

ΤΕΧΝΙΚΕΣ ΕΞΟΡΥΞΗΣ ΔΕΔΟΜΕΝΩΝ

ΕΡΓΑΣΙΑ 2

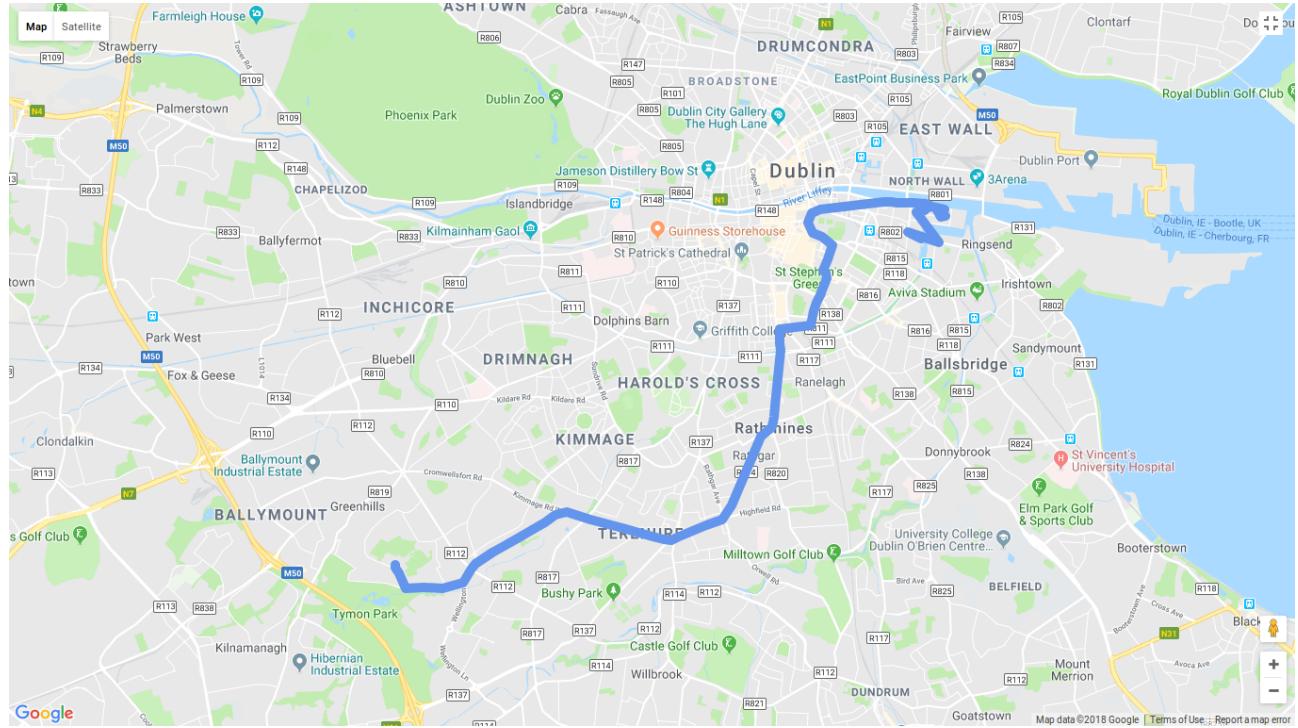
ΕΑΡΙΝΟ 2017-2018

**ΚΑΤΣΟΡΙΔΑΣ ΙΩΑΝΝΗΣ – 1115201400066
ΠΑΠΑΝΑΣΤΑΣΙΟΥ ΛΕΩΝΙΔΑΣ – 1115201500121**

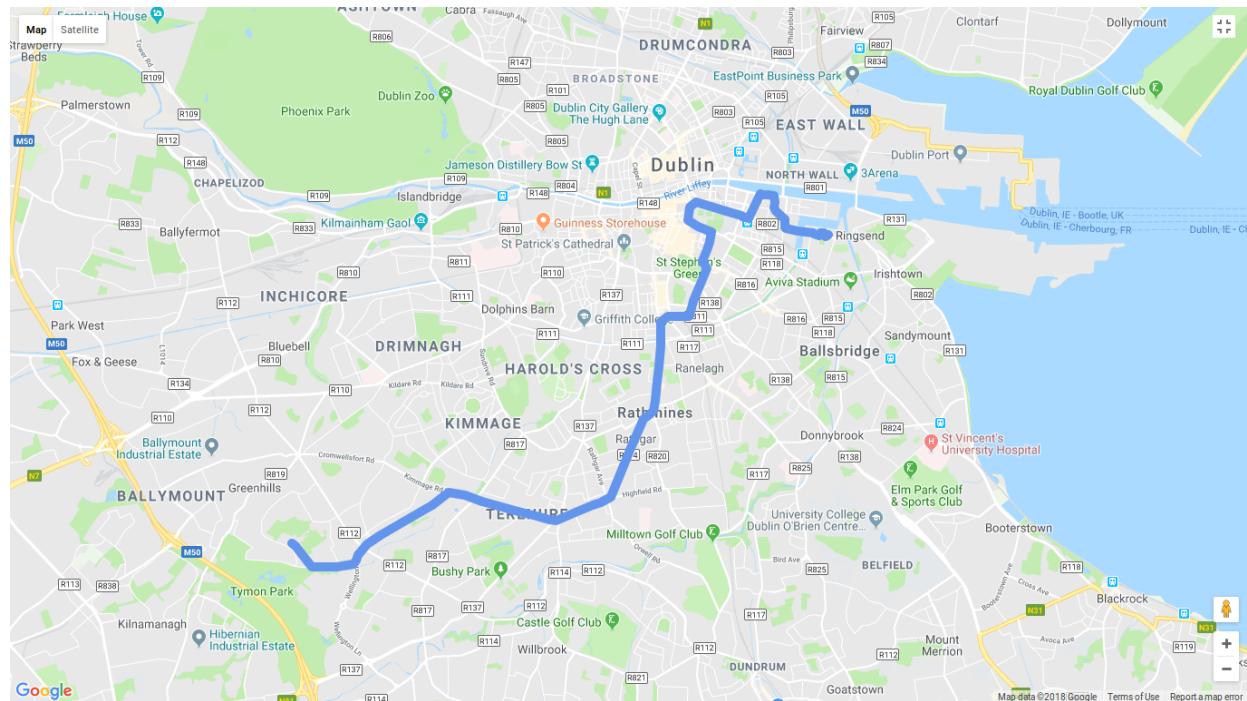
ΕΡΩΤΗΜΑ 1

Μέσω της βιβλιοθήκης gmapplot, σχεδιάστηκαν (στη τύχη) οι εξής διαδρομές:

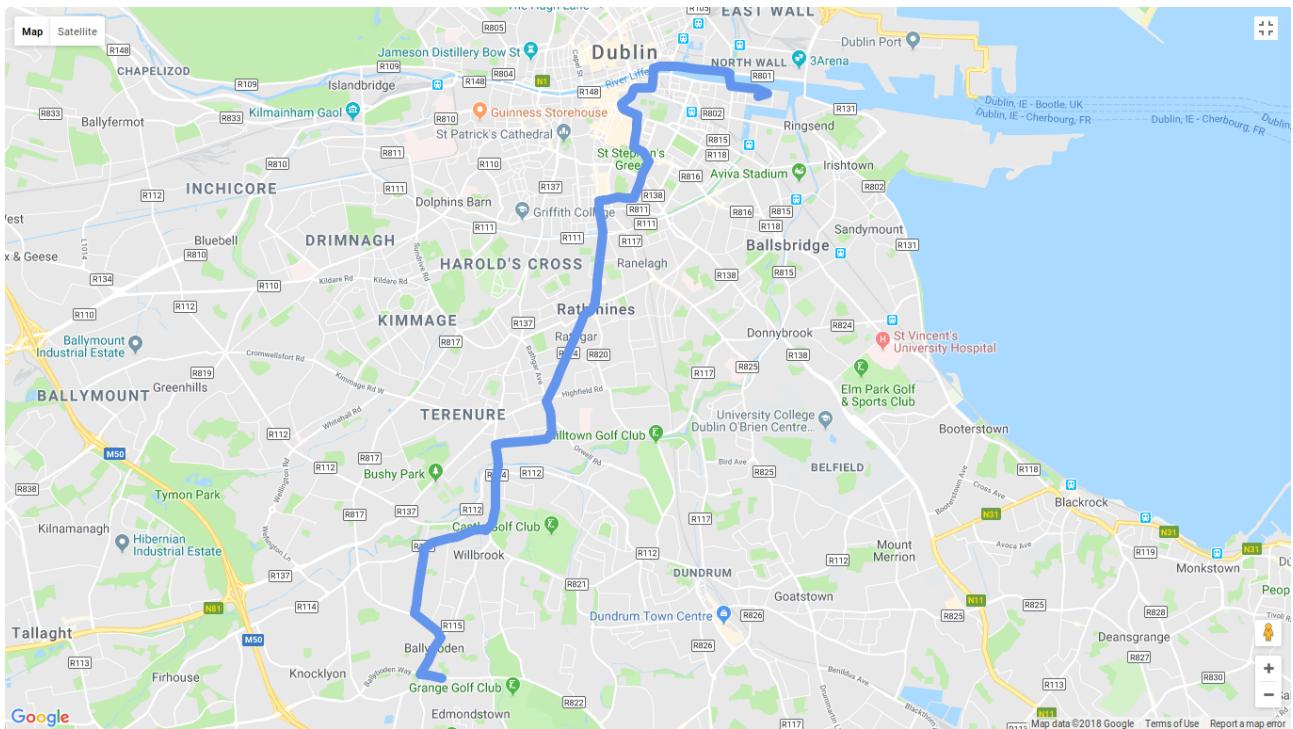
- 015A0001 (Trip ID: 10)



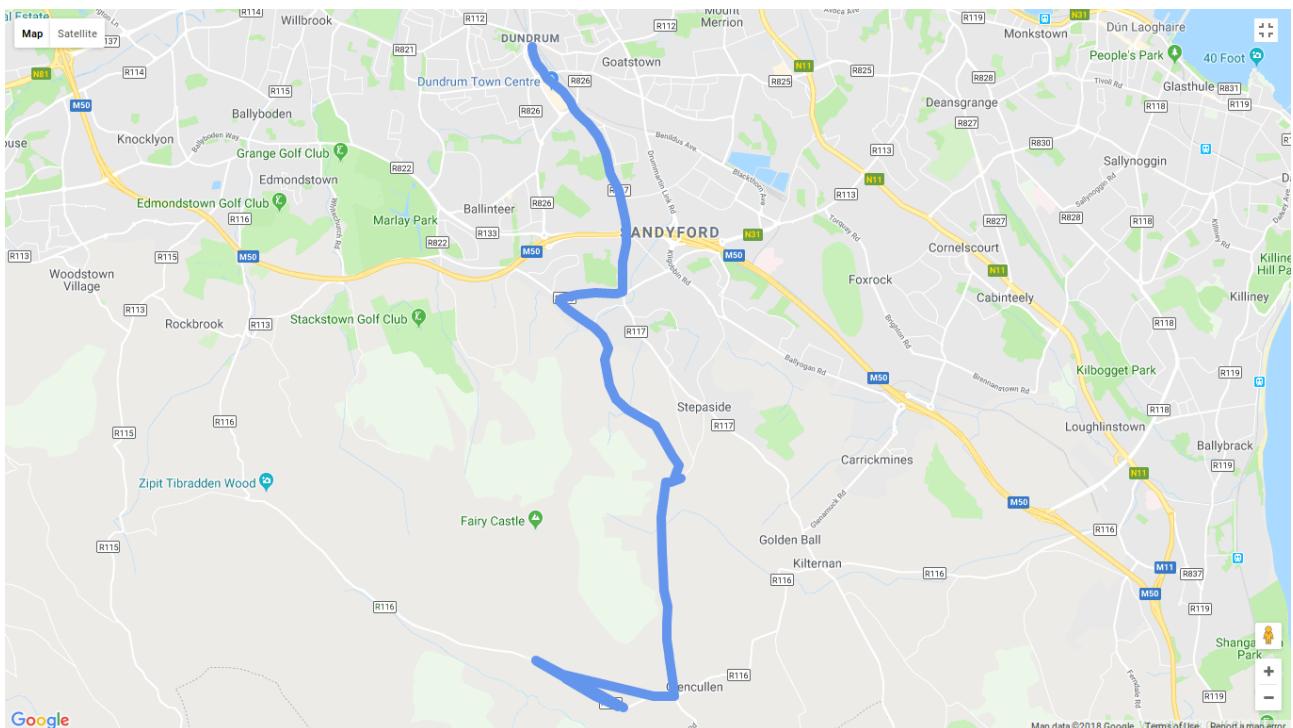
- 015A0002 (Trip ID: 20)



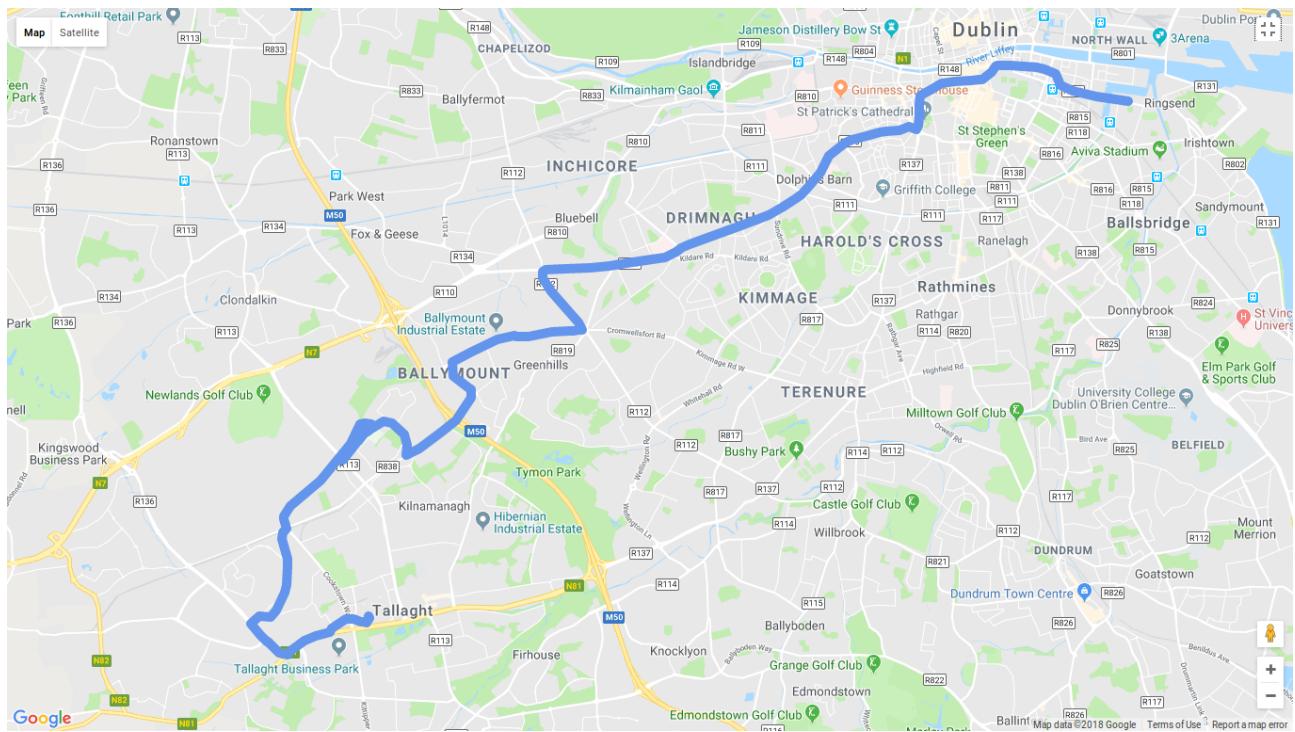
- 015B1002 (Trip ID: 30)



- 044B1001 (Trip ID: 45)



- 056A1001 (Trip ID: 50)



Οι φωτογραφίες βρίσκονται στο φάκελο Q1 - Plots που συνοδεύει το παραδοτέο.

ΕΡΩΤΗΜΑ 2Α

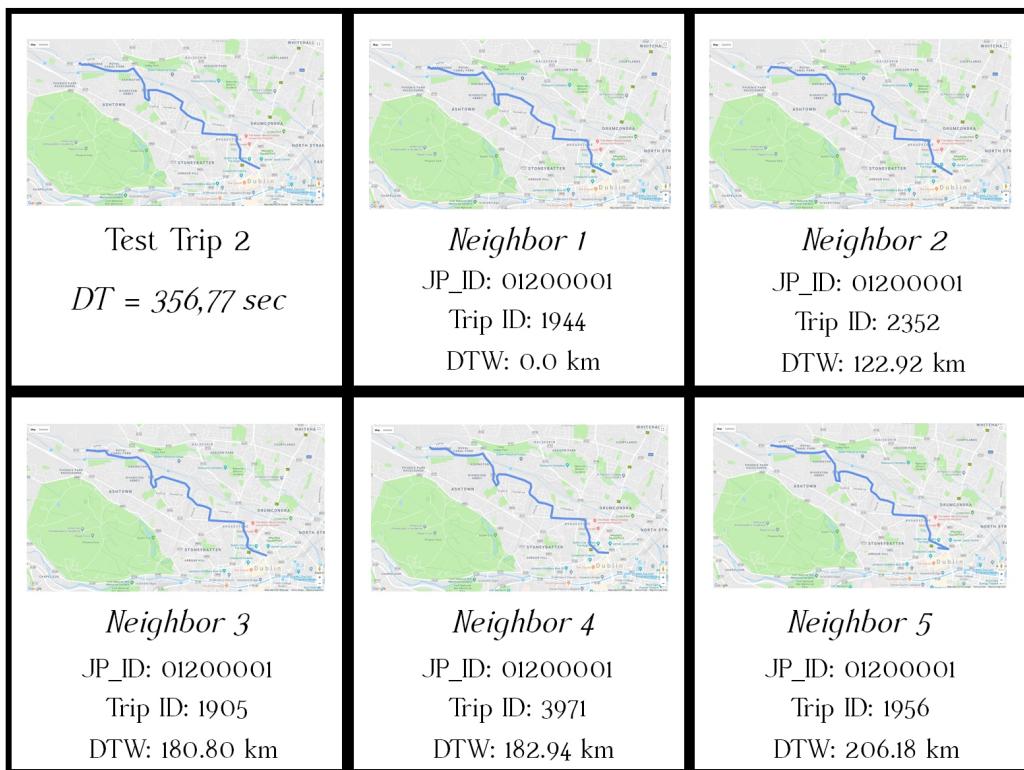
Στο ερώτημα αυτό, όπως ζητείται από την εκφώνηση, υλοποιήθηκε μια έκδοση του K-Nearest Neighbor με συνάρτηση απόστασης, τον αλγόριθμο Dynamic Time Warping. Η υλοποίηση τόσο του Dynamic Time Warping, όσο και της ειδικής έκδοσης του KNN, έγινε εξ' ολοκλήρου από εμάς. Για τη συνάρτηση απόστασης Haversine, χρησιμοποιήθηκε η έτοιμη βιβλιοθήκη της PyPi. Η υλοποίηση του DTW βρίσκεται στο αρχείο DTW.py, του KNN στο KNN.py ενώ ο κώδικας που τις καλεί βρίσκεται στο αρχείο DTWClassification.py. Οι φωτογραφίες, όπως επίσης και τα html αρχεία που περιέχουν τις διαδρομές, βρίσκονται μέσα στο φάκελο DTW. Το αρχείο που δημιουργεί τα plots, είναι το DTWPlots.py. Για κάθε test case, έχει δοθεί και το αποτέλεσμα του DTWClassification, που δείχνει ουσιαστικά τους γείτονες, την απόσταση αλλά και το JP_ID.

*Για το διάβασμα των testSet τόσο αυτού όσο και του επόμενου υποερωτήματος χρησιμοποιήθηκε το πρόγραμμα testSetTransformer.py που το μετατρέπει σε κατάλληλη μορφή. (Υλοποίηση από εμάς)

Query 1:



Query 2:



Query 3:



Query 4:



Query 5:



ΕΡΩΤΗΜΑ 2Β

Για το ερώτημα αυτό ακολουθήθηκε παρόμοια λογική με το 2Α. Τροποποιήθηκε, ελαφρώς, ο αλγόριθμος εύρεσης των κοντινότερων γειτόνων για τους σκοπούς αυτού του ερωτήματος και ο νέος αλγόριθμος βρίσκεται στο αρχείο KNNLCSS.py. Υλοποιήθηκε, από εμάς, ο αλγόριθμος Longest Common SubSequence, οποίος βρίσκεται στο αρχείο LCSS.py και με βάση αυτόν τον αλγόριθμο βρέθηκαν οι κοντινότεροι γείτονες, χρησιμοποιώντας το αρχείο LCSSClassification.py το οποίο είναι παρόμοιο με αυτό που χρησιμοποιήθηκε στο υπερώτημα Α, με μικρή διαφορά πως εκτυπώνει και τις κοινές υπακολουθίες. Έπειτα, με τη χρήση του LCSSPlots.py δημιουργούνται τα ζητούμενα αρχεία. Οι φωτογραφίες, όπως και τα plots, βρίσκονται σε υψηλή ανάλυση στο φάκελο LCSS. Όπως και στον DTW, συνοδεύεται με κάθε test case τα αποτελέσματα του LCSSClassification.

Query 1:



Query 2:



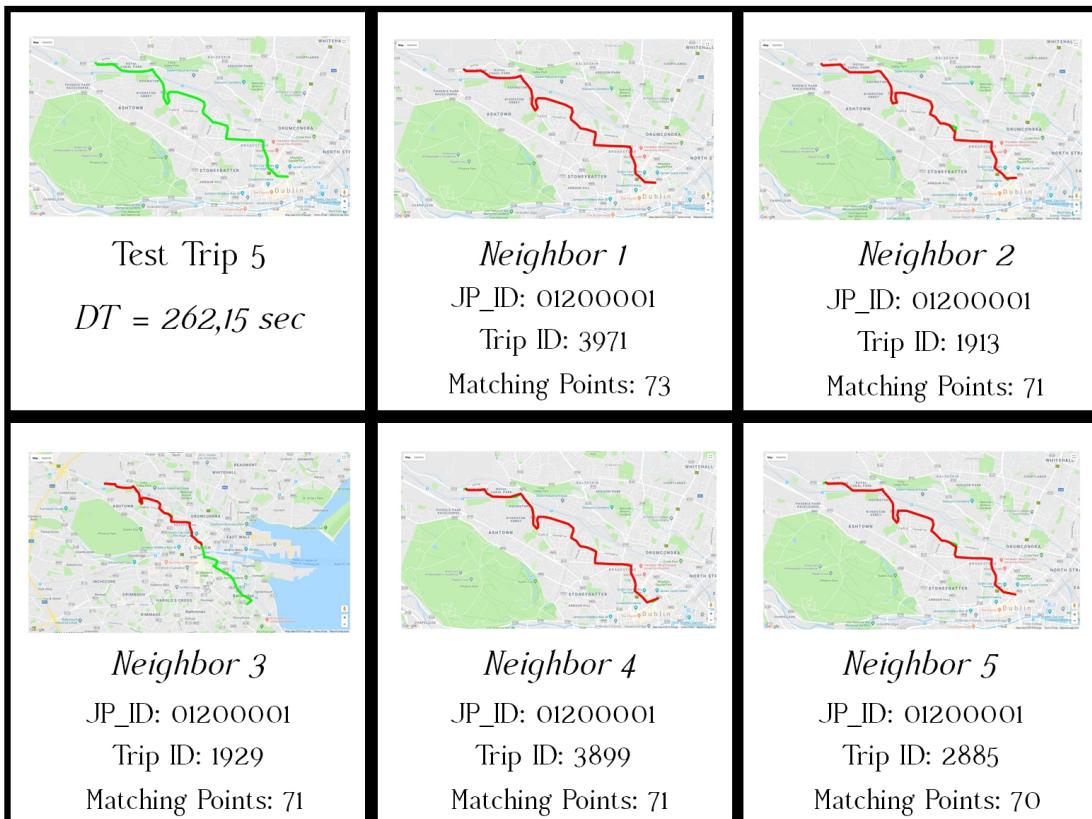
Query 3:



Query 4:



Query 5:



ΕΡΩΤΗΜΑ 3

10 Fold Cross Validation:

Για την αξιολόγηση του μοντέλου χρησιμοποιήθηκε η μέθοδος του 10 fold Cross Validation. Χρησιμοποιήθηκε η συνάρτηση cross_val_predict του sklearn που έχει ακριβώς αυτή τη λειτουργία. Για να χρησιμοποιηθεί, όμως, η συγκεκριμένη συνάρτηση, έπρεπε να μετατραπεί ο KNN σε κατάλληλη μορφή. Χρησιμοποιήθηκε μια παραλλαγή του KNN που είχαμε φτιάξει για τη προηγούμενη άσκηση. Αυτή η έκδοση του KNN είναι αρκετά πιο χρονοβόρα καθώς δε κρατάει δυναμικά τις αποστάσεις, ώστε να βρει άμεσα τους κοντινότερους γείτονες, αλλά τις ξαναπολογίζει. Με δεδομένη τη μεγάλη καθυστέρηση, αποφασίστηκε να τρέξει το CV σε ένα δείγμα 4% του αρχικού, δηλαδή σε 262 διαδρομές. Αυτές οι διαδρομές επιλέχθηκαν στη τύχη. Το αποτέλεσμα που έβγαλε η μέθοδος είναι 38,9% ακρίβεια (έτρεξε σε περίπου 13 ώρες). Αυτό το αποτέλεσμα, βέβαια, δε μπορεί να είναι αντιπροσωπευτικό, καθώς για τον υπολογισμό του χρησιμοποιήθηκε ένα πολύ μικρό ποσοστό του δείγματος ενώ παράλληλα υπάρχει και το θέμα της τυχαιότητας.

Classification:

Για τη κατηγοριοποίηση των διαδρομών, χρησιμοποιήθηκε ο KNN που χρησιμοποιήθηκε και στο 2A όπως επίσης και το αντίστοιχο αρχείο, με μικρή παραλλαγή το ταυτόχρονο υπολογισμό όλων των διαδρομών και το γράψιμό τους στο ζητούμενο csv. Ο κώδικας βρίσκεται στο αρχείο Q3Classification.py ενώ το csv δίνεται με τη ζητούμενο όνομα.