

Τεχνικές Εξόρυξης Δεδομένων  
2η Άσκηση  
Εαρινό Εξάμηνο 2018

Μανούσος Τοράκης  
Α.Μ: 1115201400201

Ελευθέριος Αργύριος Καραμπάς  
Α.Μ: 1115201400064

3 Ιουνίου 2018

## Περιεχόμενα

<b>1 Εισαγωγή</b>	<b>2</b>
<b>2 Δομή Εργασίας</b>	<b>2</b>
<b>3 Οπτικοποίηση των Δεδομένων</b>	<b>4</b>
<b>4 Υλοποίηση KNN</b>	<b>6</b>
<b>5 Εύρεση κοντινότερων γειτόνων</b>	<b>7</b>
<b>6 Εύρεση κοντινότερων υποδιαδρομών</b>	<b>11</b>
<b>7 Κατηγοριοποίηση</b>	<b>15</b>
<b>8 Cross Validation</b>	<b>15</b>

# 1 Εισαγωγή

Έχουμε ολοκληρώσει όλα τα ερωτήματα της εργασίας. Για την εύρεση των κοντινότερων γειτόνων  $k$ - $nn$  χρησιμοποιήσαμε δικιά μας υλοποίηση από την 1η Άσκηση, την οποία προσαρμόσαμε έτσι ώστε να βελτιώσουμε τον χρόνο εκτέλεσης και να πετύχουμε καλύτερη αξιοποίηση της μνήμης.

# 2 Δομή Εργασίας

To project χωρίζεται στα εξής αρχεία/φακέλους:

- map  
Περιέχει τα *html* αρχεία του ερωτήματος Οπτικοποίηση των Δεδομένων. Η ονοματολογία που ακολουθήσει για τα αρχεία είναι: *map\_{journeyPatternID}\_{TripID}*
- test\_dataset  
Περιέχει τα *dataset* αρχεία για τα ερωτήματα τις άσκησεις.
- Test\_Subways
  - Test\_Subways\_1
  - Test\_Subways\_2
  - Test\_Subways\_3
  - Test\_Subways\_4
  - Test\_Subways\_5
- Old\_Test\_Subways
  - Test\_Subways\_1
  - Test\_Subways\_2
  - Test\_Subways\_3
  - Test\_Subways\_4
  - Test\_Subways\_5
- Περιέχει τα αρχικά αποτελέσματα για το ερώτημα: Εύρεση κοντινότερων υποδιαδρομών, το οποίο το είχαμε υποστηρίξει τη μέγιστη κοινή υποακολουθία ως άνθροισμα μικρότερων υποδιαδρομών που μπορεί να είχαν μεταξύ τους κένα, αλλά μετά την διευχρύνηση στο *piazza* άλλαξε η υλοποίηση όπως ζητήθηκε.
- Test\_Trip
  - Test\_Trip\_1

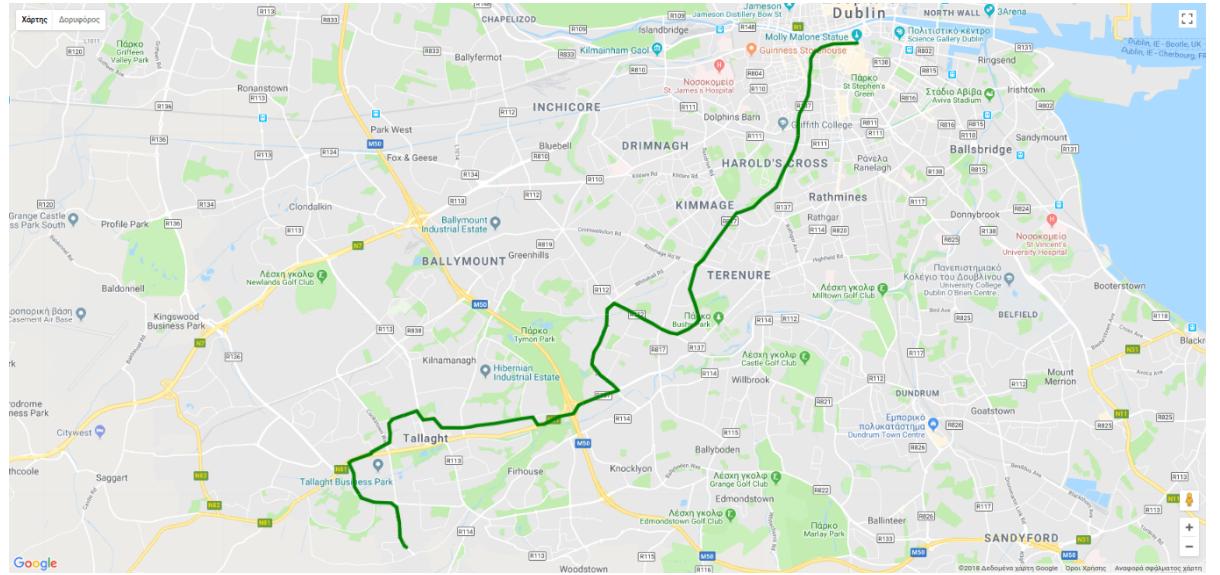
- Test\_Trip\_2
- Test\_Trip\_3
- Test\_Trip\_4
- Test\_Trip\_5

Περιέχει τα αποτελέσματα για το ερώτημα: Εύρεση κοντινότερων γειτόνων και κάθε υποφάκελος περιέχει τα 5 *html* αρχεία των κοντινότερων υποδιαδρομών, το *html* αρχείο της διαδρομής για την οποία φάχνουμε τους γείτονες και το αρχείο *results* που περιέχει την καταγραφή των αποτελεσμάτων.

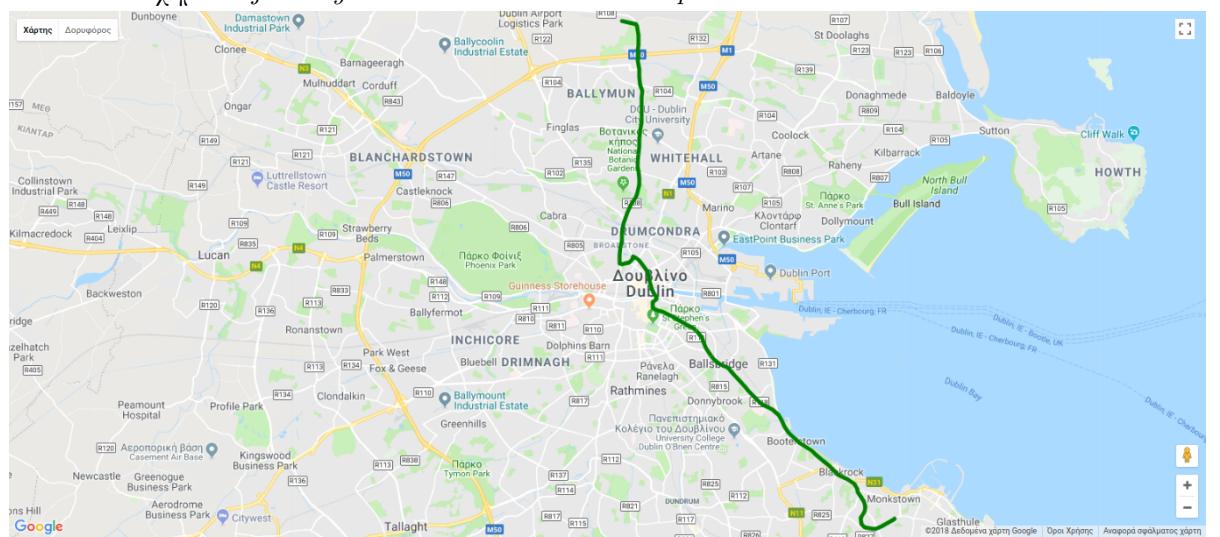
- visualization\_map.py  
Το οποίο υλοποιεί το ερώτημα Οπτικοποίηση των Δεδομένων.
- closest\_neighbors.py  
Το οποίο υλοποιεί το ερώτημα Εύρεση κοντινότερων γειτόνων.
- closest\_subways.py  
Το οποίο υλοποιεί το ερώτημα Εύρεση κοντινότερων υποδιαδρομών.
- DTW.py  
Το οποίο περιέχει την υλοποιήση της μετρικής *DTW*, όπως αυτή αναλύεται στο σύνδεσμο: [https://en.wikipedia.org/wiki/Dynamic\\_time\\_warping#Implementation](https://en.wikipedia.org/wiki/Dynamic_time_warping#Implementation)
- LCSS.py  
Έχει υλοποιηθεί ο αλγόριθμος *LCSS*, όπως αυτός επεξηγείται στο σύνδεσμο: [https://en.wikipedia.org/wiki/Longest\\_common\\_subsequence\\_problem#LCS\\_function\\_defined](https://en.wikipedia.org/wiki/Longest_common_subsequence_problem#LCS_function_defined)
- KNN.py  
Τυλοποιείται ο *KNN*, τον οποίου είχαμε δημιουργήσει στην πρώτη άσκηση του μαθήματος και του προσθέσαμε νέες λειτουργίες για να είναι λειτουργικός και για την άσκηση αυτή.
- Harvesine.py  
Τυλοποιείται ο *Haversine*, του οποίου η υλοποίηση βρέθηκε έτοιμη από: <https://gist.github.com/rochacbruno/2883505>
- classification.py  
Το οποίο περέχει κώδικα για το τελευταίο ερώτημα της άσκησης, την Κατηγοροποίηση.
- cross\_validation.py  
Το οποίο περέχει κώδικα για να κάνουμε 10-fold cross validation με το 10% των δεδομένων μας.
- testSet\_JourneyPatternIDs.csv  
Το αρχείο αυτό περιέχει τα αποτελέσματα του *classification*. Η αρίθμηση των διαδρομών (*TripID*) ξεκινάει από το 1 έως το 5.

### 3 Οπτικοποίηση των Δεδομένων

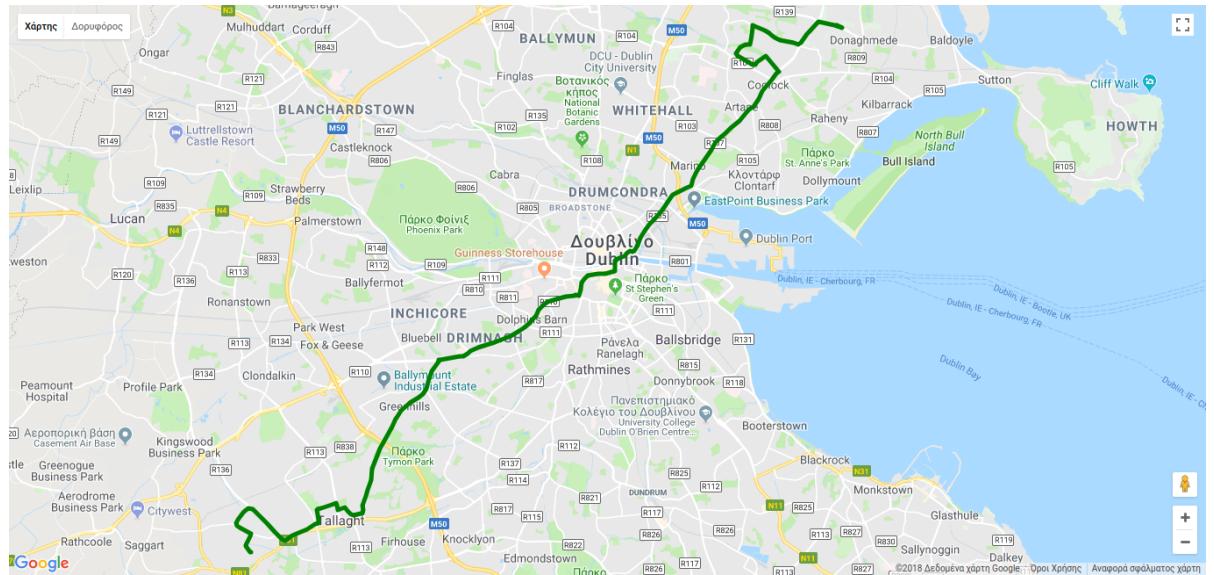
Για την οπτικοποίηση των διαδρομών χρησιμοποιήσαμε την βιβλιοθήκη *gmpplot* όπως ζητήθηκε ενώ επιλέξαμε 5 τυχαίες διαδρομές χρησιμοποιώντας την συνάρτηση *sample* της βιβλιοθήκης *random*. Οι σελίδες *html* αποθηκεύονται στον φάκελο *map*. Η ονοματολογία που ακολουθήθηκε για τα αρχεία είναι: *map\_{journeyPatternID}\_{TripID}*



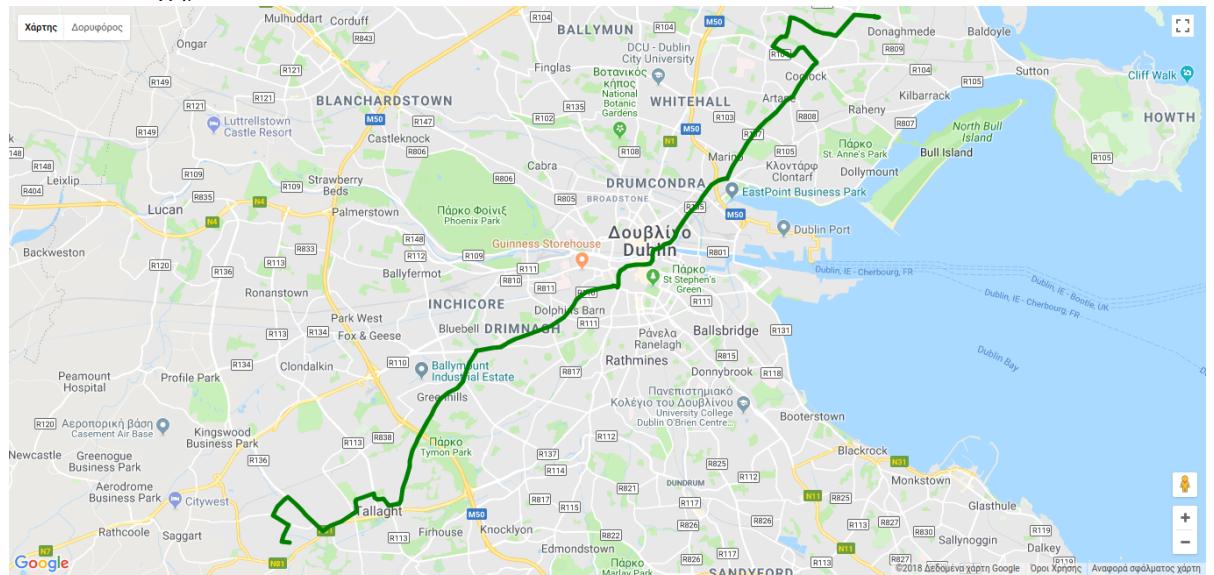
Σχήμα 1: *journeyPatternID*: 054A0001 - *TripID*: 2488



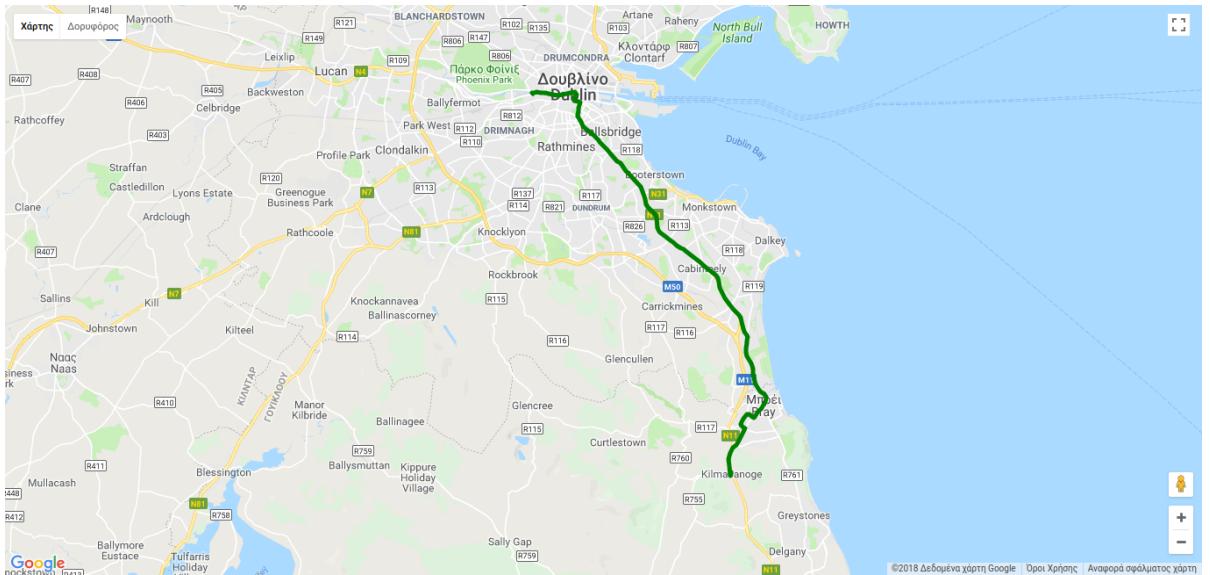
Σχήμα 2: *journeyPatternID*: 00040001 - *TripID*: 4294



Σχήμα 3: *journeyPatternID: 00271001 - TripID: 1574*



Σχήμα 4: *journeyPatternID: 00271001 - TripID: 283*



Σχήμα 5: *journeyPatternID*: 01450001 - *TripID*: 5105

## 4 Υλοποίηση *KNN*

Στην υλοποίηση μας η μετρική δίνεται ως παράμετρος στον *constructor* ενώ προσθέσαμε την δυνατότητα να ορίζεται μοναδικό σημείο για το οποίο αναζητούμε γείτονες (*origin*). Επίσης μέσω του ορίσματος *reverse* προσθέσαμε την δυνατότητα να βρίσκονται οι 'μακρύτεροι' γείτονες, δηλαδή εκείνοι για τους οποίους η μετρική επιστρέφει την μεγαλύτερη τιμή.

Έτσι με σκοπό την αποδοτικότερη χρήση της μνήμης, στο ερώτημα 2, διατρέχουμε όλα τα σημεία από το *test\_set* και για κάθε ένα από αυτά:

- Το ορίζουμε ως *origin*.
- Υπολογίζουμε τους γείτονες του καλώντας την συνάρτηση *calculate\_neighboor* για κάθε ένα από τα σημεία του *train\_set*.

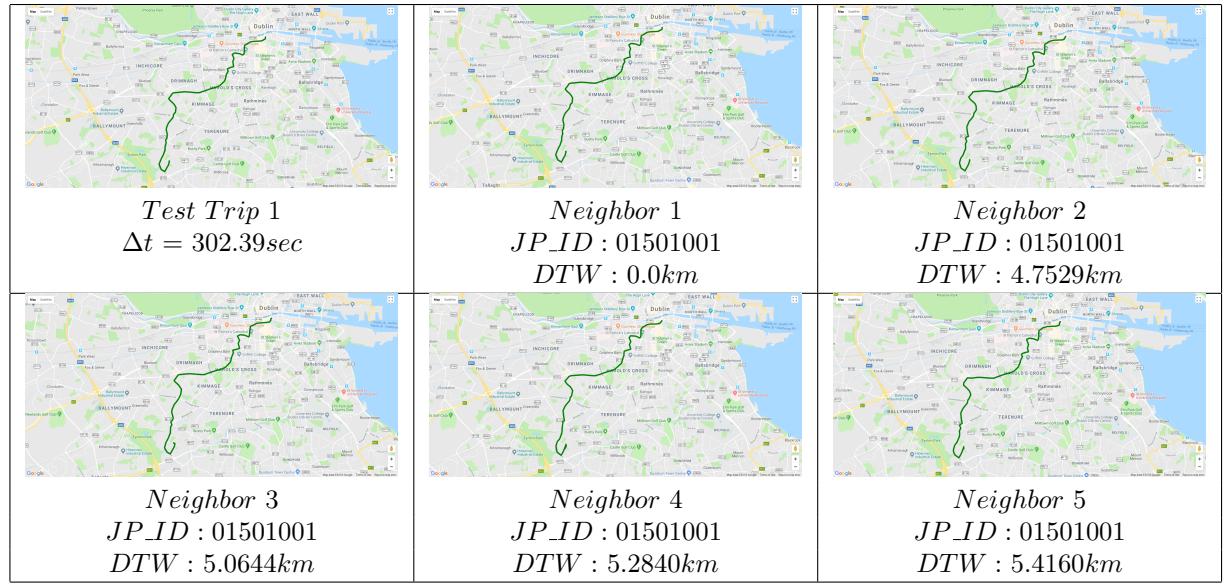
Όσον αφορά τον τρόπο που υπολογίζουμε τους γείτονες στην *calculate\_neighboor*, κρατάμε ανα πάσα στιγμή μόνο τους *k*-καλύτερους (κοντινότερους/μακρύτερους) γείτονες ταξινομημένους σε μια λίστα, την οποία ενημερώνουμε μόνο όταν ένας καινούργιος γείτονας είναι 'καλύτερος' από τον μέχρι τώρα 'χειρότερο' (τον τελευταίο της λίστας).

Συμπερασματικά πετυχαίνουμε καλύτερη αξιοποίηση της μνήμης αφού δεν κρατάμε όλο το *train\_set* στην μνήμη και κρατάμε μόνο τους *k* κοντινότερους γείτονες/αποστάσεις κάθε στιγμή. Επίσης δεν ταξινομούμε άσκοπα τους γείτονες αφού ελέγχουμε αν ο καινούργιος είναι καλύτερος από τον μέχρι τότε χειρότερο πρωτού τον προσθέσουμε στην λίστα.

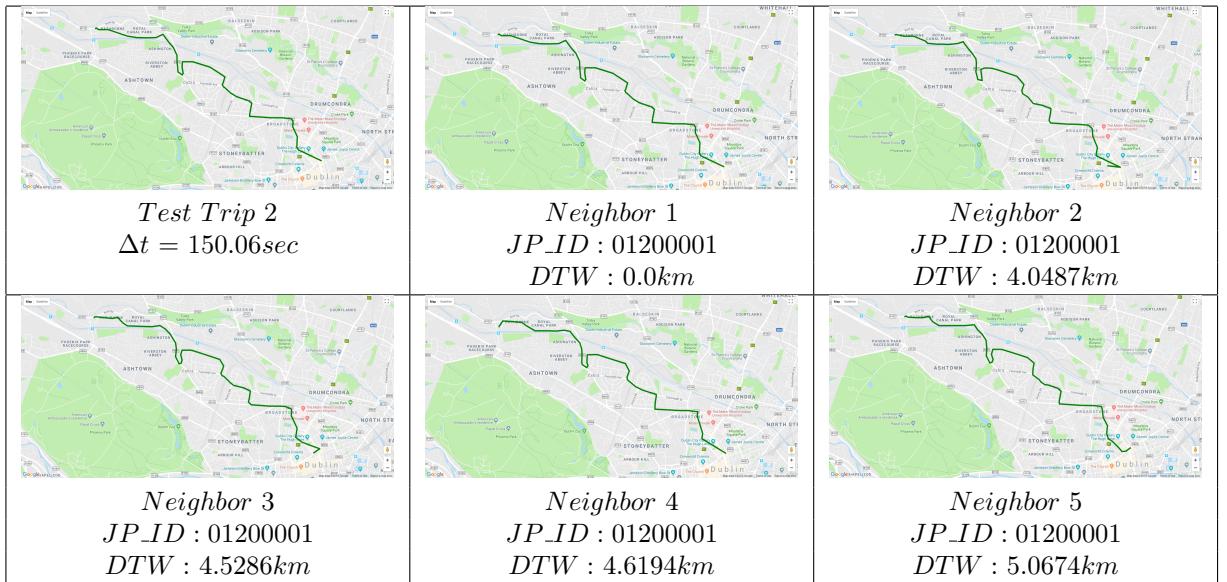
## 5 Εύρεση κοντινότερων γειτόνων

Για την επίλυση του ερωτήματος αυτού, υλοποιήθηκε από μας ο *DTW*. Τον τύπο *Haversine* τον πήραμε έτοιμο όπως αναφέρεται και παραπάνω. Τα αποτελέσματα κάθε ερωτήματος(*query*) αποθηκεύονται στον φάκελο *Test\_Trip/Test\_Trip-{X}*, όπου *X* ο αριθμός κάθε *query*.

Πίνακας 1: *Test Trip 1*



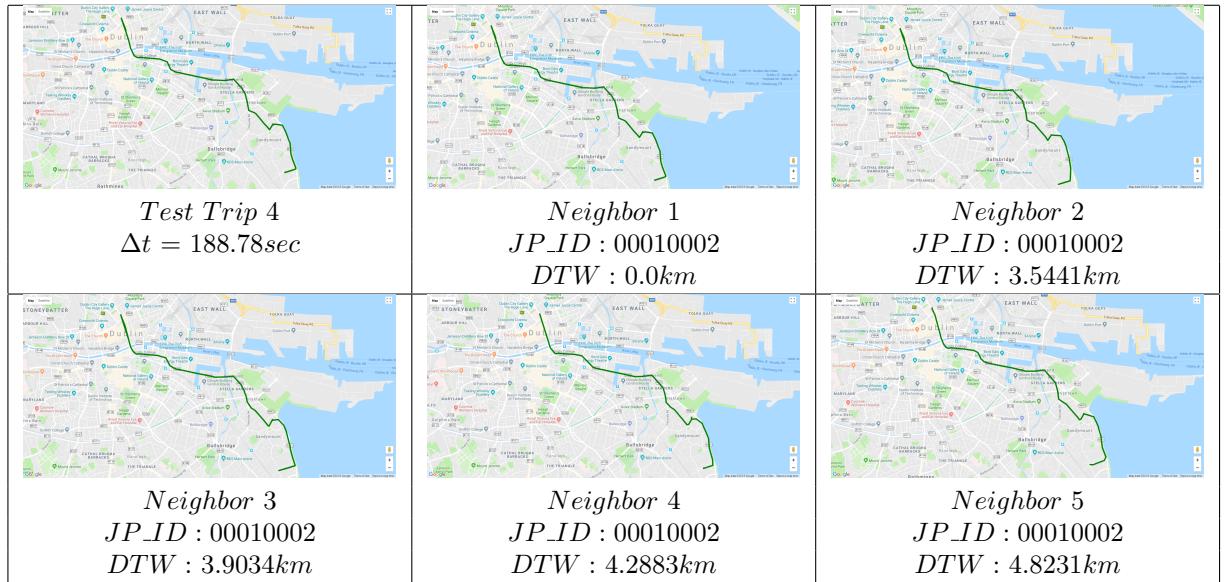
Πίνακας 2: *Test Trip 2*



Πίνακας 3: *Test Trip 3*



Πίνακας 4: *Test Trip 4*



Πίνακας 5: *Test Trip 5*



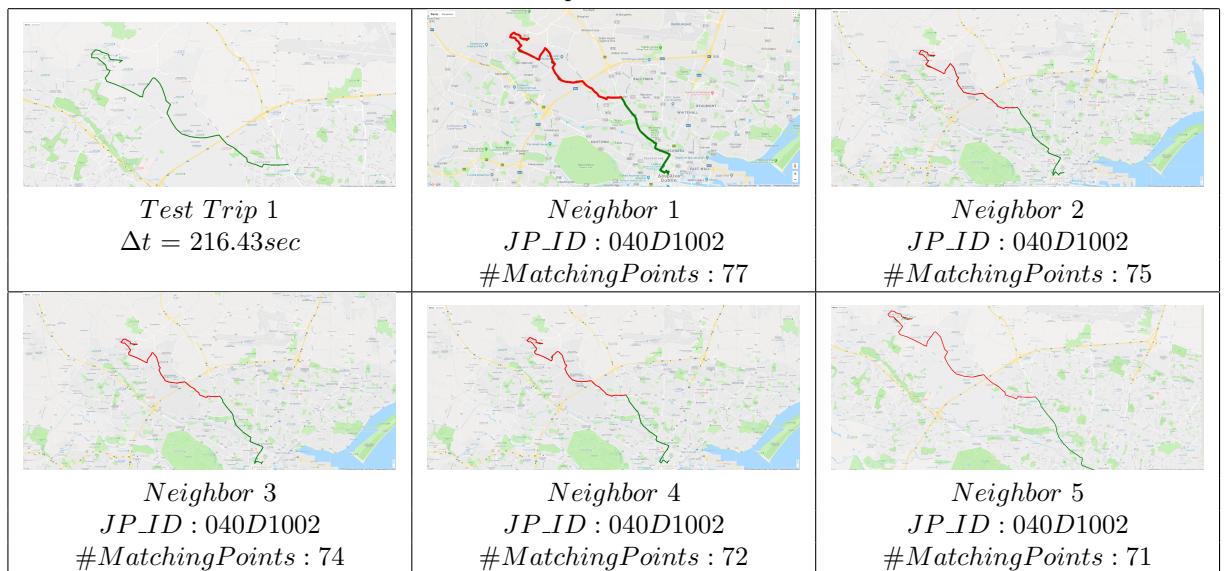
## 6 Εύρεση κοντινότερων υποδιαδρομών

Για την επίλυση του ερωτήματος αυτού, υλοποιήθηκε από μας ο *LCSS*. Τον τύπο *Haversine* τον πήραμε έτοιμο όπως αναφέρεται και παραπάνω. Τα αποτελέσματα κάθε ερωτήματος(*query*) αποθηκεύονται στον φάκελο *Test\_Subways/Test\_Subways-{X}*, όπου *X* ο αριθμός κάθε *query*.

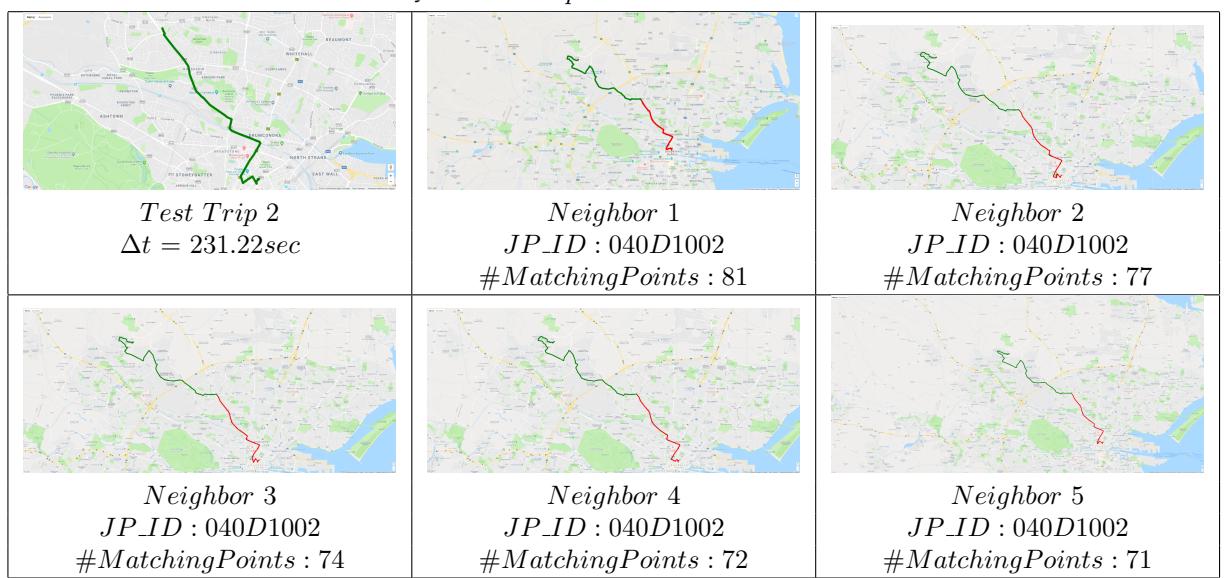
Στην αρχή η υλοποίηση που δημιουργήσαμε είχε ως αποτελέσματα οι υποδιαδρομές που φαίνοντουσαν δεν πάρθηκαν ως συνεχόμενες τροχιές, αλλά ως άνθροισμα των υποδιαδρομών πάνω στην τροχία. Μπορείτε να τις δείτε αν θέλετε στον φάκελο *Old\_Test\_Subways*, τις οποίες τις έχουμε κρατήσει αφού τις είχαμε δημιουργήσει.

Στη συνέχεια διαβάσαμε από το *piazza* πως θέλατε να είναι η εμφάνιση της υποδιαδρομής και τροποποιήθηκε καταλλήλως.

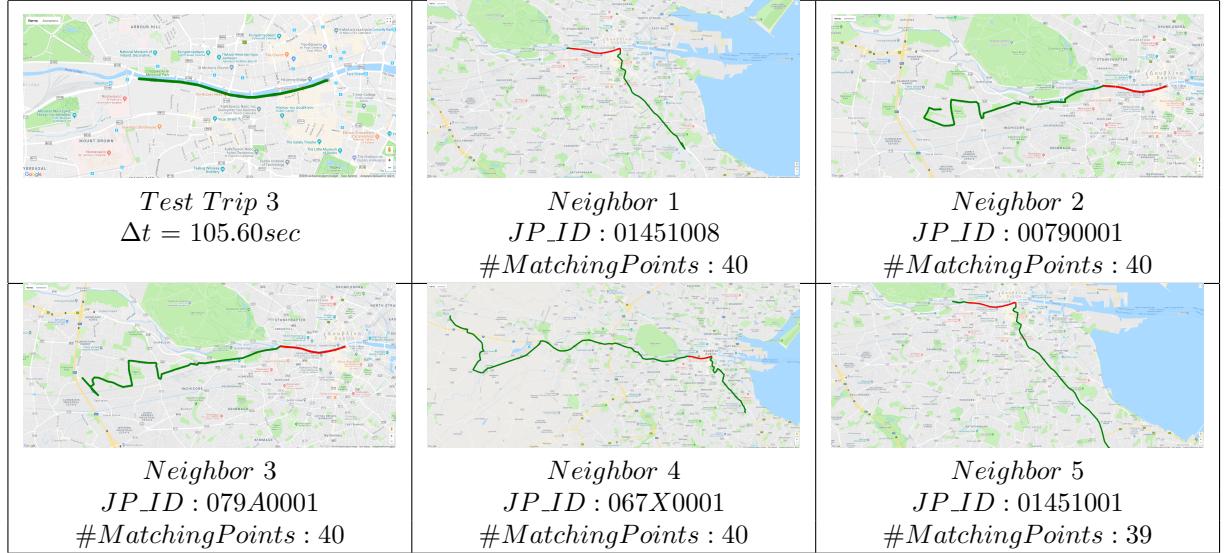
Πίνακας 6: *Test Trip 1*



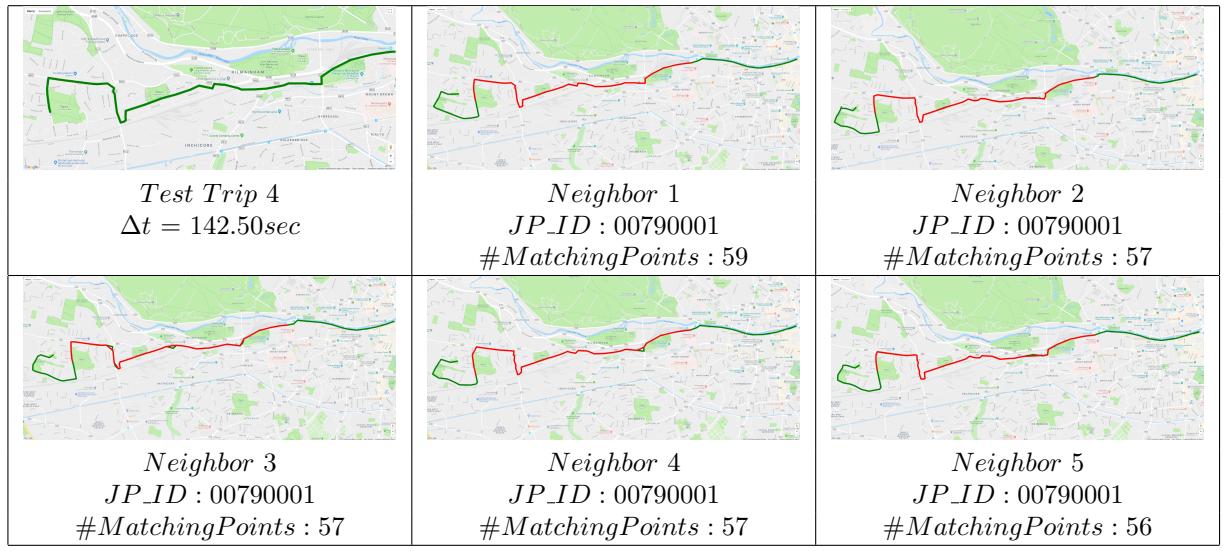
Πίνακας 7: *Test Trip 2*



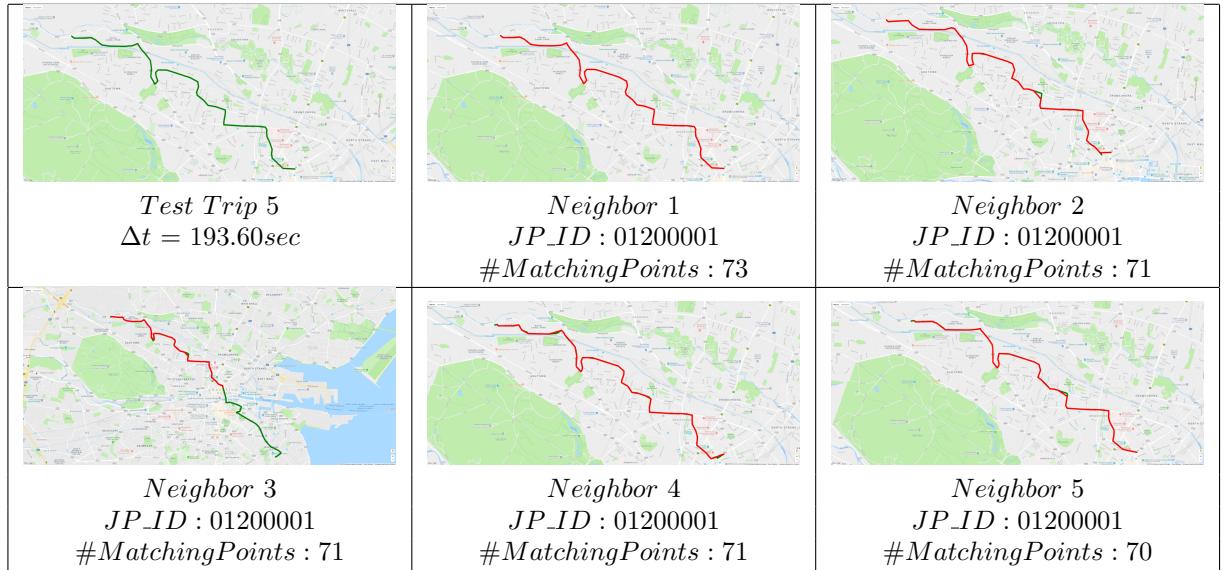
Πίνακας 8: *Test Trip 3*



Πίνακας 9: *Test Trip 4*



Πίνακας 10: *Test Trip 5*



## 7 Κατηγοριοποίηση

<i>Test.Trip.ID</i>	<i>Predicted_JourneyPatternID</i>
1	040D1002
2	01201001
3	00371001
4	00790001
5	01200001

## 8 Cross Validation

Δοκιμάσαμε να κάνουμε 10-*fold cross validation* για ένα μικρό κομμάτι των δεδομένων μας. Το οποίο αργούσε δραματικά πολύ για 5%-10% των δεδομένων. Επίσης όσο πιο λίγα ήταν τα δεδομένα που δοκιμάζαμε τόσο πιο χαμηλά ποσοστά παίρναμε καθώς δεν ήταν αρκετά για να κάνουμε *train* το μοντέλος μας.

Για 100 τροχιές που δοκιμάσαμε η μετρική *Accuracy* είχε ως αποτέλεσμα το απογοητευτικό ποσοστό 12%. Για το 5% των δεδομένων (327 τροχιές) που δοκιμάσαμε είχαμε ως αποτέλεσμα περίπου 73%. Τέλος για το 10% των δεδομένων, η μετρική *Accuracy* είχε ως αποτέλεσμα 81%. Οστόσο για να πάρουμε αποτελέσματα έπρεπε να αφήσουμε να τρέχει ακόμα και 1-2 μέρες, καθώς για μεγάλο αριθμό σημείων ανα τροχία και πολλές τροχιές, ο υπολογισμός απαιτούσε αρκετό χρόνο για όλες τα *folds* λόγω του *DTW*.

Ο λόγος που με τόσο μικρό δείγμα έχουμε γενικά χαμηλό ποσοστό στο *Accuracy* οφείλεται οτι στο διεπειδή από το συνολικό *dataset* λαμβάνουμε τυχαία κάποιο πολύ μικρό κομμάτι του, πολλές τροχιές μπορεί να μην έχουν κάποια που να ταιριάζουν έστω και λίγο μεταξύ τους, οπότε το μοντέλο δεν εκπαιδεύεται επαρκώς. Εκεί οφείλεται και πως από το 12% πήγαμε στο 73% με αύξηση του δείγματος κάτα 227 τροχιές.