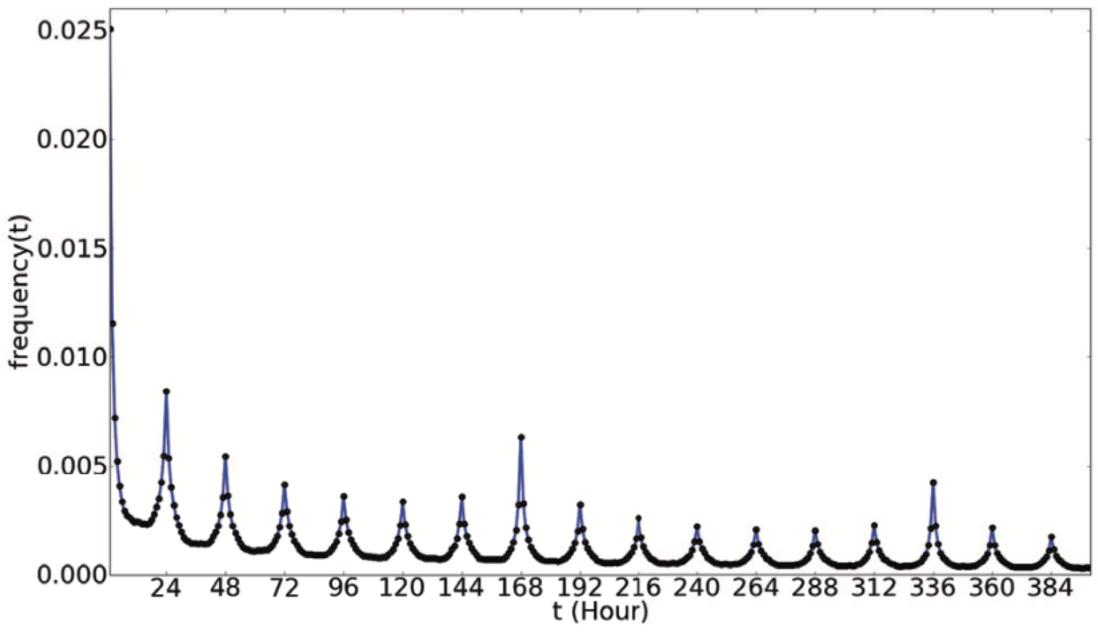
First, we will compare the properties of all users “from” New York City versus users “from” Los Angeles. Since users of location sharing services are not required to register a home location, we must algorithmically determine the home location. Note that choosing a user’s home based on the center of mass of all check-ins suffers from splitting-the-difference, by placing a user from Houston who occasionally travels to Dallas some- where in between the two cities; alternatively, directly considering the user’s most frequently checked-in venue may overlook a cluster of closely located but less individually checked-in venues. To avoid these drawbacks, we propose a simple method to geo-locate a user’s home based on a recursive grid search. First, we group check-ins into squares of one degree latitude by one degree longitude (covering about 4,000 square miles). Next, we select the square containing the most check-ins as the center and select the eight neigh- boring squares to form a lattice. We divide the lattice into squares measuring 0.1 by 0.1 square degrees and repeat the center and neighbor selection procedures. This process repeats until we arrive at squares of size 0.001 by 0.001 square degrees (covering about 0.004 square miles). Finally, we select the center of the square with the most check-ins as the “home” of the user.

Check-ins in United States

Though it is a good way of finding each user’s home but in my opinion, it is not accurate. Because not all users always share their locations every time, they visit some place. Also, some people tend to share location only when they travel at a far place or maybe sometimes, they don’t visit the nearby places that much; which can often lead to misunderstanding that maybe that is their home place. So, predicting someone’s home as the center of the cluster is not accurate.

My suggestion is to use geo-location data. Its features can also be structured as regular tabular data with numerical or categorical variables. A deep learning model could potentially learn location-based embeddings for geographical areas. For instance, embeddings based on leveraged to capture feature patterns that distinguished each such area. With the help of GPS

data of mobile phones, now it is possible to capture, track, analyze the locations and movements of people more and this opens the plethora of fascinating applications that rely on understanding location data. So, it can give more accurate data.
Returning probability – is a measure of periodic behavior in human mobility patterns. Periodic behavior is common in people's daily life (e.g., visits to work or school every weekday; visits to the grocery store on weekends) and echoes periodic behavior observed in animal migrations when animals visit the same places at the same time each year. We measure periodic behavior by the *returning probability* (or, first passage time probability), which is the probability that a user returns to a location that she first visited t hours before. Grouping all returning times of all check-ins into buckets of one-hour, we plot the distribution of returning times in which the x-axis represents the bucket of returning time, and the y-axis is the corresponding frequency for a bucket. For example, at 168 hours, the re- turning probability peaks, indicating a strong weekly return probability. Similarly, we see daily return probabilities. As time moves forward, the returning probability shows a slight negative slope, indicating the aggregate forgetfulness of visiting previously visited places (that is, the return probability is strongest for places we have visited most recently).



Distribution Of Returning Probability
Returning Probability can measure periodic behavior because if we can measure how many times a place has been visited by a user, also what is his returning probability at that place again it gives us a clear picture of where the user lives. Because finding out the frequency of visiting workplaces, groceries and other common area says a lot about the user's location. So, returning probability can measure periodic behavior of user.
Among the data preprocessing techniques described in the paper my favorite one is "Returning Probability".
Data Preprocessing:

Definition: When we talk about data, we usually think of some large datasets with a huge number of rows and columns. While that is a likely scenario, it is not always the case — data could be in so many different forms: Structured Tables, Images, Audio files, Videos, etc. In any machine learning process data preprocessing is that step in which data gets transformed or encoded to bring it to a state where the machine can easily parse it. In other words, the features of the data can be easily interpreted by the algorithm.
Data Preprocessing for Returning Probability (General Steps and An Example):
1. Data quality assessment: We need to take a good look at data and get an idea of its overall quality, relevance to our project, and consistency.
Mismatched data types: When we collect data from many different sources, it may come in different formats. While the goal of this entire process is to reformat data for machines, we still need to begin with similarly formatted data.
Mixed data values: Perhaps different sources use different descriptors for features. These value descriptors should all be made uniform.
Data outliers: Outliers can have a huge impact on data analysis results. To take care of missing data, we'll have to perform data cleaning.
2. Data cleaning: Data cleaning is the process of adding missing data and correcting, repairing, or removing incorrect or irrelevant data from a data set. Depending on the kind of data we're working with, there are several possible cleaners we'll need to run your data through.
Missing data: There are several ways to correct for missing data, but the two most common are:
Ignore the tuples: A tuple is an ordered list or sequence of numbers or entities. If multiple values are missing within tuples, we may simply discard the tuples with that missing information. This is only recommended for large data sets, when a few ignored tuples won't harm further analysis.
Noisy data: Data cleaning also includes fixing "noisy" data. This is data that includes unnecessary data points, irrelevant data, and data that's more difficult to group together.
Regression: Regression is used to decide which variables will apply to the analysis. Regression analysis is used to smooth large amounts of data.
After data cleaning, we may realize we have insufficient data for the task at hand. At this point we can also perform data wrangling or data enrichment to add new data sets and run them through quality assessment and cleaning again before adding them to our original data.
3. Data transformation: With data cleaning, we've already begun to modify our data, but data transformation will begin the process of turning the data into the proper format(s) we'll need for analysis and other downstream processes.
This generally happens in one or more of the below:
Aggregation
Normalization
Feature selection
Discreditation
Concept hierarchy generation
Aggregation: Data aggregation combines all our data together in a uniform format.
Normalization: Normalization scales our data into a regularized range so that you can compare it more accurately.

Feature selection: Feature selection is the process of deciding which variables (features, characteristics, categories, etc.) are most important to the analysis.
Discreditation: Discretization pools data into smaller intervals. It's somewhat like binning, but usually happens after data has been cleaned.
Concept hierarchy generation: Concept hierarchy generation can add a hierarchy within and between your features that wasn't present in the original data.
4. Data reduction: The more data we're working with, the harder it will be to analyze, even after cleaning and transforming it. Data reduction not only makes the analysis easier and more accurate but cuts down on data storage.
It will also help identify the most important features to the process at hand.
Attribute selection: Like discreditation, attribute selection can fit our data into smaller pools.
Numerosity reduction: This will help with data storage and transmission.
Dimensionality reduction: This, again, reduces the amount of data used to help facilitate analysis and downstream processes.
Example: We will be finding person A's returning probability at Walmart.
STEP1: DATA QUALITY ASSESSMENT:

Times	Frequency
3.45 PM	100
5.00 PM	50
8.00 PM	30

After the assessing the data suppose we got two errors that person A visits Walmart at 3.45 PM 94 times and 5.00PM 40 times which leads to us to our next step which is
STEP2: DATA CLEANING:

Times	Frequency
3.45 PM	94
5.00 PM	40

STEP3: DATA TRANSFORMATION:

Times	Frequency
3.45 PM	94
5.00 PM	40
8.00 PM	30

STEP4: DATA REDUCTION:
Now we sort the data by the frequency.

Times	Frequency
8.00 PM	30
5.00 PM	40
3.45 PM	94

From the above table we can predict that the returning probability of person A at 3.45PM is highest and 8.00 PM is lowest.