2η ΕΡΓΑΣΙΑ

ΤΕΧΝΙΚΕΣ ΕΞΟΡΥΞΗΣ ΔΕΔΟΜΕΝΩΝ

ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2017-2018

ΜΕΛΗ:
-ΓΙΑΝΝΗΣ ΓΙΑΝΝΑΚΙΔΗΣ 1115201500025
-ΚΩΝΣΤΑΝΤΙΝΟΣ ΠΑΣΧΟΠΟΥΛΟΣ 1115201500127

1. Οπτικοποίηση των δεδομένων:

Αρχείο: er1.py

Επιλέγουμε 5 διαφορετικές διαδρομές από το αρχείο train_set.csv και τις οπτικοποιούμε με την χρήση της gmplot.Η gmplot είναι μια διεπαφή που παράγει html αρχείο με βάση τα δεδομένα που της παρέχουμε πάνω στους χάρτες τις Google.

2.

(Α-1) Εύρεση κοντινότερων γειτόνων:

Αρχείο: er2_a1.py

Χρησιμοποιώντας το test_set_a1.csv προσπαθούμε να βρούμε τους 5 κοντινότερους γείτονες από το αρχείο train_set.csv χρησιμοποιώντας την τεχνική Dynamic Time Warping(DTW).Οι γεωγραφικές αποστάσεις υπολογίζονται με τον τύπο Haversine.

Η τεχνική DTW είναι ένας αλγόριθμος που υπολογίζει την ομοιότητα μεταξύ 2 χρονικών ακολουθιών με συγκεκριμμένους περιορισμούς.

Για την υλοποίηση της χρησιμοποιήσαμε την εξής πηγή:

https://anaconda.org/bioconda/fastdtw

Η εγκατάσταση αυτού του πακέτου γίνεται με την εντολή:

conda install -c bioconda fastdtw

Ο τύπος Haversine είναι μια formula που καθορίζει την κυκλική απόσταση μεταξύ 2 σημείων σε μια σφαίρα με δεδομένα το γεωγραφικό πλάτος και μήκος των σημείων.

Για την υλοποίηση της χρησιμοποιήσαμε την εξής πηγή:

https://github.com/mapado/haversine

Η εγκατάσταση αυτού του πακέτου γίνεται με την εντολή:

pip install haversine

Η οπτικοποίηση των αποτελεσμάτων στο πρότυπο της εκφώνησης γίνεται εκτός του προγράμματος,χειροκίνητα,απο εμάς.

Χρόνοι εκτέλεσης για κάθε διαδρομή του test_set.csv:

- Test Trip 1: $\Delta t = 167.805989027$ sec
- Test Trip 2: $\Delta t = 141.309678078$ sec
- Test Trip 3: $\Delta t = 165.791587114$ sec
- Test Trip 4: $\Delta t = 144.29736805$ sec
- Test Trip 5: $\Delta t = 142.326406002$ sec

(A-2) Εύρεση κοντινότερων υποδιαδρομών:

Αρχείο: er2_a2.py

Στο ερώτημα αυτό με την χρήση του test_set_a2.csv βρίσκουμε τις 5 κοντινότερες υποδιαδρομές από το αρχείο train_set.csv με χρήση της τεχνικής Longest Common Subsequence (LCSS) με χρήση όπως προηγουμένως της μετρικής Haversine.

Ο αλγόριθμος LCSS βρίσκει την μέγιστη κοινή υπακολουθία κοινών σημειών μεταξύ 2 διαδρομών.Στην συνέχεια κρατάμε για κάθε διαδρομή του test_set_a2.csv τις 5 διαδρομές από το αρχείο train_set.csv με τις 5 μεγαλύτερες κοινές υπακολουθίες.Για την υλοποίηση του αλγορίθμου χρησιμοποιήσαμε την πηγή:

https://en.wikibooks.org/wiki/Algorithm_Implementation/Strings/Longest_common_subsequence

Έγιναν οι κατάλληλες προσαρμογές-αλλαγές στην από πάνω πηγή για να ταιριάζουν με τα δεδομένα μας.

Η υλοποίηση του backtracking για να βρούμε τα κοινά γεωγραφικά μήκη και πλάτη εγίνε από εμάς και όχι με τον τρόπο της πηγής.

Ο τύπος Haversine όπως περιγράφηκε παραπάνω από την ίδια πηγή.

Η οπτικοποίηση των αποτελεσμάτων στο πρότυπο της εκφώνησης γίνεται εκτός του προγράμματος,χειροκίνητα,απο εμάς.

Χρόνοι εκτέλεσης για κάθε διαδρομή του test_set.csv:

- Test trip 1: $\Delta t = 169.725680113$ sec
- Test trip 2: $\Delta t = 169.248766899$ sec
- Test trip 3: $\Delta t = 82.7700078487$ sec
- Test trip 4: $\Delta t = 122.40190196$ sec
- Test trip 5: $\Delta t = 150.84256196$ sec

3. Κατηγοροιοποίηση

Αρχεία: a) er3.py

b)knn_functions.py

Χρησιμοποιώντας το test_set_a2.csv προσπαθούμε να βρούμε τους κοντινότερους γείτονες από το αρχείο train_set.csv χρησιμοποιώντας την μέθοδο k-neirest neighbors με k=5 για την πρόβλεψη των γραμμών των διαδρομών του test_set_a2.csv.

Η υλοποιήση του αλγορίθμου k-nn είναι παρόμοια με της πρώτης εργασίας αλλά με τις κατάλληλες προσαρμογές στα δεδομένα της δεύτερης εργασίας.

Στην συνέχεια καταγράφουμε την απόδοση του μοντέλου μας με την χρήση

της μετρικής Accuracy.

Η υλοποίηση μας λειτουργεί ορθά με ένα μέρος των δεδομένων αλλά για ολόκληρο το train_set.csv έχουμε MemoryError.Για αυτόν τον λόγο χρησιμοποιούμε 20 διαδρομές από το train_set.csv, 1 απο το test_set_a2.csv και κανουμε cross validation με 3 splits στο παραδοτέο για να αποφευχθεί το Error.