Mining frequent itemsets can be expensive. In a large, dynamic database of transactions, we can store the set of frequent itemsets and incrementally update that set upon arrival of a set of new transactions. Let *DB* denote the last state of our database and $\Delta DB$ a set of new transactions. The task is to incrementally determine the set of frequent itemsets in $DB \cup \Delta DB$ with respect to min-supp, without re-applying the Apriori-algorithm to the whole updated database $DB \cup \Delta DB$. More specifically, given the sets *DB* and $\Delta DB$ and the set of all frequent itemsets in *DB* together with their support, return the set of all frequent itemsets in $DB \cup \Delta DB$, without re-applying the Apriori-algorithm to the whole updated database $DB \cup \Delta DB$.

You can assume a (non-incremental) implementation of the Apriori-algorithm

Apriori (S: set of transactions, min-supp: float)

that returns all itemsets that are frequent in S together with their support. Note that min-supp is a relative frequency threshold.

a) Prove the following property: If an itemset is not frequent in *DB* and not frequent in $\Delta DB$, then it cannot be frequent in $DB \cup \Delta DB$.

Referred from lecture notes & textbook

Let $z$ denote minimum support threshold, and let $A$ denote some itemset

If $A$ is not frequent in $DB$:

$$\frac{|\{x \in DB \mid A \subseteq x\}|}{|DB|} < z$$

$$\Rightarrow |\{x \in DB \mid A \subseteq x\}| < z|DB|$$

If $A$ is not frequent in $\Delta DB$:

$$\frac{|\{x \in \Delta DB \mid A \subseteq x\}|}{|\Delta DB|} < z$$

$$\Rightarrow |\{x \in \Delta DB \mid A \subseteq x\}| < z|\Delta DB|$$

Using the above equations

$$|\{x \in DB \cup \Delta DB \mid A \subseteq x\}| < \underbrace{z|DB| + z|\Delta DB|}_{z(|DB| + |\Delta DB|)}$$

DB & ΔDB are disjoints

$$\therefore |DB \cup \Delta DB| = |DB| + |\Delta DB|$$

Now using the above two equations

$$\frac{|\{x \in DB \cup \Delta DB \mid A \in x\}|}{|DB \cup \Delta DB|} < \underbrace{\frac{z(|DB| + |\Delta DB|)}{|DB \cup \Delta DB|}}$$

$$\Rightarrow \frac{z\cancel{(|DB| + |\Delta DB|)}}{\cancel{(|DB| + \Delta DB)}}$$

$$\Rightarrow z$$

$$\therefore \frac{|\{x \in DB \cup \Delta DB \mid A \in x\}|}{|DB \cup \Delta DB|} < z$$

$\therefore$ A is infrequent in $DB \cup \Delta DB$

Also we can use Anti-monotonicity property which states:

Each subset a frequent itemset is also frequent

$$\forall T_1 \subset I, T_2 \subseteq I : T_1 \subseteq T_2 \wedge freq(T_2, D) \Rightarrow freq(T_1, D)$$

because of

$$\forall T_1 \subseteq I, T_2 \subseteq I : T_1 \subseteq T_2 \Rightarrow sup(T_1, D) \geq sup(T_2, D)$$

If an item set is not frequent, then any superset cannot be frequent.

b) Based on the property that we have proven in a), as a first step, the incremental Apriori algorithm applies the non-incremental Apriori algorithm to $\Delta DB$ to determine the frequent itemsets in $\Delta DB$ and their support. After having performed this first step, for which itemsets do you need to count the support in *DB*, and for which itemsets do you need to count the support in $\Delta DB$? Explain why the support counting is necessary.

The Apriori Algorithm does not return the support of the itemsets that are infrequent in DB. Therefore we will count the support in DB for those item-sets that are frequent in $\Delta DB$ and not in DB as these itemsets may become frequent in $DB \cup \Delta DB$

Again, the algorithm does not return the support of the itemsets that are infrequent in $\Delta DB$. Therefore we will count the support in $\Delta DB$ for those item-sets that are frequent in DB and not in $\Delta DB$ as these itemsets may become frequent in $DB \cup \Delta DB$

**Assignment 3.2 (60 Marks)**

Implement the Local Outlier Factor (LOF) algorithm in python. For every point, compute the LOF value and determine whether the point is an outlier or not, based on a threshold that you have to choose. Your program shall read a file of two-dimensional points in CSV-format and produce a plot of the dataset, where the color of a point is red if the point is an outlier and blue otherwise.

You have to implement LOF "from scratch", i.e. you cannot use an existing LOF implementation, but you do not have to implement the plot function from scratch.

Apply your program on the outliers-3.csv dataset provided **here** (https://coursys.sfu.ca/2020fa-cmpt-459-d1/pages/Assignment_3_Datasets) and assume the data set is in the same directory as the program.

Your program should accept one input, the hyperparameter k.

In your PDF report:

- Explain how you set the hyperparameter k.
- Explain how you choose the threshold for determining whether a point is an outlier or not.
- Attach a screenshot of the LOF plot for outliers-3.csv data set.

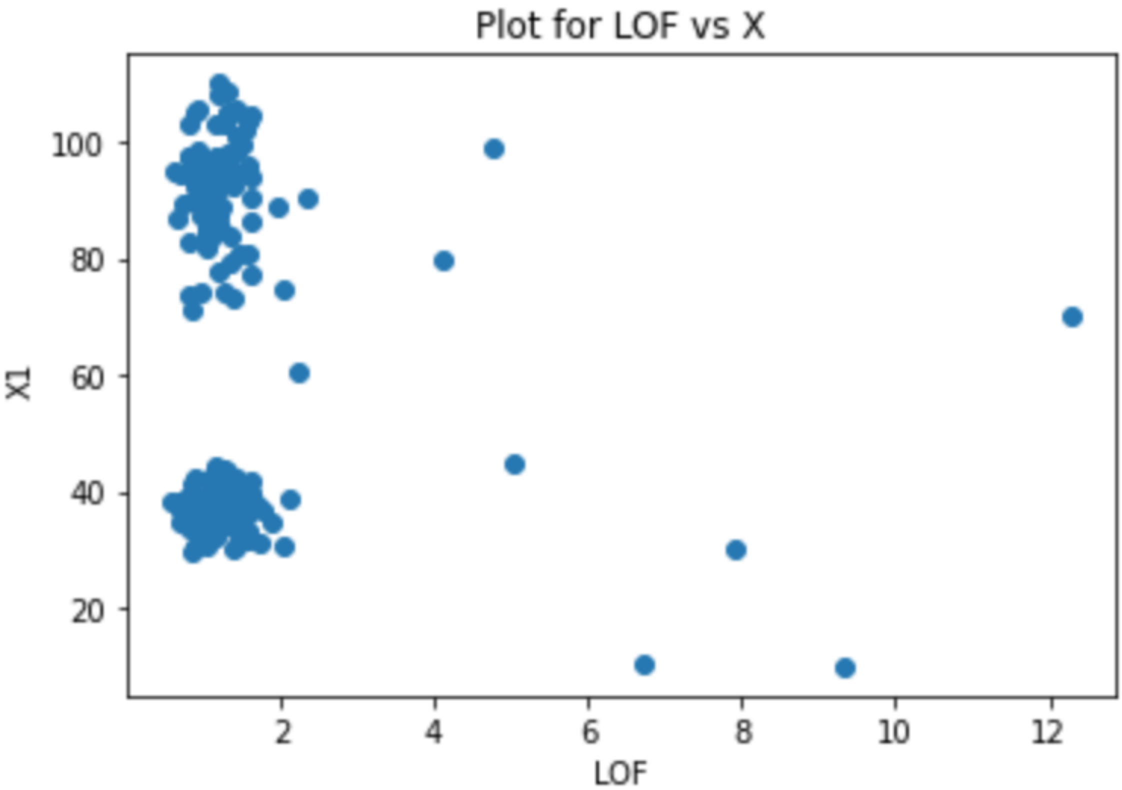
**Explain how you set the hyperparameter k**

If k is too small => not robust enough. If k is too big => not local enough

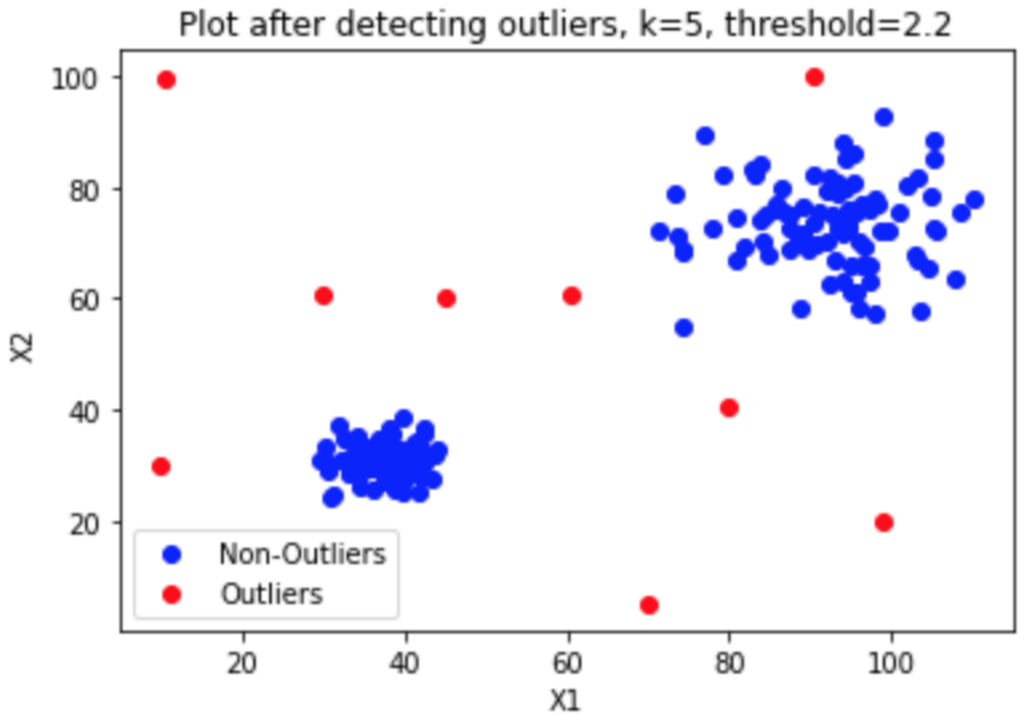
Therefore, k is chosen that maximizes value of LOF for a particular outlier. To set the hyperparameter k, we will first look at the shape of the data set by df.shape = (199, 2).Then decided to set k=sqrt(199)=floor(14.1067)=14. At k=14, I found this is not optimal, therefore I took two k's – min(k) and max(k) and started comparing values to till I find optimal k. Since the dataset wasn't big enough, I chose k=5 to be the optimal k for the dataset. I also tried trial and error method to see if k chosen at random works but the results were not good. k=5 gives the maximum LOF for my model.

**Explain how you choose the threshold for determining whether a point is an outlier or not.**

To determine a good threshold, I plotted LOF again X1 to check where most values lie and find pattern in dataset. From the point I realised most values are before LOF=2. Therefore I started with threshold = 2 and incremented it by 0.1 till I got an optimal threshold for my lof. As you can see in the graph below most points are below LOF=2. We can also normalize the lof values and plot them to find optimum threshold.




**Screenshot of the LOF plot for the data set:**



LINK FOR THE GOOGLE COLLAB:
https://colab.research.google.com/drive/1hx9z49LCx-4Xfh5UyE8sQyUdQ6cvdwtw?usp=sharing

How to run the code:
Please run the entire code from start to bottom, all together.