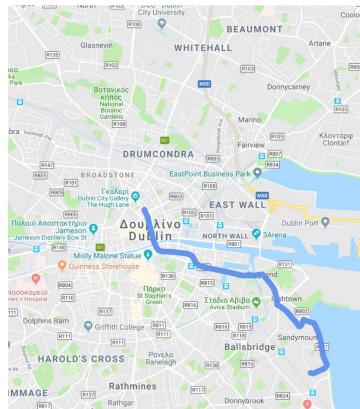
Neighbor 2
JP_ID 00010002
DTW 3.54415661464



Neighbor 3
JP_ID 00010002
DTW 3.90347495017



Neighbor 4
JP_ID 00010002
DTW 4.28835007873



Neighbor 5
JP_ID 00010002
DTW 4.83905670127



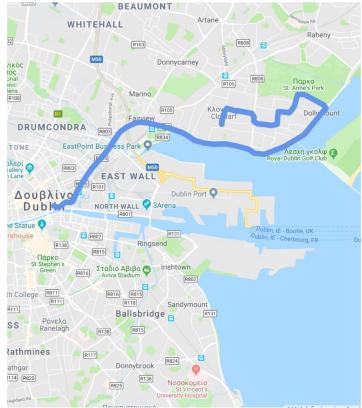
Test Trip 5
 $\Delta t = 113.921000004$



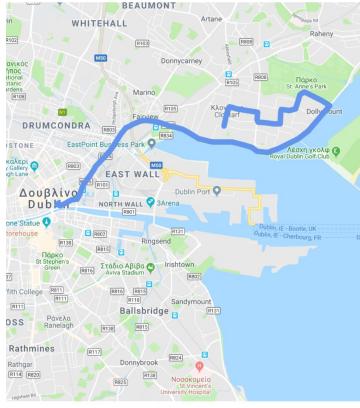
Neighbor 1
JP_ID 01300001
DTW 0.0



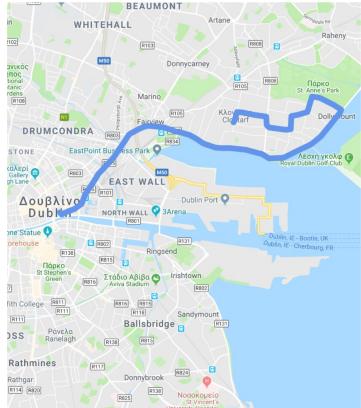
Neighbor 2
JP_ID 01300001
DTW 30147567111



Neighbor 3
JP_ID 01300001
DTW 6.49882195023



Neighbor 4
JP_ID 01300001
DTW 6.5043985644

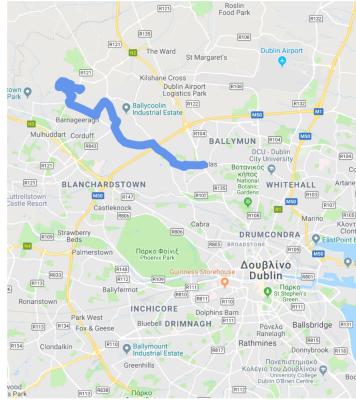


Neighbor 5
JP_ID 01300001
DTW 6.63023162718

Ερώτημα α2: Εύρεση κοντινότερων υποδιαδρομών

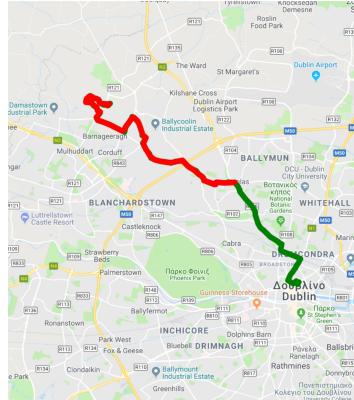
Στο ερώτημα αυτό εντοπίσαμε τα k κομμάτια των διαδρομών του αρχείου *test set a2* που είναι παρόμοια με τις διαδρομές του αρχείου *train set*. Χρησιμοποιήθηκε η *haversine* απόσταση για τον έλεγχο ομοιότητας ($<= 200m$). Και ο αλγόριθμος *LCSS* (δυναμικός προγραμματισμός) του οποίου ο φευδοκώδικας πάρθηκε έτοιμος και προσαρμόστηκε στις ανάγκες του ερωτήματος.

Παραθέτουμε τα αποτελέσματά μας



Test Trip 1

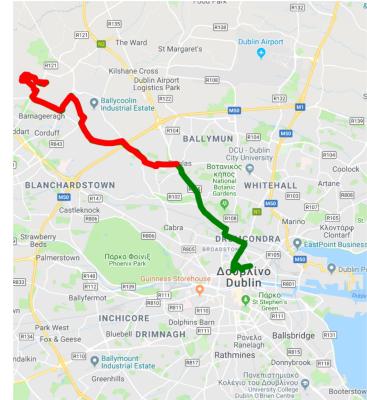
$\Delta t = 591.786999941$



Neighbor 1

JP_ID: 040D1002

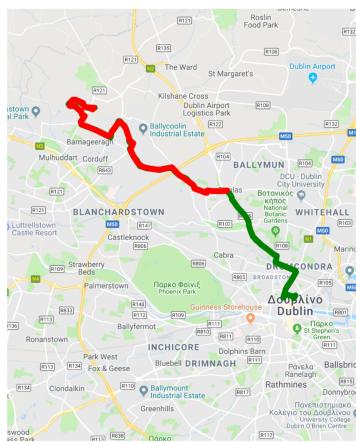
#Matching Points: 77



Neighbor 2

JP_ID: 040D1002

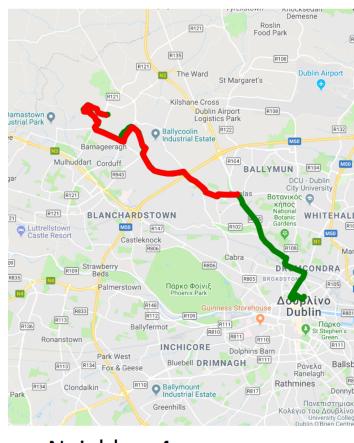
#Matching Points: 75



Neighbor 3

JP_ID: 040D1002

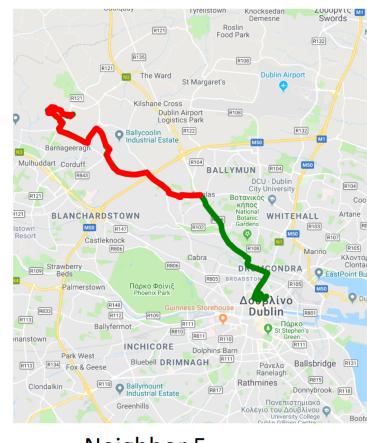
#Matching Points: 74



Neighbor 4

JP_ID: 040D1002

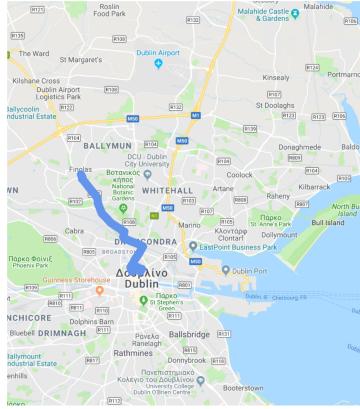
#Matching Points: 72



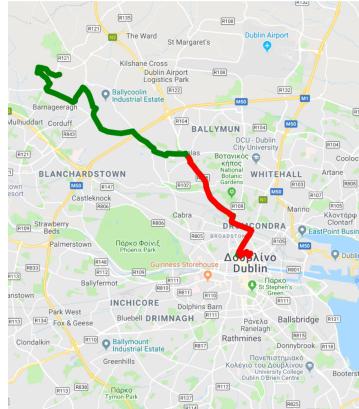
Neighbor 5

JP_ID: 040D1002

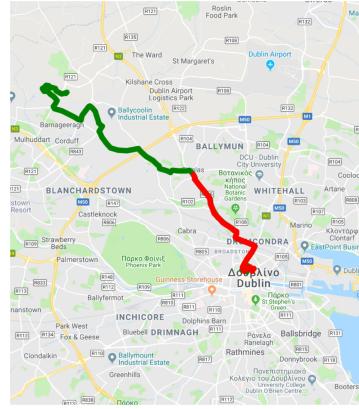
#Matching Points: 71



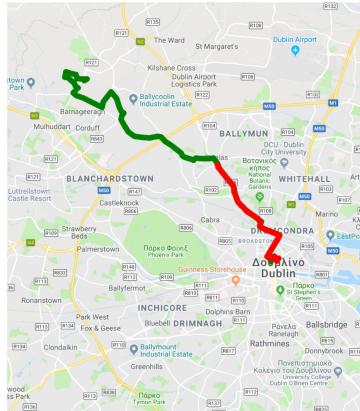
Test Trip 2
 $\Delta t = 634.562000036$



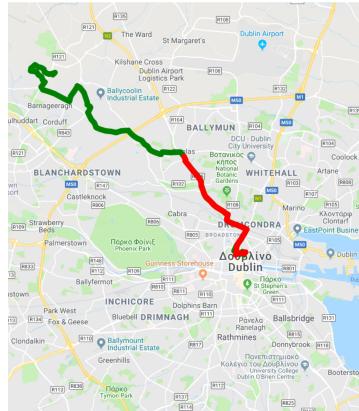
Neighbor 1
JP_ID: 040D1002
#Matching Points: 81



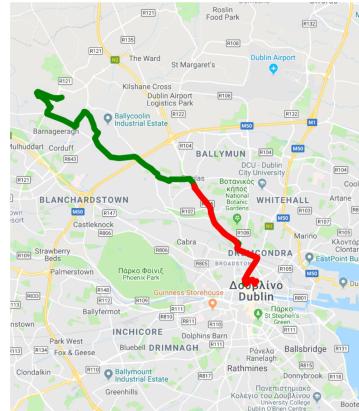
Neighbor 2
JP_ID: 040D1002
#Matching Points: 77



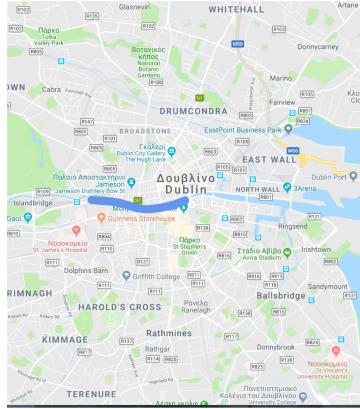
Neighbor 3
JP_ID: 040D1002
#Matching Points: 74



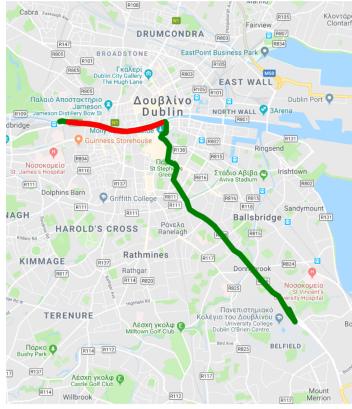
Neighbor 4
JP_ID: 040D1002
#Matching Points: 72



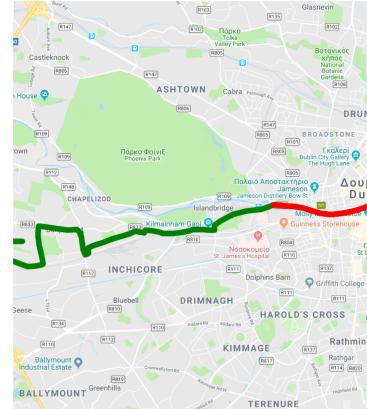
Neighbor 5
JP_ID: 040D1002
#Matching Points: 71



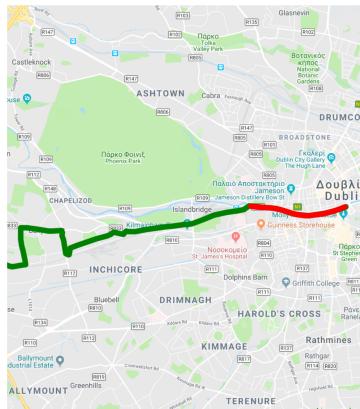
Test Trip 3
 $\Delta t = 303.467000008$



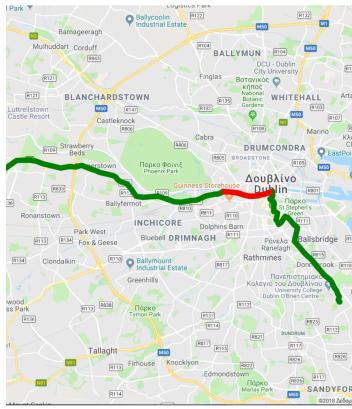
Neighbor 1
JP_ID: 01451008
#Matching Points: 40



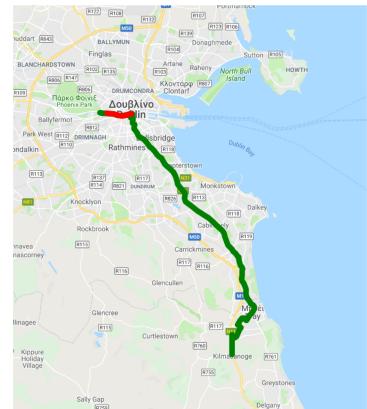
Neighbor 2
JP_ID: 00790001
#Matching Points: 40



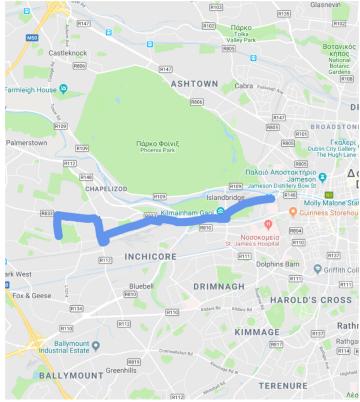
Neighbor 3
JP_ID: 079A0001
#Matching Points: 40



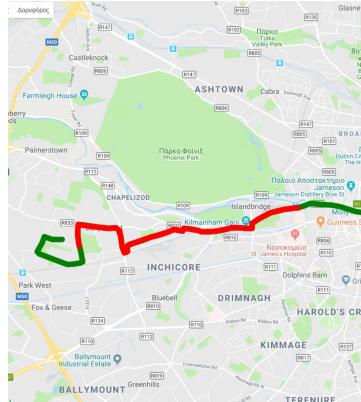
Neighbor 4
JP_ID: 067X0001
#Matching Points: 40



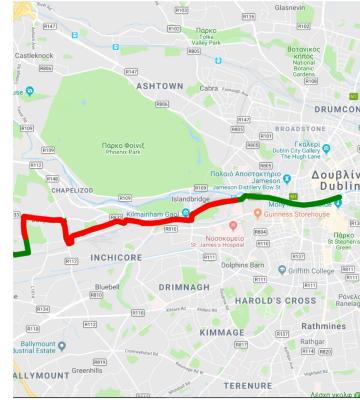
Neighbor 5
JP_ID: 01451001
#Matching Points: 39



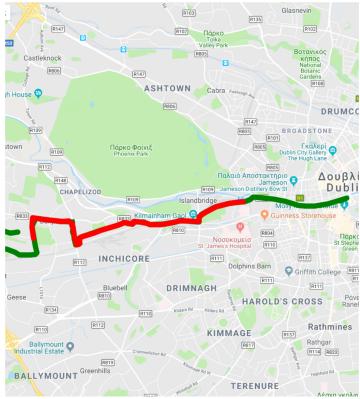
Test Trip 4
 $\Delta t = 449.243999958$



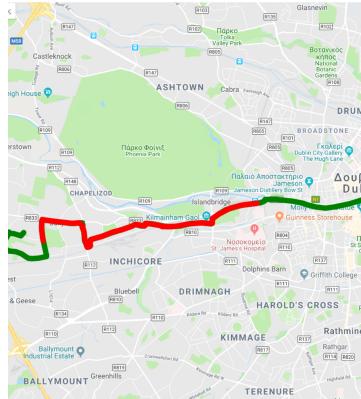
Neighbor 1
JP_ID: 00790001
#Matching Points: 59



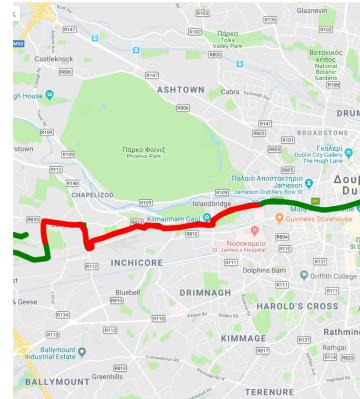
Neighbor 2
JP_ID: 00790001
#Matching Points: 57



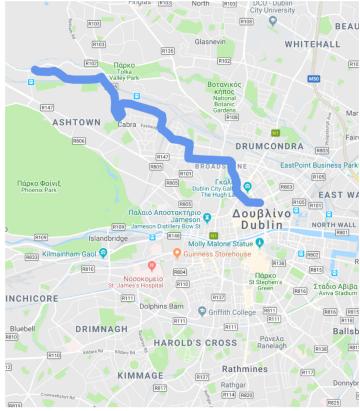
Neighbor 3
JP_ID: 00790001
#Matching Points: 57



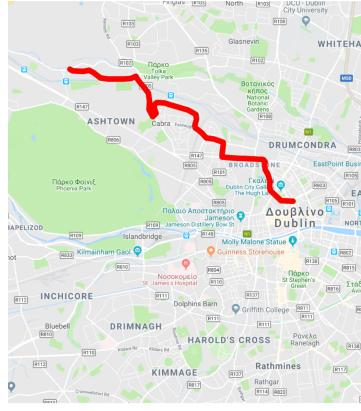
Neighbor 4
JP_ID: 00790001
#Matching Points: 57



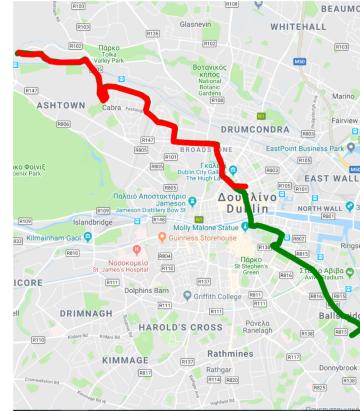
Neighbor 5
JP_ID: 00790001
#Matching Points: 56



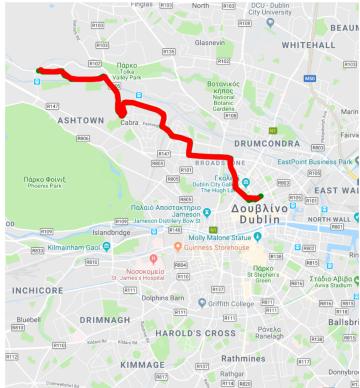
Test Trip 5
 $\Delta t = 560.739000082$



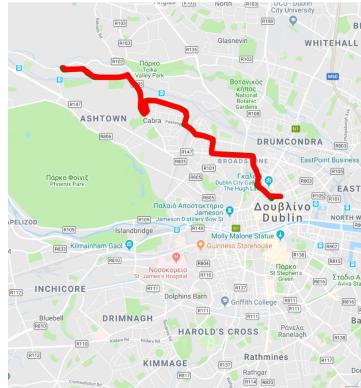
Neighbor 1
JP_ID: 01200001
#Matching Points: 73



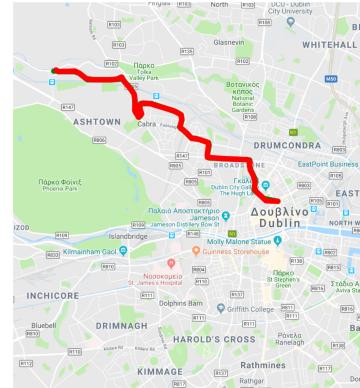
Neighbor 2
JP_ID: 01200002
#Matching Points: 71



Neighbor 3
JP_ID: 01200001
#Matching Points: 71



Neighbor 4
JP_ID: 01200001
#Matching Points: 71



Neighbor 5
JP_ID: 01200001
#Matching Points: 70

3. Κατηγοριοποίηση

Για το ερώτημα αυτό χρησιμοποιήθηκε ο αλγόριθμος KNN της πρώτης εργασίας με τις αναγκαίες τροποποιήσεις. Αρχικά κατηγοριοποιήθηκε το $testSet$ και γράφτηκε στο αντίστοιχο .csv της εκφώνησης. Στη συνέχεια πραγματοποιήθηκε το $10 - Crossfold validation$ για το 5% του dataset με αποτέλεσμα να βγάλει 0.03% accuracy. Για μεγαλύτερο δείγμα (10%) το accuracy ήταν 0.06%. Τα αποτελέσματα αυτά είναι λογικά γιατί έχουμε να κατηγοριοποιήσουμε 5 test instances σε 416 JPid, και το $trainSet$ είναι σχετικά μικρό.

*

Για την ανάπτυξη του προγράμματος ακολουθήθηκαν οι βασικές αρχές του *software engineering*, οπότε υπάρχει το αρχείο *helpers.py* στο οποίο είναι όλες οι κοινόχρηστες συναρτήσεις των ερωτημάτων.

Οι χρόνοι εκτέλεσης είναι γραμμένοι στις εικόνες των ερωτημάτων α1,α2. Ως αναφορά το ερώτημα 3 δοκιμάστηκε με δύο διαφορετικά ποσοστά (5,10 %) και σε δύο διαφορετικά λειτουργικά συστήματα. Συγκεκριμένα : Σε *Windows10* (χωρίς παραλληλία, γιατί δεν υποστηρίζεται η εντολή *fork*, και σε επεξεργαστή *i7 – 2200*) για το *Crossvalidation* έτρεχε περίπου 3 ώρες μόνο για το 5%. Σε *Linux(Ubuntu)* με 4 – *jobs* έτρεχε περίπου 10 λεπτά για το 5% και περίπου 1 ώρα για το 10% με 8 *jobs*.