Τμήμα Μηχανικών Η/Υ και Πληροφορικής
Πολυτεχνική Σχολή

ΟΜΑΔΙΚΉ ΕΡΓΑΣΙΑ ΔΙΑΧΥΤΟΣ ΥΠΟΛΟΓΙΣΜΟΣ

3ο μέρος

ΟΜΑΔΑ 0

ΝΑΚΚΑΣ ΝΙΚΟΛΑΟΣ ΑΜ:1054359

ΣΥΡΟΚΩΣΤΑΣ ΚΩΝΣΤΑΝΤΙΝΟΣ ΑΜ:1054339

ΠΑΠΑΚΟΣ ΜΙΛΤΙΑΔΗΣ ΑΜ:1054360

Λήψη δεδομένων

Για την εκπαίδευση του μοντέλου μηχανικής μάθησης της ομάδας μας κατεβάσαμε τα δεδομένα των συσκευών της ομάδας μας καθώς και όσων άλλων είχαν αντίστοιχη δομή στα δεδομένα που ανέβαζαν. Για να το κάνουμε αυτό καθώς το ThingsBoard Community Edition δεν επιτρέπει τη δημιουργία συνολικών reports φτιάξαμε ένα widget το οποίο κατεβάζει όλα τα δεδομένα σε μορφή CSV. Τον κώδικα για το widget μπορείτε να τον βρείτε στο Widget Bundle της ομάδας μας στο ThingsBoard.

Προεπεξεργασία δεδομένων

Τα δεδομένα που πήραμε από τους χρήστες τα επεξεργαστήκαμε με κατάλληλο τρόπο ώστε να έχουμε καλύτερα αποτελέσματα πρόβλεψης. Συγκεκριμένα κάναμε drop μη αριθμητικά δεδομένα και δεδομένα που υποδηλώνουν ταυτότητα συσκευής χρηστή. Ακόμα NaN δεδομένα αντικαταστάθηκαν με 0. Υπήρξε πείραμα να μπει αντί για 0 η μέση τιμή της εκάστοτε στήλης αλλά είχε μικρότερη απόδοση το μοντέλο μας.

Δημιουργία και εκπαίδευση μοντέλου

Δημιουργήσαμε ένα μοντέλο χρησιμοποιώντας sklearn βιβλιοθήκη και έτοιμο μοντέλο LinearRegression . Το μοντέλο αυτό μας έδωσε σχετικά καλά αποτελέσματα με ακρίβεια 37%.

Η ακρίβεια είναι σχετικά μικρή ωστόσο μας επιτρέπει να κάνουμε integrate το μοντέλο στο σύστημα μας.

Τον κώδικα του μοντέλου μας μπορείτε να τον βρείτε σε ένα .zip αρχείο στο github όπου υπάρχει όλος ο κώδικας της ομάδας μας. Το link για το συγκεκριμένο αρχείο είναι: https://github.com/NikoNakkan/Diaxytos/blob/master/MLModel.zip

Μεταφορά μοντέλου στο cloud

Σε αντίθεση με τον αρχικό σχεδιασμό μας (αναφορά 1) αποφασίσαμε να μην ενσωματώσουμε το μοντέλο στην εφαρμογή μας με την χρήση του TensroFlow Lite αλλά να φτιάξουμε ένα cloud end-point με τη χρήση του Google App Engine όπου το μοντέλο θα τρέχει. Το μοντέλο εκπαιδεύτηκε τοπικά με τα δεδομένα που κατεβάσαμε από το ThingsBoard, αποθηκεύτηκε σε μορφή .pkl και ανέβηκε στο cloud.

Έτσι κάθε φορά που η εφαρμογή μας χρειάζεται να κάνει μία πρόβλεψη κάνει μία POST request στο endpoint https://github.com/NikoNakkan/Diaxytos/blob/master/MLModel.zip

περνώντας τις παραμέτρους εισόδου στο σώμα του request και ως απάντηση παίρνει την τιμή που προβλέπει το μοντέλο.

Το Cloud Endpoint έχει γραφτεί σε Python για να είναι συμβατό με το μοντέλο και η βιβλιοθήκη Flask χρησιμοποιήθηκε για τη διαχείριση των endpoints.

Η συγκεκριμένη προσέγγιση για την ενσωμάτωση του μοντέλου στην εφαρμογή μας, μας προσφέρει τα εξής πλεονεκτήματα:

- Μείωση κατανάλωσης των πόρων του συστήματος (CPU, μνήμης) αφού η πρόβλεψη δεν γίνεται στη συσκευή.
- Μείωση μεγέθους εφαρμογής.
- Ευκολία αλλαγής του μοντέλου καθώς νέα δεδομένα έρχονται από τους χρήστες (καθώς αν αλλάξουμε το μοντέλο στον server διατηρώντας το ίδιο endpoint url η αλλαγή θα είναι εμφανής και στην εφαρμογή του χρήστη χωρίς αυτός να χρειαστεί να κάνει κάποιο update).
- Μεγαλύτερη ακρίβεια στις προβλέψεις του μοντέλου (37%), επειδή όταν υλοποιήσαμε το μοντέλο με TensorFlow και Keras για να το εκάστοτε την εφαρμογή μας (μέσω TensorFlow Lite) η ακρίβεια των προβλέψεων μας έπεσε στο 3%.

Βέβαια αυτή η προσέγγιση έχει και ορισμένα μειονεκτήματα το βασικότερο εκ των οποίων είναι η κατανάλωση δικτύου (και ειδικά δεδομένων όταν αυτά χρησιμοποιούνται από τον χρήστη) για να γίνει το request. Ωστόσο η κατανάλωση δικτύου από μία POST request είναι πολύ μικρή και γι’ αυτό αποφασίσαμε να ακολουθήσουμε αυτή την προσέγγιση.

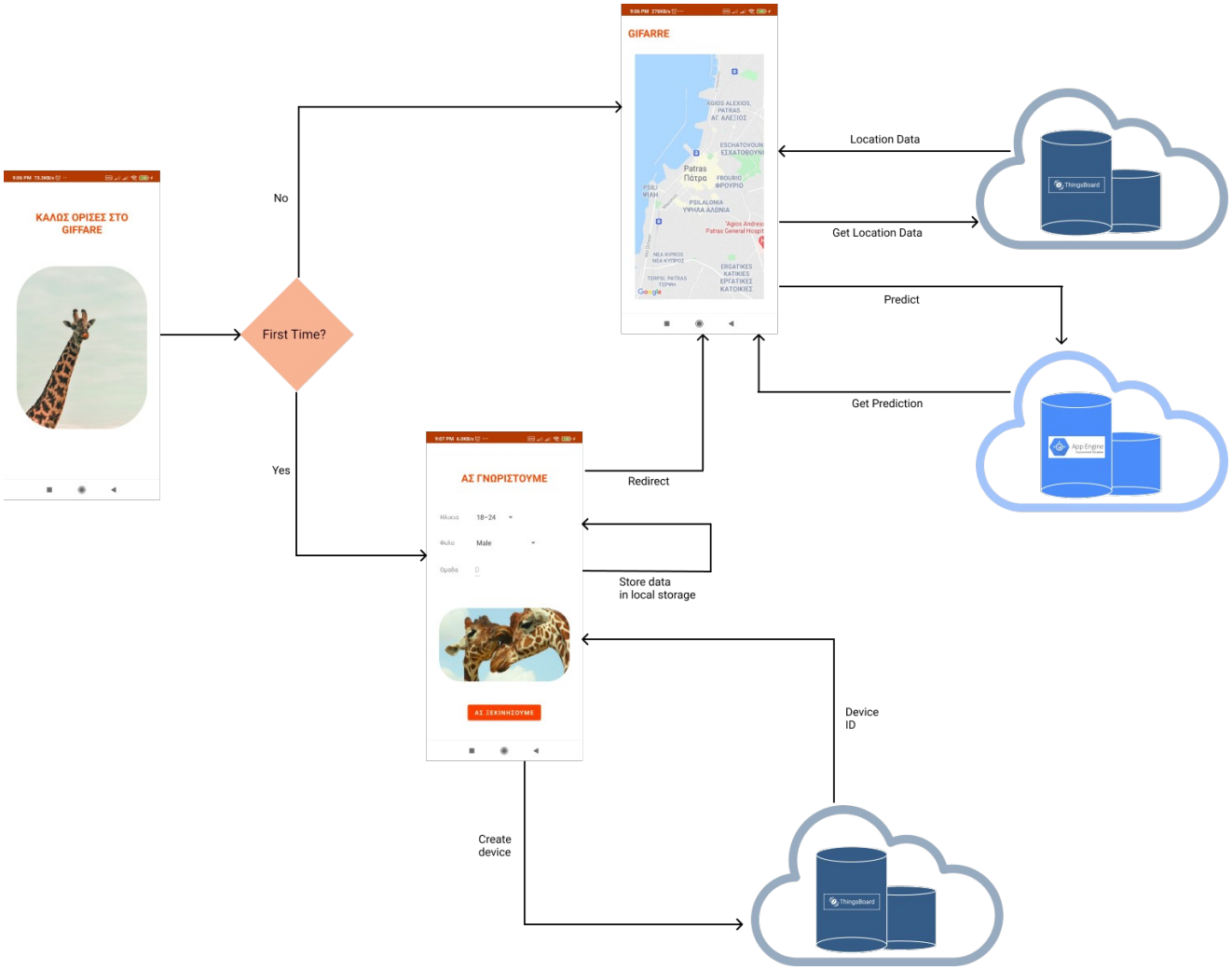
Τον κώδικα του Cloud Endpoint μας μπορείτε να τον βρείτε σε ένα .zip αρχείο στο github όπου υπάρχει όλος ο κώδικας της ομάδας μας. Το link για το συγκεκριμένο αρχείο είναι:
https://github.com/NikoNakkan/Diaxytos/blob/master/AppEngine.zip

Χρήση μοντέλου

Η εφαρμογή χρησιμοποιεί το μοντέλο με τον εξής τρόπο. Κάθε φορά που ο χρήστης ανοίγει το κινητό του και η εφαρμογή εντοπίζει αυτή την ενέργεια ενεργοποιείται ένα alarm* με συχνότητα περίπου 15 λεπτά (η μέθοδος *setInexactRepeating* χρησιμοποιείται για εξοικονόμηση ενέργειας). Κάθε φορά που το alarm γίνεται triggerred η εφαρμογή στέλνει ένα request στο cloud endpoint με τα δεδομένα που χρειάζεται το μοντέλο για την είσοδο. Το μοντέλο προβλέπει ένα timestamp με βάση αυτά τα δεδομένα εισόδου και το επιστρέφει στην εφαρμογή. Το timestamp αυτό ουσιαστικά είναι η πρόβλεψη του μοντέλου σχετικά με το πότε θα “έπρεπε” να είχε γίνει ένα event (κλείσιμο οθόνης) με βάση τα

συγκεκριμένα δεδομένα εισόδου. Σε περίπτωση που αυτό το timestamp είναι μικρότερο από το τωρινό (ο χρήστης θα “έπρεπε” να είχε κλείσει το κινητό του νωρίτερα) η εφαρμογή βγάζει ένα notification στον χρήστη που του προτείνει να κλείσει το κινητό. Όταν ο χρήστης κλείσει το κινητό του το συγκεκριμένο alarm ακυρώνεται και θα ενεργοποιηθεί ξανά όταν ο χρήστης ξανανοίξει το κινητό του.

Έτσι η τελική αρχιτεκτονική του συστήματος μας είναι:



Μπορείτε να βρείτε τον κώδικας ολόκληρης της εφαρμογής μας στο github, σε αυτό το link: https://github.com/NikoNakkan/Diaxytos

* όταν λέμε alarm δεν αναφερόμαστε σε ένα ξυπνητήρι στο κινητό του χρήστη αλλά σε ένα επαναλαμβανόμενο event όπως ορίζεται εδώ