ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2017-18

ΤΕΧΝΙΚΕΣ ΑΝΑΛΥΣΗΣ ΔΕΔΟΜΕΝΩΝ ΥΨΗΛΗΣ ΚΛΙΜΑΚΑΣ

Κατηγοριοποίηση χωρο-χρονικών δεδομένων

Κανακάκης Παναγιώτης - M1516 - pkanakakis@di.uoa.gr
Γαλούνη Κωνσταντίνα - M1524 - kgalouni@di.uoa.gr

ΑΘΗΝΑ

Φεβρουάριος 2018

Πίνακας Περιεχομένων

Προ-επεξεργασία των Δεδομένων

Η εργασία αυτή αφορά την επεξεργασία και κατηγοριοποίηση χωροχρονικών δεδομένων, και πιο συγκεκριμένα τροχιές λεωφορείων στην ευρύτερη περιοχή του Δουβλίνου. Ως είσοδος δίνονται τα στίγματα που κατέγραψε ένα GPS σε κάθε λεωφορείο σε συγκεκριμένες χρονικές στιγμές. Συνεπώς, έχουμε στη διάθεσή μας σύνολα από αναγνωριστικό λεωφορείου, αναγνωριστικό δρομολογίου, χρονική στιγμή, γεωφραφικό μήκος και γεωγραφικό πλάτος.
Για την εξαγωγή ολόκληρων διαδρομών από τα παραπάνω δεδομένα, αρχικά με τη βοήθεια της συνάρτησης dropna() αφαιρούμε από το αρχείο όλες τις καταχωρήσεις για τις οποίες δεν είναι διαθέσιμες όλες οι τιμές.
Στη συνέχεια, διατρέχουμε όλο το αρχείο και για κάθε νέο συνδυασμό λεωφορείου - δρομολογίου δίνουμε ένα νέο χαρακτηριστικό διαδρομής. Σε περίπτωση που ο συνδυασμός λεωφορείου - δρομολογίου έχει ήδη καταχωρηθεί ως διαδρομή και δεν υπάρχει νέος συνδυασμός με κάποια από αυτές τις δύο τιμές, ανανεώνεται η διαδρομή εισάγοντας σε αυτήν το νέο στιγμιότυπο του GPS που διαβάσαμε μόλις από το αρχείο.
Στο τέλος αυτής της επεξεργασίας έχουμε ως αποτέλεσμα το αρχείο “trips.csv” με 7486 διαδρομές.

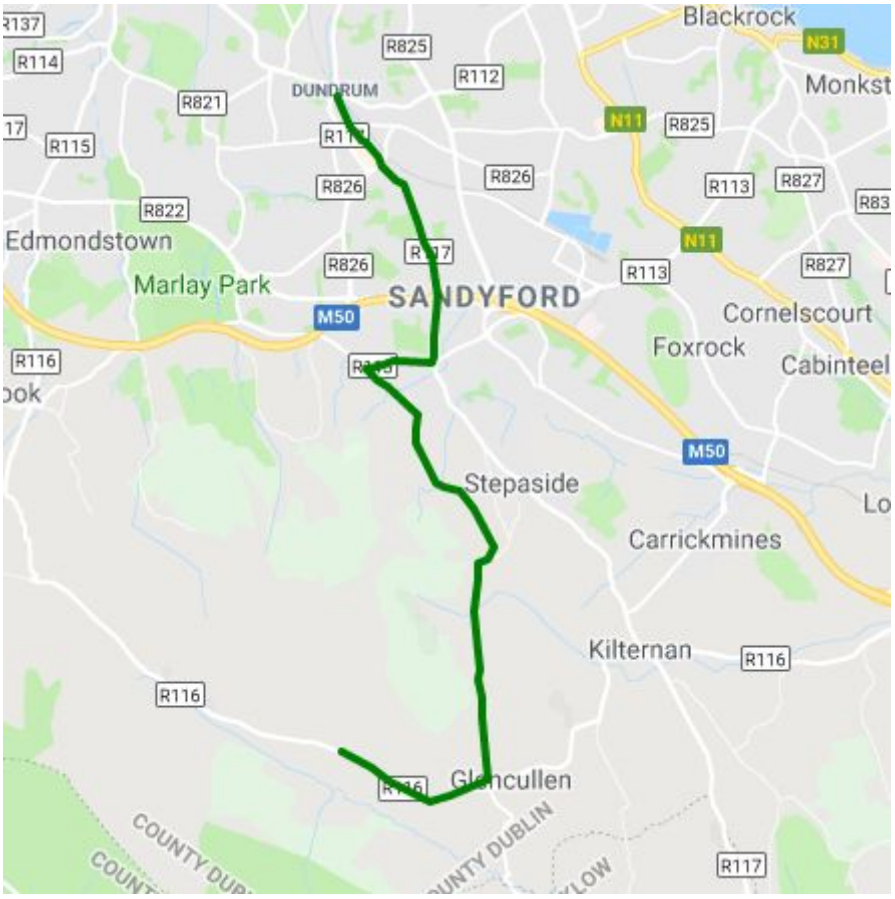
Καθαρισμός των Δεδομένων

Σε αυτό το στάδιο, στόχος είναι να παραμείνουν μόνο οι διαδρομές για τις οποίες ο θόρυβος δεν είναι πολύ μεγάλος. Ως θόρυβο σε αυτήν την περίπτωση ορίζουμε τόσο η συνολική απόσταση της διαδρομής να είναι μικρότερη από 2 χιλιόμετρα, όσο και η απόσταση μεταξύ δύο σημείων μιας διαδρομής να είναι μεγαλύτερη από 2 χιλιόμετρα. Στην πρώτη περίπτωση η διαδρομή είναι πολύ μικρή για να θεωρήσουμε πως αυτή ήταν μια πραγματική διαδρομή λεωφορείου, ενώ στη δεύτερη περίπτωση, η διαδρομή είχε “μεγάλα κενά”, συνεπώς, δε μπορούμε να γνωρίζουμε τι συνέβη στο ενδιάμεσο της διαδρομής.
Εάν η διαδρομή είχε μόνο 1 σημείο, απορρίπτεται διότι είναι μικρότερη από 2 χιλιόμετρα σίγουρα, ενώ σε αντίθετη περίπτωση, πραγματοποιείται έλεγχος με τη βοήθεια του τύπου haversine. Ο υπολογισμός της απόστασης αυτής έχει υλοποιηθεί στο αρχείο utils.py. Η haversine απόσταση είναι η ελάχιστη απόσταση μεταξύ δύο σημείων στην επιφάνεια μιας σφαίρας δεδομένων των συντεταγμένων τους. Η σφαίρα για τα δεδομένα του προβλήματος θεωρούμε πως είναι η Γη, με ακτίνα 6371 χιλιόμετρα.
Όπως αναφέρθηκε, τα δεδομένα που υπήρχαν στο αρχείο trips.csv είναι 7486, λόγω τη συνολικής απόστασης αφαιρέθηκαν από το αρχείο 214 εγγραφές και λόγω της μέγιστης απόστασης μεταξύ δύο σημείων 643 εγγραφές. Τελικά το αρχείο tripsClean.csv περιέχει 6629 διαδρομές. Οι μετρήσεις αυτές φαίνονται και στην παρακάτω εικόνα, που είναι μέρος της εκτέλεσης του παραδοτέου.

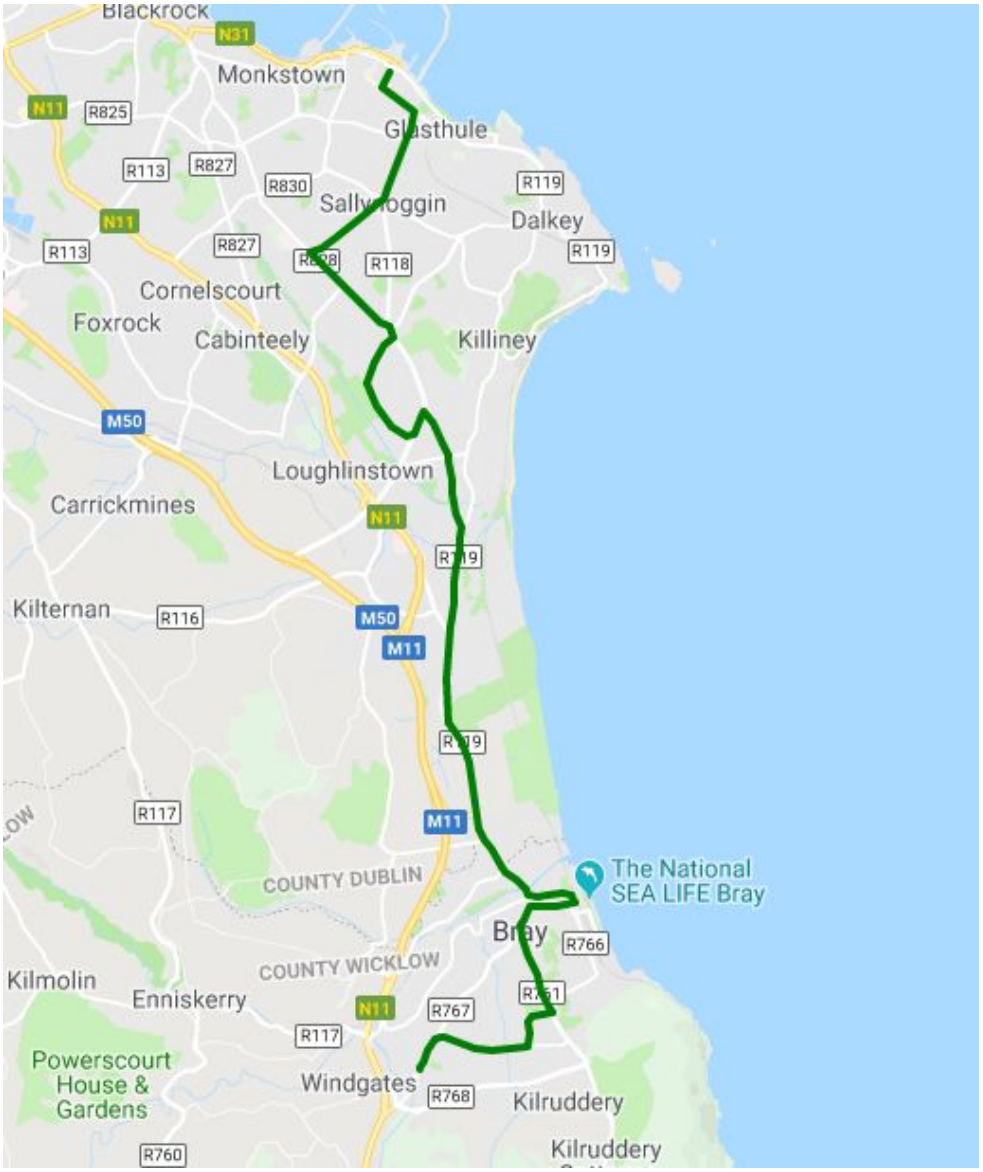
Οπτικοποίηση των Δεδομένων

Με χρήση της βιβλιοθήκης gmplot (όπως προτείνεται και στην εκφώνηση), έχουν οπτικοποιηθεί 5 τυχαίες διαδρομές από το αρχείο tripsClean.csv και αποθηκεύονται ως αρχεία html στο φάκελο plots με όνομα “tripID:XXXX_JourID:XXXXXXXX.html”, όπου X είναι τα αντίστοιχα χαρακτηριστικά διαδρομής και δρομολογίου.
Με το GoogleMapPlotter του gmplot απεικονίζεται ο χάρτης της Γης, με επίκεντρο και εστίαση σύμφωνα με τα ορίσματα. Ως επίκεντρο δίνονται τα σημεία στη μέση της διαδρομής που απεικονίζεται.
Παρακάτω φαίνονται τα κομμάτια των πέντε τυχαία επιλεγμένων διαδρομών όπως έχουν ζωγραφιστεί από το gmplot και το ποιές διαδρομές είναι.

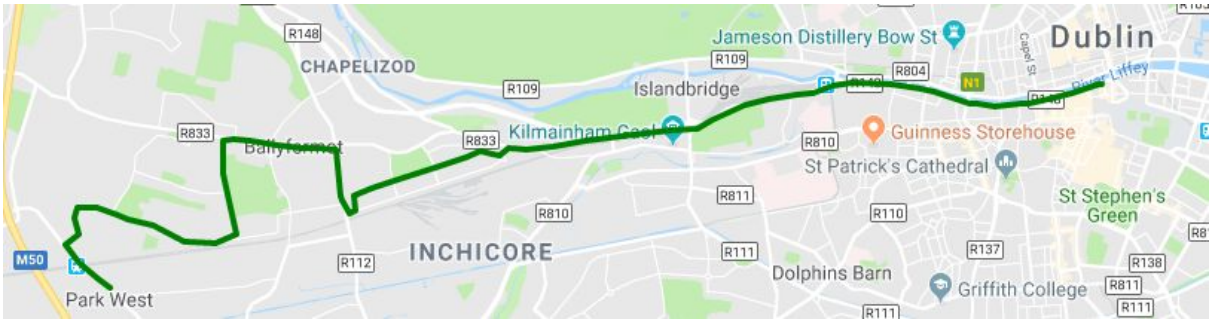
tripID: 953 - JournayID: 044B0001



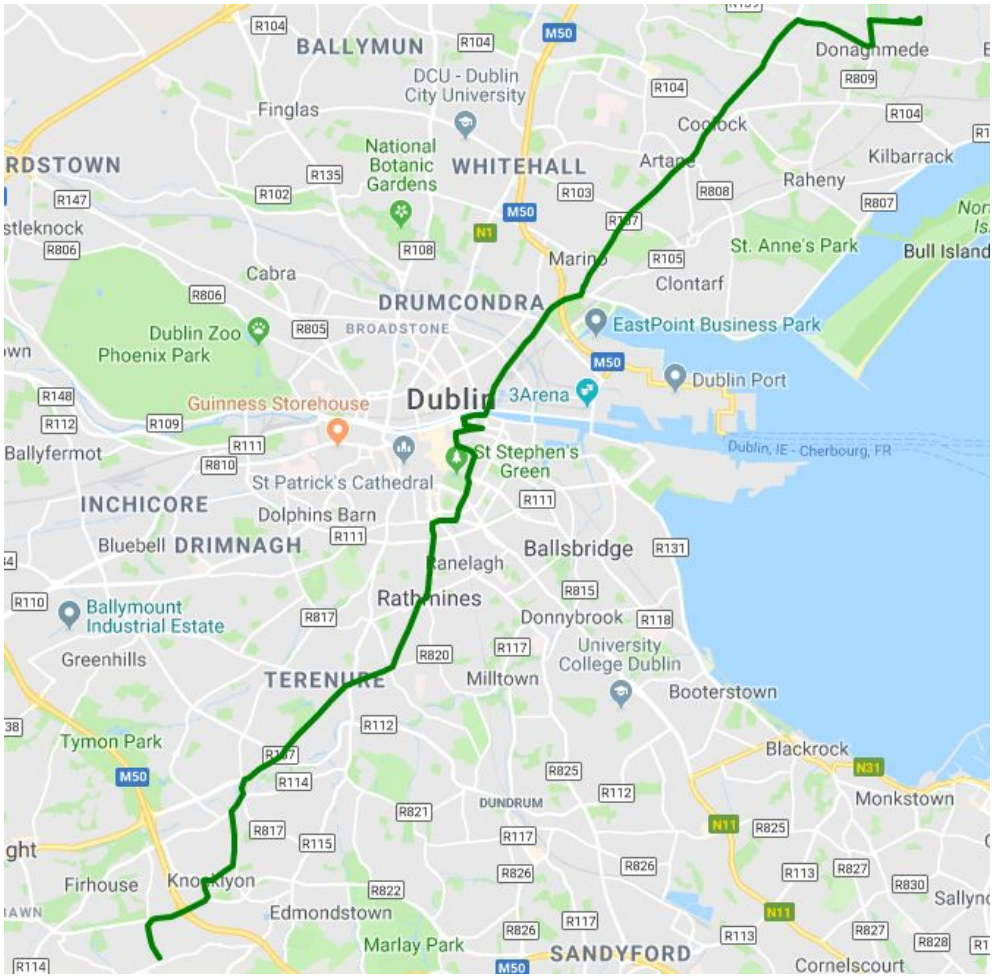
tripID: 1824 - JournayID: 045A0001



tripID: 2636 - JournayID: 079A0001



tripID: 3306 - JournayID: 00150001



tripID: 5731 - JournayID: 00591001

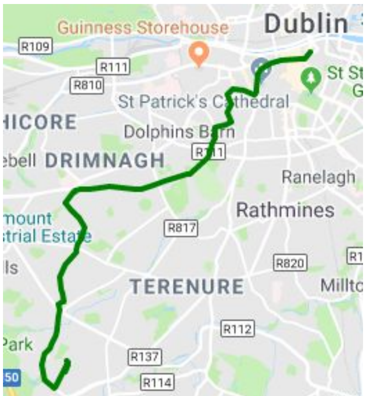
Εύρεση Κοντινότερων Γειτόνων

Σε αυτό το μέρος της εργασίας καλούμαστε να βρούμε τους πέντε κοντινότερους γείτονες για κάθε μία από τις πέντε διαδρομές του αρχείου test_set_a1.csv. Για να γίνει αυτό, υλοποιείται ο αλγόριθμος Dynamic Time Warping (DTW) με συνάρτηση απόστασης ξανά την haversine. Όπως παρατηρήθηκε, ο καλύτερος γείτονες για κάθε διαδρομή του test αρχείου, έχει απόσταση 0, είναι δηλαδή η ίδια διαδρομή. Για το λόγο αυτό, βρίσκουμε και οπτικοποιούμε τους 6 κοντινότερους γείτονες. Τα αρχεία που παράγονται για την οπτικοποίηση αποθηκεύονται στο φάκελο plots/dtw/i, όπου i είναι ο αριθμός της test διαδρομής ξεκινώντας από το 0. Στο φάκελο αυτό, η test διαδρομή έχει όνομα “dtw_route_i.html”, και οι γείτονές της “dtw_neighbour_j_for_route_i.html”.

Test Trip 1

Ο συνολικός χρόνος που απαιτήθηκε είναι 53.2077970505 sec.

Test Trip 1	Neighbor 1	Neighbor 2
	JP_ID: 01501001	JP_ID: 01501001
	Δt: 0.0184309482574	Δt: 0.0252001285553
	DTW: 0.0	DTW: 3.51027542309



Neighbor 3	Neighbor 4	Neighbor 5
JP_ID: 01501001	JP_ID: 01501001	JP_ID: 01501001
Δt: 0.045294046402	Δt: 0.0270788669586	Δt: 0.0206298828125
DTW: 3.79078798424	DTW: 3.99161597352	DTW: 4.11536447233

Neighbor 6 - JP_ID: 01501001 - Δt: 0.0327959060669 - DTW: 4.11995464624

Test Trip 2

Ο συνολικός χρόνος που απαιτήθηκε είναι 37.1940369606 sec.

Test Trip 2	Neighbor 1	Neighbor 2
	JP_ID: 01200001	JP_ID: 01200001
	Δt: 0.00653910636902	Δt: 0.00753092765808
	DTW: 0.0	DTW: 2.79253834528

Neighbor 3	Neighbor 4	Neighbor 5
JP_ID: 01200001	JP_ID: 01200001	JP_ID: 01200001
Δt: 0.0106830596924	Δt: 0.00746893882751	Δt: 0.0113458633423
DTW: 3.37155283205	DTW: 3.38822974421	DTW: 3.47847778707

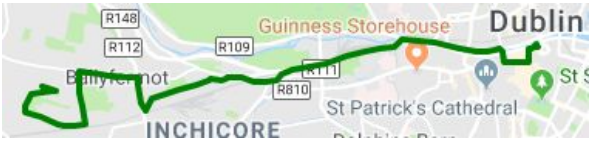
Neighbor 6 - JP_ID: 01200001 - Δt: 0.00779390335083- DTW: 3.49236225658

Test Trip 3

Ο συνολικός χρόνος που απαιτήθηκε είναι 46.7280359268 sec.

Test Trip 3	Neighbor 1
	JP_ID: 00791001
	Δt: 0.0141639709473
	DTW: 0.0

Neighbor 2	Neighbor 3
JP_ID: 00791001	JP_ID: 00791001
Δt: 0.0302209854126	Δt: 0.0249180793762
DTW: 4.69619616101	DTW: 4.76790562543

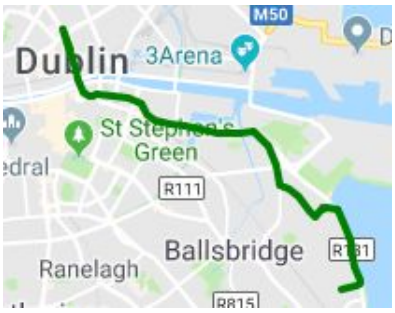


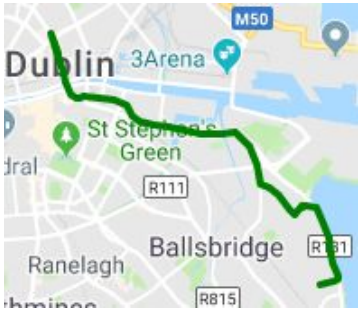
Neighbor 4	Neighbor 5
JP_ID: 00791001	JP_ID: 00791001
Δt: 0.0159640312195	Δt: 0.0209858417511
DTW: 4.80136467351	DTW: 4.86325644338

Neighbor 6 - JP_ID: 00791001 - Δt: 0.0189390182495 - DTW: 4.93316407793

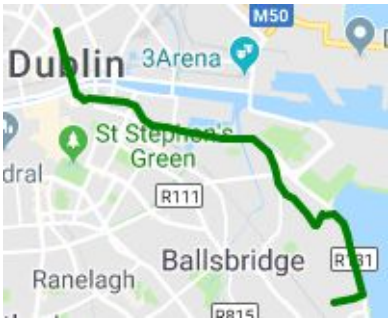
Test Trip 4

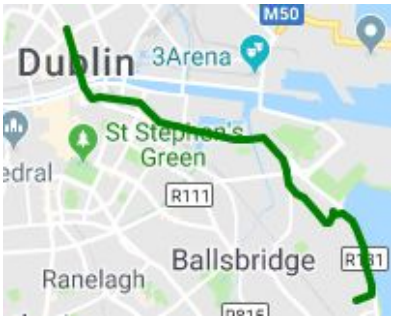
Ο συνολικός χρόνος που απαιτήθηκε είναι 40.1331870556 sec.

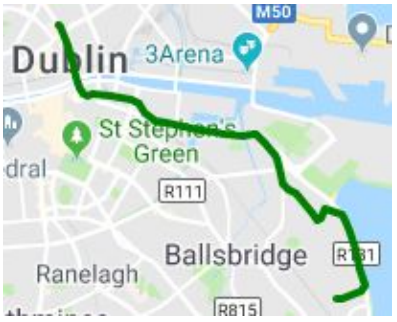




Test Trip 4	Neighbor 1	Neighbor 2
	JP_ID: 00010002	JP_ID: 00010002
	Δt: 0.00834608078003	Δt: 0.0090229511261
	DTW: 0.0	DTW: 2.45555901977





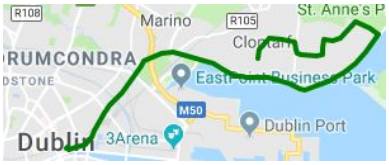


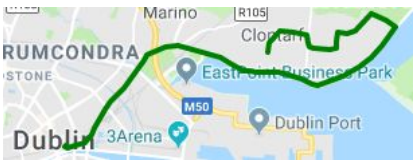
Neighbor 3	Neighbor 4	Neighbor 5
JP_ID: 00010002	JP_ID: 00010002	JP_ID: 00010002
Δt: 0.0116798877716	Δt: 0.0113418102264	Δt: 0.00945520401001
DTW: 2.84590172819	DTW: 3.21060329842	DTW: 3.45813089732

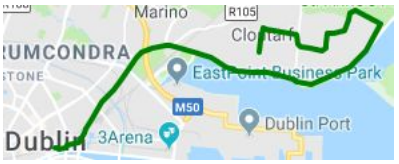
Neighbor 6 - JP_ID: 00010002 - Δt: 0.00995302200317 - DTW: 3.46167920751

Test Trip 5

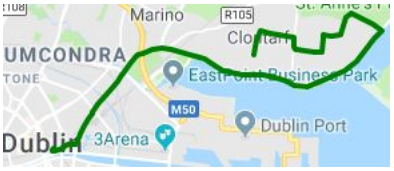
Ο συνολικός χρόνος που απαιτήθηκε είναι 39.0480968952 sec.

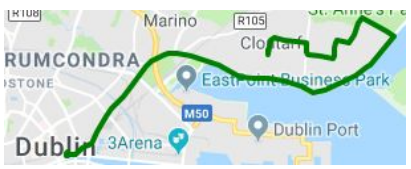


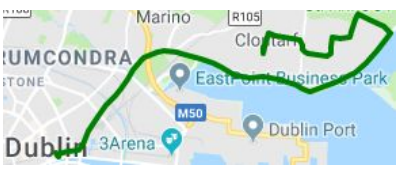




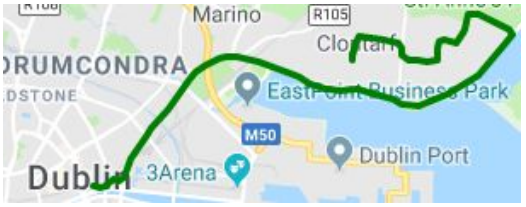
Test Trip 5	Neighbor 1	Neighbor 2
	JP_ID: 01300001	JP_ID: 01300001
	Δt: 0.0184850692749	Δt: 0.0143210887909
	DTW: 0.0	DTW: 4.31391065409







Neighbor 3	Neighbor 4	Neighbor 5
JP_ID: 01300001	JP_ID: 01300001	JP_ID: 01300001
Δt: 0.0171201229095	Δt: 0.00959992408752	Δt: 0.0105841159821
DTW: 4.46726121545	DTW: 4.6400373459	DTW: 4.6903051312



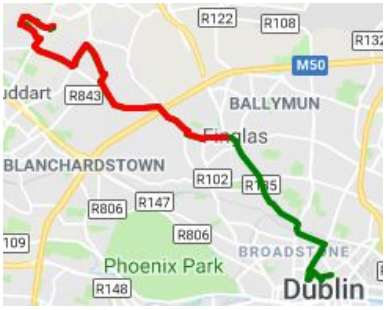
Neighbor 6 - JP_ID: 01300001 - Δt: 0.0104789733887 - DTW: 4.73263659722

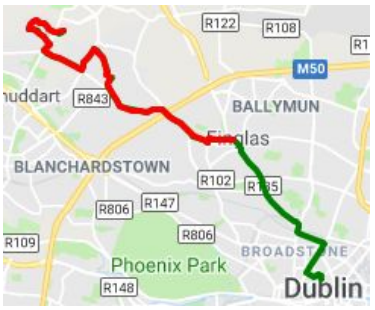
Εύρεση Κοντινότερων Υποδιαδρομών

Σε αυτό το μέρος της εργασίας καλούμαστε να βρούμε τους πέντε γείτονες για κάθε μία από τις πέντε διαδρομές του αρχείου test_set_a2.csv των οποίων οι κοινές υπακολουθίες είναι μέγιστες. Για να γίνει αυτό, υλοποιείται ο αλγόριθμος Longest Common Sub Sequence (LCSS) με συνάρτηση απόστασης ξανά την haversine. Ο αλγόριθμος έχει υλοποιηθεί με δυναμικό προγραμματισμό και χτίζοντας ένα διδιάστατο πίνακα, από τον οποίο στη συνέχεια βρίσκουμε τη μέγιστη κοινή υπακολουθία μεταξύ δύο διαδρομών. Για τις ανάγκες οπτικοποίησης αυτού του ερωτήματος έχει υλοποιηθεί μία ακόμα συνάρτηση plot στην οποία καλείται το gmplot χρωματίζοντας με πράσινο ολόκληρη τη διαδρομή του γείτονα και με κόκκινο χρώμα τα σημεία της κοινής υπακολουθίας με το αρχείο τεστ. Τα αρχεία που παράγονται για την οπτικοποίηση αποθηκεύονται στο φάκελο plots/lcss/i, όπου i είναι ο αριθμός της test διαδρομής ξεκινώντας από το 0. Στο φάκελο αυτό, η test διαδρομή έχει όνομα “lcss_route_i.html”, και οι γείτονές της “lcss_neighbour_j_for_route_i.html”. Σημειώνεται πως για λόγους ομοιομορφίας με τον DTW και σε αυτόν τον αλγόριθμο εμφανίζουμε τους 6 γείτονες.

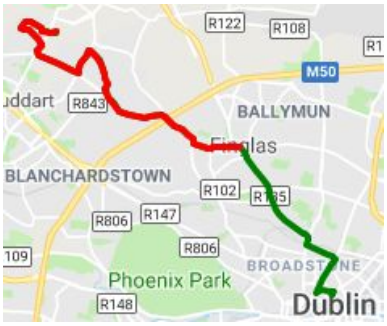
Test Trip 1

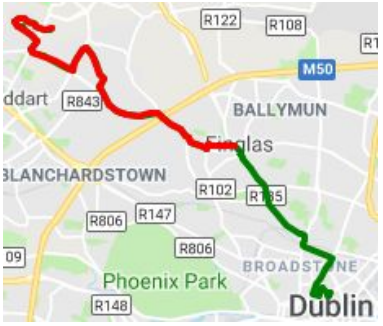
Ο συνολικός χρόνος που απαιτήθηκε είναι 97.1797120571 sec.



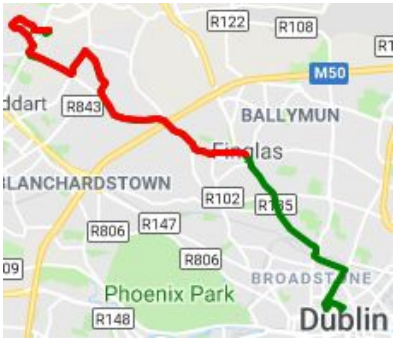


Test Trip 1	Neighbor 1	Neighbor 2
	JP_ID: 040D1002	JP_ID: 040D1002
	Δt: 0.06287693977355957	Δt: 0.051941871643066406
	Matching Points: 78	Matching Points: 78





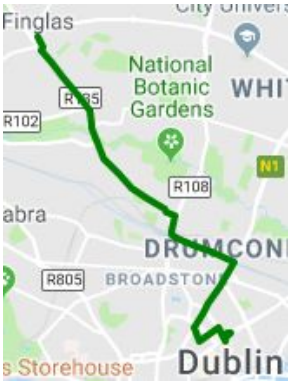
Neighbor 3	Neighbor 4	Neighbor 5
JP_ID: 040D1002	JP_ID: 040D1002	JP_ID: 040D1002
Δt: 0.04571890830993652	Δt: 0.042829036712646484	Δt: 0.04944109916687012
Matching Points: 76	Matching Points: 76	Matching Points: 75



Neighbor 6 - JP_ID: 040D1002 - Δt: 0.04593396186828613 Matching Points: 73

Test Trip 2

Ο συνολικός χρόνος που απαιτήθηκε είναι 94.8388090134 sec.



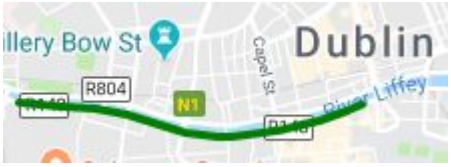
Test Trip 2	Neighbor 1	Neighbor 2
	JP_ID: 040D1002	JP_ID: 040D1002
	Δt: 0.06099200248718262	Δt: 0.05045509338378906
	Matching Points: 82	Matching Points: 78

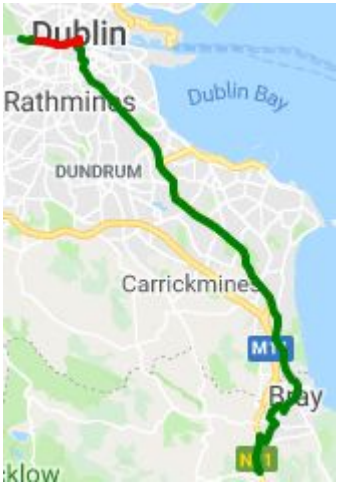
Neighbor 3	Neighbor 4	Neighbor 5
JP_ID: 040D1002	JP_ID: 040D1002	JP_ID: 040D1002
Δt: 0.05040597915649414	Δt: 0.04040098190307617	Δt: 0.04716014862060547
Matching Points: 75	Matching Points: 74	Matching Points: 73

Neighbor 6 - JP_ID: 040D1002 - Δt: 0.05058789253234863 Matching Points: 73

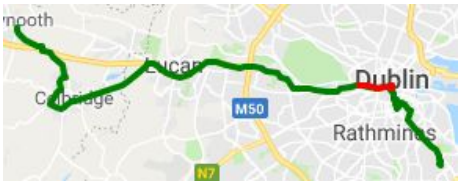
Test Trip 3

Ο συνολικός χρόνος που απαιτήθηκε είναι 51.4915921688 sec.





Test Trip 3	Neighbor 1	Neighbor 2
	JP_ID: 01451001	JP_ID: 01451001
	Δt: 0.038316965103149414	Δt: 0.039628028869628906
	Matching Points: 40	Matching Points: 40



Neighbor 3	Neighbor 4	Neighbor 5
JP_ID: 067X0001	JP_ID: 01451001	JP_ID: 01451008
Δt: 0.04439520835876465	Δt: 0.03886699676513672	Δt: 0.016983985900878906
Matching Points: 40	Matching Points: 40	Matching Points: 40

Neighbor 6 - JP_ID: 00790001 - Δt: 0.015681028366088867 Matching Points: 40

Test Trip 4

Ο συνολικός χρόνος που απαιτήθηκε είναι 70.0899960995 sec.

Test Trip 4	Neighbor 1
	JP_ID: 00790001
	Δt: 0.022548913955688477
	Matching Points: 59

Neighbor 2	Neighbor 3
JP_ID: 00790001	JP_ID: 00790001
Δt: 0.023218154907226562	Δt: 0.02439403533935547
Matching Points: 59	Matching Points: 59

Neighbor 4	Neighbor 5
JP_ID: 00790001	JP_ID: 00790001
Δt: 0.024245023727416992	Δt: 0.024641990661621094
Matching Points: 59	Matching Points: 59

Neighbor 6 - JP_ID: 00790001 - Δt: 0.023375988006591797 Matching Points: 59

Test Trip 5

Ο συνολικός χρόνος που απαιτήθηκε είναι 84.5711319447 sec.

Test Trip 5	Neighbor 1	Neighbor 2
	JP_ID: 01200001	JP_ID: 01200001
	Δt: 0.017630815505981445	Δt: 0.02554011344909668
	Matching Points: 73	Matching Points: 73

Neighbor 3	Neighbor 4	Neighbor 5
JP_ID: 01200001	JP_ID: 01200001	JP_ID: 01200001
Δt: 0.04794716835021973	Δt: 0.023923158645629883	Δt: 0.02017807960510254
Matching Points: 72	Matching Points: 71	Matching Points: 71

Neighbor 6 - JP_ID: 01200001 - Δt: 0.024174928665161133 Matching Points: 71

Διαφορές μεταξύ του αλγόριθμου LCSS και του Dynamic Time Warping

Στην ανάλυση χρονοσειρών, η Δυναμική Χρονική Στρέβλωση (Dynamic Time Warping-DTW) είναι ένας αλγόριθμος για τη μέτρηση της ομοιότητας μεταξύ δύο χρονικών σειρών που μπορεί να διαφέρουν στο χρόνο ή την ταχύτητα. Σε γενικές γραμμές, ο DTW είναι μία μέθοδος που υπολογίζει τη βέλτιστη αντιστοιχία μεταξύ δύο δεδομένων ακολουθιών (π.χ. χρονοσειρές) με ορισμένους περιορισμούς. Οι ακολουθίες
«στρεβλώνονται» μη γραμμικά στη διάσταση του χρόνου για τη μέτρηση της ομοιότητάς τους, ανεξάρτητης από ορισμένες μη γραμμικές παραλλαγές στη χρονική διάσταση.
Το πρόβλημα της Μέγιστης Κοινής Υπακολουθίας (Longest Common Subsequence-LCS) αφορά στην εύρεση της μέγιστης υπακολουθίας, κοινής σε όλες τις ακολουθίες ενός σετ ακολουθιών (συχνά 2).
Η βασική διαφορά των δύο αλγορίθμων έγκειται στο γεγονός ότι στο DTW όλα τα στοιχεία αντιστοιχίζονται στις δύο ακολουθίες και με αυτόν τον τρόπο επιτυγχάνεται μία ένα-προς-ένα χαρτογράφηση. Το αποτέλεσμα αυτής της συμπεριφοράς του αλγορίθμου είναι ότι υπάρχει κίνδυνος οι ακραίες τιμές να στρεβλώσουν τη μέτρηση της απόστασης, δηλαδή το τελικό αποτέλεσμα. Σε αντίθεση, σύμφωνα με τον αλγόριθμο LCSS, οι απομακρυσμένες τιμές δεν αντιστοιχίζονται στην άλλη ακολουθία (επιτρέπονται τα κενά) και με αυτόν τον τρόπο υπάρχει μικρότερη παραμόρφωση της απόστασης/ομοιότητας εξαιτίας του θορύβου.
Συμπεραίνουμε ότι ο αλγόριθμος LCSS είναι προτιμότερος και αποδοτικότερος σε δεδομένα που περιέχουν αρκετό θόρυβο, καθώς είναι σε θέση να τον αγνοήσει και να παράγει αποτέλεσμα σύμφωνα με τα πραγματικά ωφέλιμα δεδομένα. Αντιθέτως, στην περίπτωση που τα δεδομένα είναι στην πραγματικότητα αυτά που εξετάζει ο αλγόριθμος και δεν έχει εισχωρήσει θόρυβος, ο αλγόριθμος DTW είναι ο καταλληλότερος.
Ως προς τη χρονική πολυπλοκότητα, στην περίπτωση σύγκρισης δύο ακολουθιών (τροχιές σύγκρισης), και οι δύο αλγόριθμοι χρειάζονται O(n*m), όπου n και m τα μήκη των δύο ακολουθιών αντίστοιχα. Αξίζει να σημειωθεί πως υπάρχουν βελτιστοποιήσεις του LCSS τόσο για καλύτερους χρόνους, όσο και για αποδοτικότερη διαχείριση μνήμης.

Εξαγωγή των Features για Κατηγοριοποίηση

Σε αυτό το ερώτημα ζητάται η αναπαράσταση των διαδρομών σε ένα διδιάστατο πίνακα και κατά συνέπεια σαν ένα σύνολο από κελιά. Για να πραγματοποιηθεί αυτή η μετατροπή, απαιτείται η αντιστοίχιση του γεωγραφικού μήκους και γεωγραφικού πλάτους κάθε σημείου κάθε διαδρομής σε έναν ακέραιο αριθμό, ο οποίος θα είναι το μέγεθος της μίας διάστασης του πίνακα. Με τη χρήση του MinMaxScaler, πραγματοποιείται η αντιστοίχιση των συντεταγμένων σε ακεραίους. Για την αναπαράσταση των σημείων σε κελιά, ελέγχουμε κάθε σημείο σε ποιό κελί του πίνακα αντιστοιχεί και σε αυτή τη θέση αυξάνουμε ένα μετρητή κατά ένα κάθε φορά που ένα σημείο “πέφτει” μέσα στο κελί αυτό. Ο λόγος που αυξάνεται ο μετρητής και δεν εισάγεται απλά το 1 για αυτό το κελί, είναι πως δίνουμε κάποιο βάρος στα κελιά στα οποία η διαδρομή δεν έχει μόνο ένα σημείο, αλλά περισσότερα. Για να αποφασιστεί το μέγεθος του πίνακα έγιναν διάφορα πειράματα, συναρτήσει των κατηγοριοποιητών και καταλήξαμε ως βέλτιστο μέγεθος το 20x20.

Κατηγοριοποίηση

Για την κατηγοριοποίηση των δεδομένων χρησιμοποιούνται τα νέα δεδομένα που παράχθηκαν στο προηγούμενο ερώτημα ως είσοδος στους classifiers, προκειμένου να βρεθούν τα JournayId των trips. Η αξιολόγηση των αποτελεσμάτων γίνεται με 10-fold Cross Validation, μία τεχνική η οποία χωρίζει τα δεδομένα σε 10 κομμάτια με όλους τους πιθανούς τρόπους, χρησιμοποιώντας τα εννέα από αυτά για να εκπαιδεύσει τους κατηγοριοποιητές και το δέκατο για να προβλέψει το ζητούμενο.

Οι αλγόριθμοι που ζητούνται είναι ο K-nearest Neighbors, ο Logistic Regression και ο Random Forest.

Ο KNN αλγόριθμος αναζητά τους κ κοντινότερους γείτονες και είναι ένας lazy learning αλγόριθμος, που καθυστερεί τη διαδικασία μοντελοποίησης του συνόλου εκπαίδευση μέχρι να γίνει απαραίτητο. Η πολυπλοκότητά του είναι πολυωνυμική και αυτό φαίνεται από το γεγονός ότι είναι εμφανώς ταχύτερος από τους άλλους δύο. Σαν κ δοκιμάστηκαν διάφορες τιμές, όπως για παράδειγμα κ=4 και κ=5, οι οποίες ωστόσο έδιναν χειρότερα αποτελέσματα από το κ=1.

Ο αλγόριθμος logistic regression είναι ο πιο αργός από τους 3, γιατί κάνει οπισθοδρόμηση προσπαθώντας να κατηγοριοποιήσει τα δεδομένα. Είναι ικανός να διαχειριστεί αραία αλλά και πυκνά δεδομένα και ουσιαστικά είναι ένας αλγόριθμος που έχει τις ρίζες του στη στατιστική, αφού μοντελοποιείται με πιθανότητες. Στην παράμετρο C, η οποία ρυθμίζει το πρόβλημα του overfitting, έγιναν διάφορες δοκιμές και καταλήξαμε σε 0.5.

Τέλος, ο κατηγοριοποιητής Random Forest χτίζει σταδιακά δέντρα απόφασης κάνοντας μια σειρά από ερωτήσεις στα χαρακτηριστικά. Η τεχνική αυτή είναι αρκετά ακριβή γιατί εξαρτάται από το πλήθος των δεδομένων, τον αριθμό τον χαρακτηριστικών και το σχήμα του δέντρου, έχοντας λογαριθμική πολυπλοκότητα. Σε αυτήν την τεχνική, η οποία αποδείχθηκε και η πιο αποδοτική για τα δεδομένα μας, έγιναν οι περισσότερες δοκιμές στην είσοδο. Συγκεκριμένα εξετάστηκε το μέγιστο βάθος (π.χ. max_depth=50), ο αριθμός των δέντρων (π.χ. n_estimators=100) και ο αριθμός των χαρακτηριστικών για το διαχωρισμό και την κατασκευή υποδέντρου (π.χ. max_features=25). Τα νούμερα που προτιμήθηκαν προέκυψαν από

πειραματική μελέτη και σε συνάρτηση με το μέγεθος του διδιάστατου πίνακα των δεδομένων.

Παρακάτω φαίνονται τα αποτελέσματα για διάφορες δοκιμές των κατηγοριοποιητών αναλόγως το μέγεθος του πίνακα.

Για μέγεθος πίνακα 30x31 (ο πίνακας χτιζόταν με διαφορετικό τρόπο):
k-Nearest Neighbor - Accuracy: 0.79 (+/- 0.17)
Logistic Regression - Accuracy: 0.85 (+/- 0.15)
Random forest - Accuracy: 0.86 (+/- 0.14)

Για μέγεθος πίνακα 20*20
k-Nearest Neighbor - Accuracy: 0.80 (+/- 0.17)
Logistic Regression - Accuracy: 0.86 (+/- 0.15)
Random forest - Accuracy: 0.87 (+/- 0.14)

Beat The Benchmark

Εφόσον ο αλγόριθμος random forest είχε τα καλύτερα αποτελέσματα, προτιμήθηκε για αυτό το μέρος της εργασίας. Παρόλα αυτά, αξίζει να σημειωθεί πως δοκιμάστηκαν ακόμα δύο κατηγοριοποιητές, ο linear SVM και ο rbf SVM.

Για μέγεθος πίνακα 30x31
linear-SVM - Accuracy: 0.83 (+/- 0.16)
rbf-SVM - Accuracy: 0.77 (+/- 0.17)

Ένα πρόβλημα που εντοπίστηκε, ήταν η αδυναμία να εκφράσουμε στα δεδομένα κατεύθυνση, με αποτέλεσμα διαδρομές που ήταν ανάποδες να θεωρούνται ίδιες. Για το λόγο αυτό πραγματοποιήθηκε επιπλέον προεπεξεργασία των δεδομένων όταν εξάγονται στα κελιά. Συγκεκριμένα, αντί να υπολογίζεται μόνο το βάρος για ένα κελί που έχει πολλαπλά σημεία μιας διαδρομής, υπολογίζεται και βάρος ανάλογα με τις συντεταγμένες του επόμενου σημείου της διαδρομής. Τα βάρη αυτά είναι 8 διαφορετικές τιμές, από 0 έως 8, και υπολογίζονται όπως ένα αναλογικό ρολόι. Εάν το επόμενο σημείο της διαδρομής βρίσκεται βορειοανατολικά του τρέχοντος (επάνω και δεξιά δηλαδή), το κελί θα έχει επιπλέον βάρος 1, ένα είναι κάτω νοτιοδυτικά, το κελί θα έχει επιπλέον βάρος 5 κ.ο.κ. Αξίζει να σημειωθεί πως η παραπάνω απλοποιημένη λογική, έγινε προσπάθεια να γίνει με ngrams παίρνοντας δηλαδή τα σημεία ανά δυάδες για παράδειγμα, αλλά αντιμετωπίσαμε διάφορα προβλήματα στον τρόπο που τα δεδομένα εισάγονται στους κατηγοριοποιητές.
Τέλος, υπήρξε και η ιδέα του υπολογισμού του μεγέθους του πίνακα, πέρα από τα πειράματα, με βάση τη μέγιστη και την ελάχιστη συντεταγμένη των διαδρομών, ωστόσο θεωρήθηκε πως κάτι τόσο στοχευμένο και χειροκίνητο δε θα ήταν σωστή αντιμετώπιση. Επιπλέον έγιναν και κάποιες δοκιμές για μείωση των διαστάσεων με τη χρήση svd, χωρίς κάποια βελτίωση στα αποτελέσματα.

Εκτελέσεις των classifiers μετά την επεξεργασία με τα νέα βάρη φαίνονται παρακάτω:

Για μέγεθος πίνακα 16x16
k-Nearest Neighbor - Accuracy: 0.87 (+/- 0.13)
Logistic Regression - Accuracy: 0.87 (+/- 0.14)
Random forest - Accuracy: 0.90 (+/- 0.14)
linear-SVM - Accuracy: 0.88 (+/- 0.13)
rbf-SVM - Accuracy: 0.45 (+/- 0.18)

Για μέγεθος πίνακα 20x20
k-Nearest Neighbor - Accuracy: 0.87 (+/- 0.13)
Logistic Regression - Accuracy: 0.88 (+/- 0.14)
Random forest - Accuracy: 0.90 (+/- 0.13)

Σημείωση: Σε διάφορα σημεία του παραδοτέου κώδικα υπάρχει πολυνηματικός κώδικας με τη χρήση pool για ταχύτερη εκτέλεση των αλγορίθμων.