Improving outlier detection based on geographical position

Lukas Andrijauskas

The Introduction

I am working in a company focused on app development and data processing. Each day, hour, or minute they get a lot of new information **measured in terabytes**. The company is collecting that data from 3300 regions (ADA's) that drastically range in size, population, and amount of users. Because of this, it is impossible to check company is gathering a "nominal" amount of data from each region. In this project, we will be focusing on dataflow.

The Problem

Since we are talking about actual geographical regions, it is essential to notice that they could somehow affect each other. For instance, some areas had a **blackout** because of a **natural disaster** or a **regional holiday**. These events affect people that live there. In turn, they will affect the number of people that use a specific application. It might **spike** or **dip** in usage. It is fair to recognize that some neighboring regions or **"clusters"** might have the same events that affect their user numbers.
Also, **remember that an outlier is not equal to an outlier**. Some outliers can be just **"minor outliers"** that spiked or dipped just by chance. **"Minor outliers"** are **insignificant** because they do **not affect the data much**. We are interested in **"major outliers"** that are happening for a reason. A good indicator of **"major outliers" are regions with additional outliers near them—more reasons to look for these clusters.**

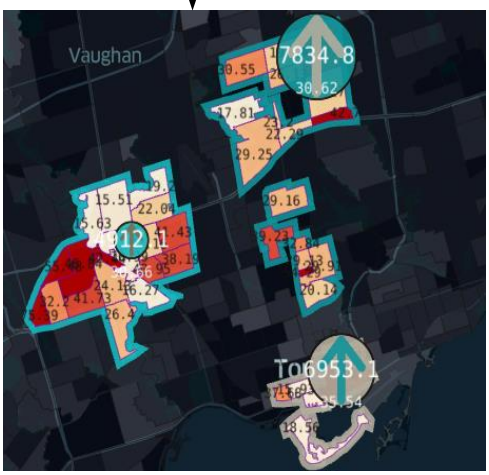
To find true outlier regions, we have to take into account their geographical location and disregard lone and minor outliers

The solution

Finding outlier candidates
We used the standard score method. We set a primary threshold z-score value to catch regions that might be outliers. We added a constant (additional sigma) to add to the standard deviation to skip really small outliers (bigger std, smaller z score).

Finding clusters
Clustered nearby regions by setting a buffer zone for each area to indicate whether the nearby areas are in one cluster. In the end we summed up individual regions' values to see clusters as one big region.

Finding outliers
We used the standard score method again but this time on clusters and with a bigger z-score. Clusters with more than one region have way bigger z-score values, so we automatically filter all lone outliers with minor deviations by using a second, more giant threshold.



In the end, we visualized our results using keplerGL. We chose KeplerGL because it has zoom and explores capabilities, and visualization is not static. You can modify it by adding or removing layers, filtering out data, and so on.