SensitiveMood Project Presentation

Group 19

Project Description

What we do: SensitiveMood is a mobile app that keeps track of your friends' sentiment based on tweet data and detects device's anomaly based on accelerometer data.

How we do it: We develop our anomaly detection (*sentiment and device movement) model using unsupervised machine learning algorithm and statistics. The entire architecture is microservices accomplished using serverless framework.

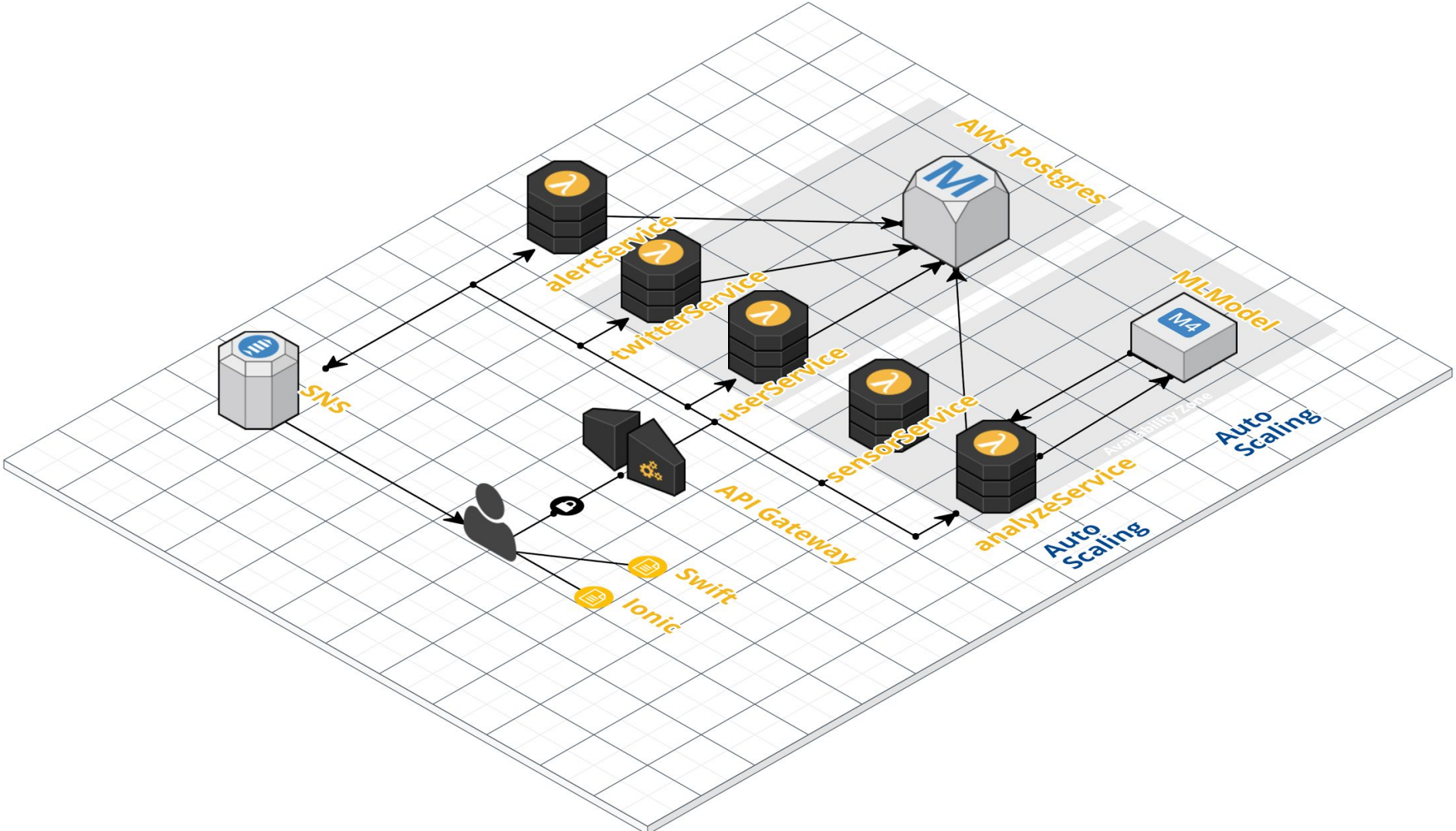
Architecture Solution

Inspired by the guest lecturer and impressed with its auto-scalability and lightweightness, we invest our time in learning how to build applications using serverless framework.

Microservices: we divide the entire backend system into 5 different microservices, UserService, TwitterService, SensorService, AlertService, and a powerful, machine learning based AnalyzeService.

Serverless Framework: AWS Lambda + AWS API Gateway + AWS RDS

Alert Functionality: AWS SNS


AWS postgres
alertService
twitterService
userService
sensorService
analyzeService
SNS
API Gateway
ML Model
M4
Auto Scaling
Auto Scaling
Swift
Ionic

Modelling - How it works

Goal: set alert threshold by first profiling users through K-means clustering.

Alert logic: predict which cluster this user belongs to -> get associated threshold -> alert if the user’s next window behavior breaches the threshold

Modelling - K-means

1. Tweet-data based clustering model

Features: follower count, favorite count, friend count, status count, total activity count, averaged tweet sentiment score (context-window size =7 days)

2. Accelerometer-data based clustering model

Features: averaged X coordinates, averaged Y coordinates, averaged Z coordinates value, accumulative X coordinates changing value, accumulative Y coordinates changing value, accumulative Z coordinates changing value (context-window = 1 second)

Modelling - Tweet-data based model

Cluster ~1000 unique twitter users historical behavior

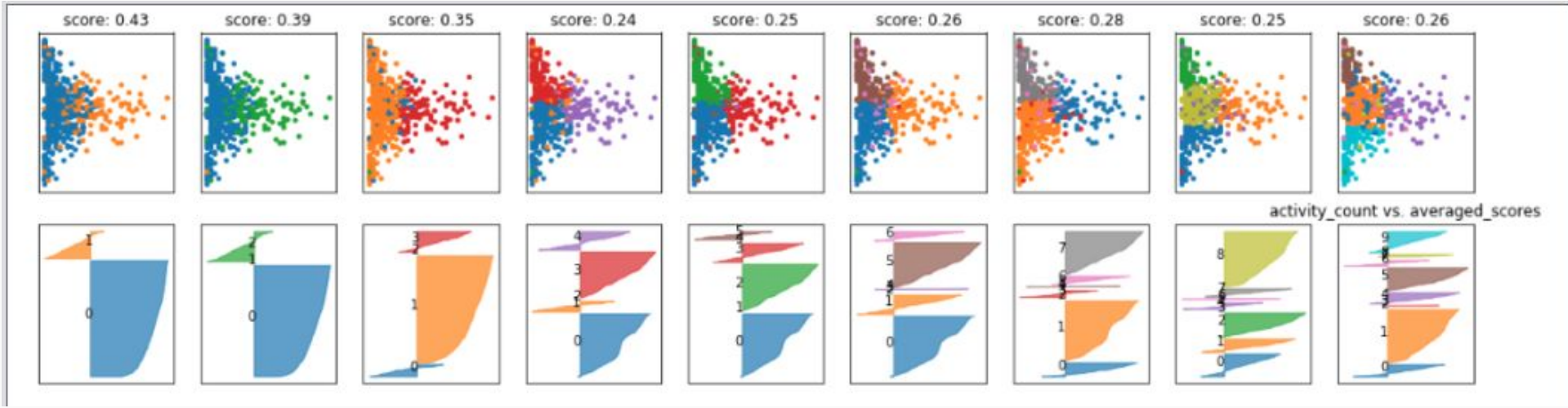
Found 2 clusters based on silhouette score.

Examine the cluster center to determine the characteristics of each cluster.

- Smaller cluster has large activity, follower, friend, status count, but relatively negative sentiment score.

Calculate standard deviation of sentiment score and account activity and use that as the alert threshold.

Modelling-Twitter Model-Silhouette Plot



Modelling - Accelerometer-data based model

Cluster ~10,000 second-level aggregated accelerometer features collected from 387 unique device.

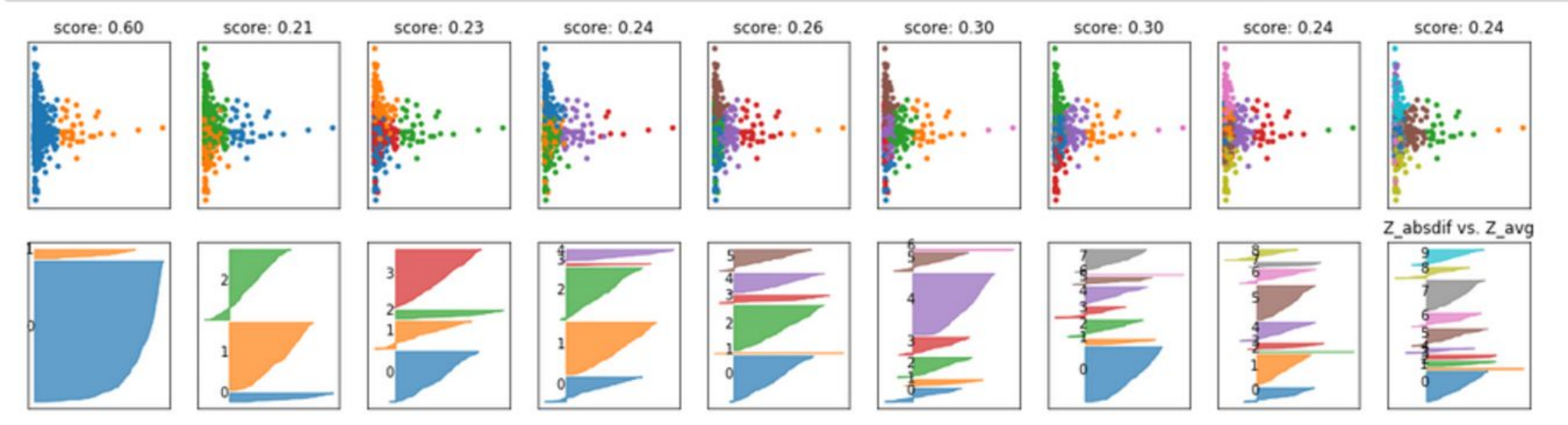
Found 2 clusters based on silhouette score.

Examine the cluster center to determine the characteristics of each cluster.

- Smaller cluster has large x,y,z absolute changes (more tilt, lift, twist movement)
- Large cluster is relatively more stable

Calculate standard deviation of each feature and use that as the alert threshold.

Modelling-Accelerometer Model - Silhouette Plot



Alert Logic Summary:

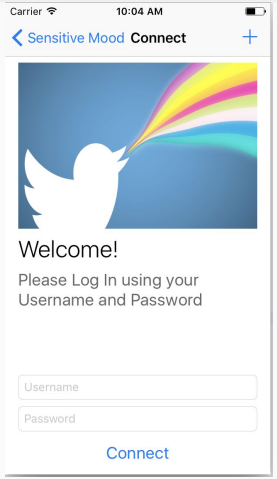
1. Twitter Model

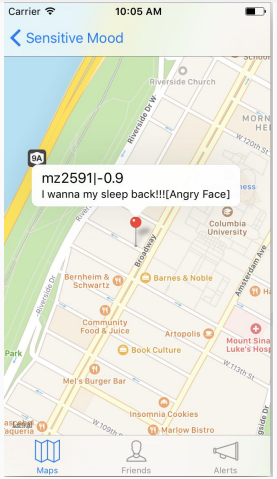
Step1. For each new user, wait till we have one-week (T1) worth of data, aggregate it and send feature values to the clustering model for profiling.
Step2. Use the model prediction (cluster number) to set its alert threshold.
Step3. At the end of second week (T2), we calculate the accumulated activities and averaged sentiment score in the past week, comparing against the threshold and send out alerts if apply.
Step4. Redo Step1-2-3 (use the current T2's aggregated features to reprofile user, and get updated threshold, and apply T3's sentiment and activity score to send alert)

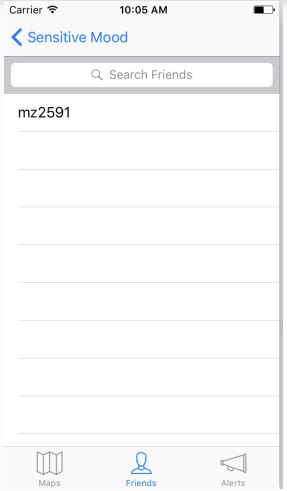
2. Accelerometer Model

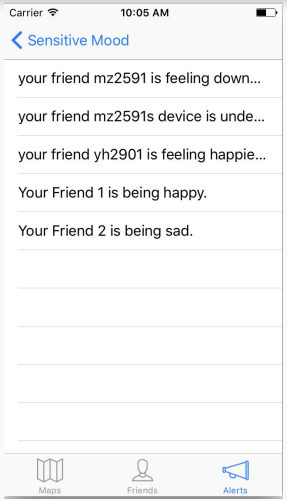
Step1: get aggregated second level accelerometer data for each device, send them through the model to profile the behavior. Get the alert threshold based on predicted cluster number.
Step2: schedule a job to get new accelerometer data of a later context window, if upon re-clustering, the new cluster number changed from cluster 0 to cluster 1, then we alert the user that the device of interest is undergoing drastic movement; if upon sampling, the newly aggregated accelerometer data breaches the stored threshold, we alert the user of the device's abnormal behavior as well.

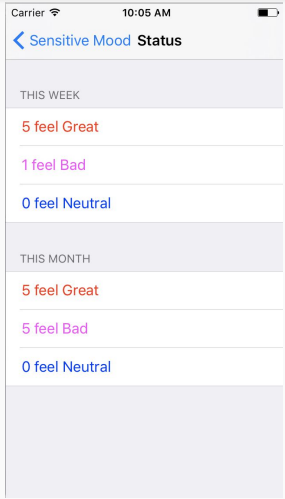
Screen Shots

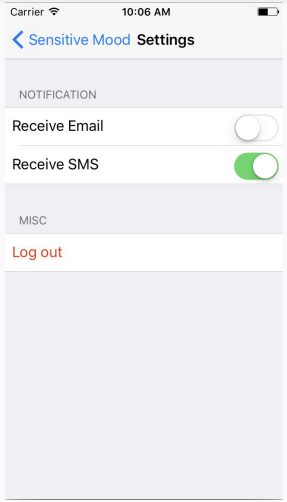












Demo

Eye beholds ~

Code Flow

```sql
# better formatted for reading:
SELECT a.T_seclevel as T, avg(a.X) as X_avg, avg(a.Y) as Y_avg, avg(a.Z) as Z_avg,
         sum(a.X_absdiff) as X_absdiff, sum(a.Y_absdiff) as Y_absdiff, sum(a.Z_absdiff) as Z_absdiff
FROM
 (SELECT Device,from_unixtime(CAST(T/1000 as BIGINT), 'yyyy-MM-dd HH:MM:SS')  as T_seclevel, X, Y, Z,
        abs(LAG(X, 1, 0) OVER (PARTITION BY  Device ORDER BY T) - X) as X_absdiff,
        abs(LAG(Y, 1, 0) OVER (PARTITION BY  Device ORDER BY T) - Y) as Y_absdiff,
        abs(LAG(Z, 1, 0) OVER (PARTITION BY  Device ORDER BY T) - Z) as Z_absdiff
  FROM acc where Device = 7)a
GROUP BY a.Device , a.T_seclevel;
```

sql

Topics **SNS topic for Alerts**

Publish to topic | Create new topic | Actions

Filter

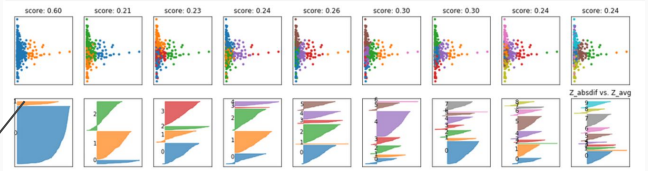
Name	ARN
AlertEmail	arn:aws:sns:us-east-1:472999334680:AlertEmail
AlertSMS	arn:aws:sns:us-east-1:472999334680:AlertSMS
AlertTopic	arn:aws:sns:us-east-1:472999334680:AlertTopic
CCTopic	arn:aws:sns:us-east-1:472999334680:CCTopic

Alertservice

```
import * as AWS from "aws-sdk";
import * as Lambda from "aws-lambda";
import * as Request from 'request-promise';

import { DbConnection } from "./db/dbconnection";

export async function handler(event: any, context: Lambda.Context, callback: Lambda.C
    let sns = new AWS.SNS();
    let conn = new DbConnection();

    let accRes = await conn.query('SELECT deviceid, a.T, avg(a.X) as X_avg, avg(a.Y)

    let d14 = new Date();
    d14.setDate(d14.getDate() - 14);
    let d7 = new Date();
    d7.setDate(d7.getDate() - 7);

    let lateTweet = await conn.query('select  userid, avg(followercount) as followert
    let newTweet = await conn.query('select  userid, avg(followercount) as followerCo

    for (var i = 0; i < lateTweet.length; i++) {
        var lt = lateTweet[i];
        var nt = newTweet[i];
        let tres = JSON.parse(await Request.get('http://35.184.204.97:8081/getTstats

        let acount = Math.abs(nt.activity_count - lt.activity_count);
        let sscore = nt.averaged_scores - lt.averaged_scores;
        let sscoreabs = Math.abs(sscore);

        if (acount > tres.activity_count_std) {
            await conn.query('INSERT INTO alert (userid, friendid, alertcontent, crea
            sns.publish({
                Message: `Your Friend ${nt.userid} is having abnormal activities.`,
                TopicArn: 'arn:aws:sns:us-east-1:472999334680:AlertEmail'
            }, (e, r) => {
            });
            sns.publish({
                Message: `Your Friend ${nt.userid} is having abnormal activities.`,
                TopicArn: 'arn:aws:sns:us-east-1:472999334680:AlertSMS'
            }, (e, r) => {
            });
        }

        if (sscoreabs > tres.sentiment_score_std) {
            await conn.query('INSERT INTO alert (userid, friendid, alertcontent, crea
            sns.publish({
                Message: `Your Friend ${nt.userid} is being ${sscore > 0 ? 'happy' :
                TopicArn: 'arn:aws:sns:us-east-1:472999334680:AlertEmail'
            }, (e, r) => {
            });
            sns.publish({
                Message: `Your Friend ${nt.userid} is being ${sscore > 0 ? 'happy' :
                TopicArn: 'arn:aws:sns:us-east-1:472999334680:AlertSMS'
            }, (e, r) => {
            });
        }
    }
```

model

```
# scale data
s = StandardScaler()
X_scaled = s.fit_transform(X_train)

%matplotlib inline
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
fig, axes = plt.subplots(3, 9, subplot_kw={'xticks': (), 'yticks':()}, figsize=(20, 5))
plt.title("Z_absdif vs. Z_avg", loc='center')
for ax, n_clusters in zip(axes.T, [2, 3, 4, 5, 6, 7, 8, 9, 10]):
    X = X_scaled
    km = KMeans(n_clusters=n_clusters)
    km.fit(X)
    ax[0].scatter(X[:,predic_map['Z_absdif']], X[:,predic_map['X_avg']],c=plt.cm.Vega10(km.labels_), s=10) #'activity_c
    silhouette_plot(X, km.labels_, ax=ax[1])
    ax[0].set_title("score: {:.2f}".format(silhouette_score(X, km.labels_)))
```



Thank You!