

Θέμα 3ο: Machine Learning - Ομαδοποίηση δεδομένων με εκτέλεση του k-means αλγόριθμου

Βήματα:

1. Για την επεξεργασία των δεδομένων έπρεπε να βρεθεί ένα πρακτικό περιβάλλον.

Οπότε κατέληξα στο jupyter notebook το οποίο εγκατέστησα στον master μέσω του anaconda, το οποίο εγκατέστησα στον master και στον slave για να έχω την κατάλληλη έκδοση python που χρειαζόμουν, μιας και η python 2.7 δεν υποστηρίζεται από το jupyter.

Προσοχή!! Για να τρέξουμε τον κώδικα μέσω jupyter notebook χρησιμοποιούμε το αρχείο Kmeans.ipynb και για να τρέξουμε τον κώδικα με την εντολή spark submit χρησιμοποιούμε το αρχείο Kmeans.py , για τον δεύτερο τρόπο πρέπει να εγκαταστήσουμε στον master και slave το findspark μέσω της εντολής conda install -c conda-forge findspark . Το .ipynb και το .py διαφέρουν κατά μια εντολή.

2. Αφού είχα το περιβάλλον έτοιμο , μέσω του master φόρτωσα τα δεδομένα μου στο hdfs και αρχισα την επεξεργασία τους μέσω του jupyter.
3. Επέλεξα το "Θέμα 3ο: Machine Learning - Ομαδοποίηση δεδομένων με εκτέλεση του k-means αλγόριθμου".
4. Βλέπουμε ότι χρειαζόμαστε μόνο το γεωμετρικό μήκος και πλάτος , επομένως φορτώνουμε μόνο το yellow_tripdata_1m.csv
5. Από αυτό το πακέτο δεδομένων χρειαζόμαστε μόνο 2 εγγραφές από κάθε γραμμή δηλαδή το γεωγραφικό μήκος και γεωγραφικό πλάτος , και ταυτόχρονα επιθυμούμε να μην έχουν τιμές που δεν αντιστοιχούν στην Νέα Υόρκη και πιθανώς τοποθετήθηκαν καταλάθος
6. Με έναν Mapper έχουμε σαν είσοδο τα δεδομένα και επιλέγουμε να πάρουμε μόνο τα 2 κελιά που μας ενδιαφέρουν από κάθε γραμμή τα οποία χωρίζονται με "," και ταυτόχρονα πραγματοποιούμε έναν καθαρισμό των δεδομένων μας, για να κρατήσουμε μόνο όσα στοιχεία βρίσκονται στα εξής όρια :

- $40 \leq \text{latitude} \leq 41$
- $-75 \leq \text{longitude} \leq -73$ Τα όρια αυτά είναι προσεγγιστικά για την Νέα Υόρκη.

7. Κανω cache τα δεδομένα μου , έτσι ώστε να αυξηθεί η απόδοση και να έχω άμεση πρόσβαση σε αυτά αργότερα στο for loop.

8. Τα 5 πρώτα τα αναθέτουμε στη μεταβλητή centroids που είναι τα κέντρα μας.

9. Είμαστε έτοιμοι να εκτελέσουμε για 3 επαναλήψεις τον αλγόριθμο k-means.

10. Κάθε επανάληψη έχει 3 βήματα :

- Για κάθε εγγραφή του dataset μας βρίσκουμε σε ποια απο τις 5 τοποθεσίες του centroids είναι πιο κοντά σύμφωνα με την απόσταση Haversine. Η διαδικασία γίνεται με έναν Mapper στον οποίο κάθε είσοδο την στέλνουμε σε μια συνάρτηση που υπολογίζει 5 αποστάσεις Haversine (1 για κάθε κέντρο) και επιστρέφει το index (0-4) που υποδηλώνει σε ποιο κέντρο είναι πιο κοντά. Έτσι με την έξοδο του Mapper έχουμε σαν κλειδί το index , και σαν value μια τουπλά (α,β), όπου α=(γεωμετρικό μήκος, γεωμετρικό πλάτος) και β=1 το οποίο θα μας χρησιμεύσει στον reducer παρακάτω για να υπολογίσουμε το πόσες εμφανίσεις είχαμε για το συγκεκριμένο index.
- Στο επόμενο βήμα έχουμε την εφαρμογή του Reducer που παίρνει όλα τα στοιχεία με το ίδιο key , δηλαδή index, και αθροίζει σε μια μεταβλητή όλα τα γεωμετρικά μήκη , σε μια άλλη μεταβλητή όλα τα γεωμετρικά πλάτη και σε μια άλλη όλους τους άσσους από κάθε στοιχείο .Το αποτέλεσμα αυτού του Reducer είναι 5 στοιχεία .
- Στο τελευταίο βήμα μέσω ενός Mapper παίρνουμε αυτά τα 5 στοιχεία και διαιρούμε τα αθροίσματα με τον counter για να πάρουμε τον μέσο όρο των μηκών και πλατών. Το αποτέλεσμα αυτό το αναθέτουμε στο centroids και πηγαίνουμε στην επόμενη επανάληψη.

11. Έπειτα από τις 3 επαναλήψεις θέλουμε να αποθηκεύσουμε τα αποτελέσματα μας , αλλά θέλουμε το key-index κάθε κεντρικής συντεταγμένης να μην αρχίζει από το 0. Οπότε μέσω ενός Mapper αυξάνουμε κατά 1 τον index σε κάθε εγγραφή.

12. Εκτυπώνουμε τα αποτελέσματα μας και τα αποθηκεύουμε σε ένα αρχείο με το όνομα **output** στο hdfs.