Synthetic datasets

1. Comparing run time between Sklearn and Top N LOF

The size of the data is 90,000 rows.
Parameters for Top N LOF: c = 1, d = 2/10/20 (depending on the dimensionality of the dataset), k = 20, n = 90000, r = 1
How data is generated: gaussian distribution with mean 0 and standard deviation 1
The parameter n is set to 90,000 so that Top N LOF will have to calculate the LOF scores for all points.
Sklearn's default k value is 20.
The training time can be very long (more than 8 hours for 10 dimensions and above) if there is no pruning.

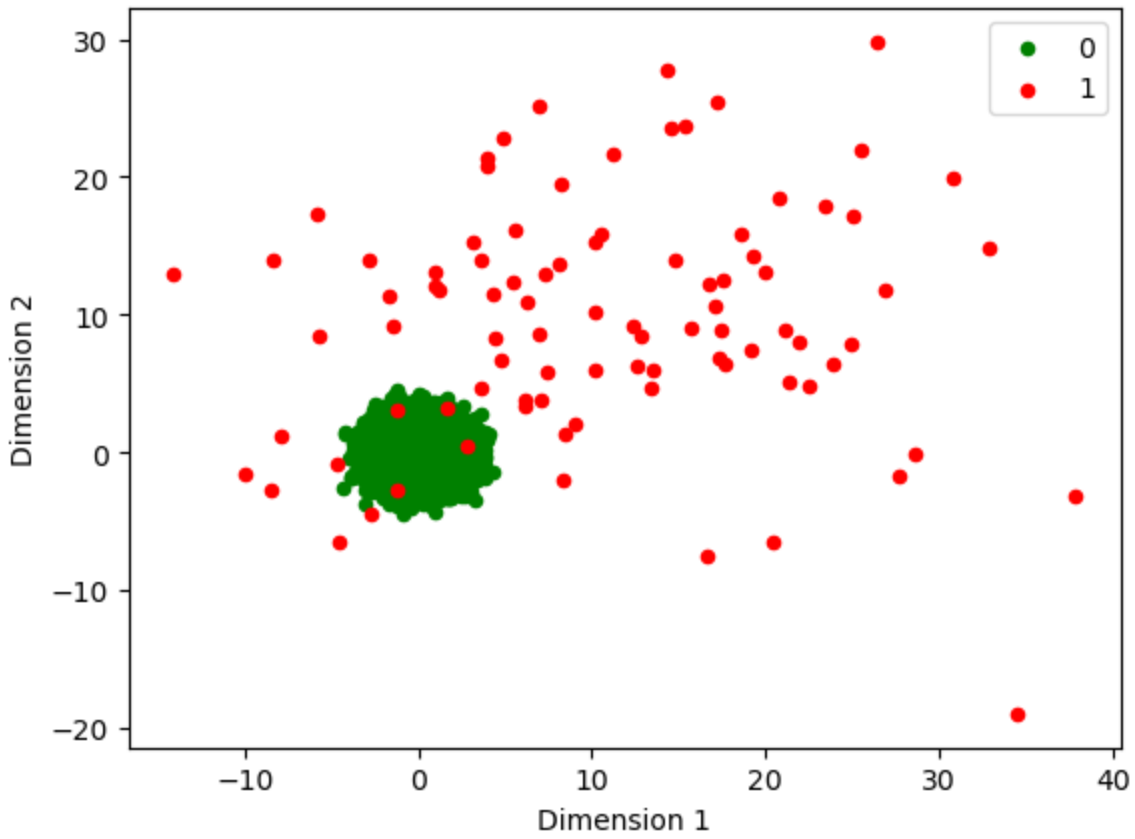
- Too small c and r can lead to very large cf trees and need a lot of rebuilding
- Too large k will lead to very long time in calculating the LOF values

Number of dimensions	Top N LOF (in seconds) with c = 2, d = 2/10/20, k = 2, n = 90000, r = 2	Top N LOF (in seconds) with c = 1, d = 2/10/20, k = 20, n = 90000, r = 1	Sklearn LOF (in seconds)
2	2	4	0.4
10	180	1914	36.9
20	492	2909	104.8

Training time is longer, which is expected since the tree in BIRCH clustering is built for nothing.

2. What if anomalies are introduced?

The figure below shows a sample of the synthetic dataset with anomalies. The inliers are sampled from a gaussian distribution with mean 0 and standard deviation 1 while the outliers are sampled from a different gaussian distribution with mean 10 and standard deviation 10.



The red points are outliers, whereas the green points are inliers.

Parameters for Top N LOF: c = 10, d = 2/10/20 (depending on the dimensionality of the dataset), k = 30, n = 90 (0.1% of 90000), r = 10

Number of dimensions	Top N LOF (in seconds)	Sklearn LOF (in seconds)
2	8	0.4
10	184	38.3
20	251	107.2

It seems that with anomalies, Top N LOF will take a shorter time. However, printout shows that no points/clusters are pruned, as shown below.

```
Set Arguments.......
K = 30
Top-N = 90
Input File Path =  C:\Users\User\Desktop\DA_toolkit\open_source_results\data\synthetic_2\dim_2.csv
Output File Path = C:\Users\User\Desktop\DA_toolkit\open_source_results\results\synthetic_2\dim_2
Dim =  2
Domain Range =  10.0
Cluster Radius Rate =  10.0
.............Start training CF-Tree..............
.............Start Generating Micro Clusters..............
Size of Micro Clusters: 1
.............Start Assigning Points to Micro Clusters..............
...................Build a R-Tree for Micro Clusters...........
..............End Preprocessing..............
..............Preprocessing takes 0 seconds.............
..............End Computing Kdistance Bounds..............
..............End Computing LOF Bounds..............
Number of Pruned Clusters: 0
Number of Pruned Points: 0
..............End Removing Micro Clusters..............
..............Start Computing KNN for Unpruned Points.............
..............Start Computing KNN for Some Pruned Points............. 0
..............Start Computing KNN for Some Pruned Points............. 0
..............Start Computing LRD for Unpruned Points.............
..............Start Computing LRD for Some Pruned Points............. 0
..............Start Computing LOF for Unpruned Points.............
```

Why the difference?

- Sklearn uses numpy while Top N LOF has for loops.
- No points/clusters are pruned (according to the printouts).
  Java code uses for loops to compute LOF, which is slower than sklearn's numpy.

Excessive overhead computational costs for Top N LOF leads to increased runtime.

3. Increasing the dataset size to 500,000 rows with no anomalies

How data is generated: gaussian distribution with mean 0 and standard deviation 1

Number of dimensions	Top N LOF (in seconds) with c = 2, d = 2/10/20, k = 2, n = 10, r = 2	Sklearn LOF (in seconds)
2	8	2.8
10	> 28800 (8 hours)	898.7
20	> 28800 (8 hours)	3071.7

The results are similar to experiment 1, where top n lof still took longer than sklearn LOF.

4. Comparing F1 score for outlier class

The size of data used is 90000.

How data is generated: 0.1% will be anomalies. Inliers follow a gaussian distribution with mean 0 and standard deviation 1. Outliers follow a gaussian distribution with mean 10 and standard deviation 10 (similar to experiment 2).
Parameters for Top N LOF: c = 10, d = 2/10/20 (depending on the dimensionality of the dataset), k = 30 (k = 20 will give same predictions as sklearn as no points are pruned), n = 90 (0.1% of 90000), r = 10
Parameters for sklearn: contamination set as 0.1%

Number of dimensions	Top N LOF	Sklearn
2	42.2%	33.3%
10	46.7%	35.6%
20	47.8%	38.9%

Sklearn has a lower f1 score, but this is only because k = 30 will give a better performance.

5. How does the parameter n affect Top N LOF's performance?

The dimensionality of the data is 2.
How data is generated: 0.1% will be anomalies. Inliers follow a gaussian distribution with mean 0 and standard deviation 1. Outliers follow a gaussian distribution with mean 10 and standard deviation 10. (similar to experiment 2)
To vary n, the size of the data used will be varied while maintaining the percentage of anomalies to be 1%.

Parameters for Top N LOF: c = 10, d = 2, k = 30, n = 1% of data size, r = 10
Parameters for sklearn: contamination set as 1%

Value of N	F1 Score (Top N LOF)	F1 Score (Sklearn)
9	88.9%	88.9%
90	47.8%	40%
900	28.3%	24.8%

Comparing the results, Top N LOF does better. If k = 20, then the results will be the same as no clusters are pruned.

As n increases, f1 score decreases. This is probably due to the fact that outliers are nearer to one another and less likely to be outliers since the data space remains largely unchanged.

6. Quick fix to be able to prune clusters

Size of data is 90000 rows
How data is generated: gaussian distribution with mean 0 and standard deviation 1
The pruning conditions are changed to allow pruning to occur.

Dimensions	Top N LOF (in seconds) with c = 1, d = 2/10/20, k = 2, n = 10, r = 1	Sklearn (in seconds)
2	2	0.4
10	1427	38.9
20	2449	104.7

Runtime is still longer. This is probably due to the usage of for loops in the Java codes.

No guarantee to be able to prune clusters if the ‘wrong’ hyperparameters are used.

7. Calculating R and tuning C

Size of data is 90000 rows
How data is generated: gaussian distribution with mean 0 and standard deviation 1
C has been tweaked so that clusters can be pruned and the number of clusters is not too many and not too little
R = l2 norm of the distance between centre point of data and furthest data point

D = 2/10/20 (depending on dimensionality of dataset)
K = 20, which is the default parameter for Sklearn LOF
N = 10

Dimensions	c	r	Top N LOF (in seconds)	Sklearn LOF (in seconds)
2	0.1	6.9	5.8	0.4
10	0.1	20.4	514.1	39
20	0.1	35.4	1042.6	101.7

Runtime is longer, which is expected since there is excessive overhead computation (i.e. BIRCH clustering).

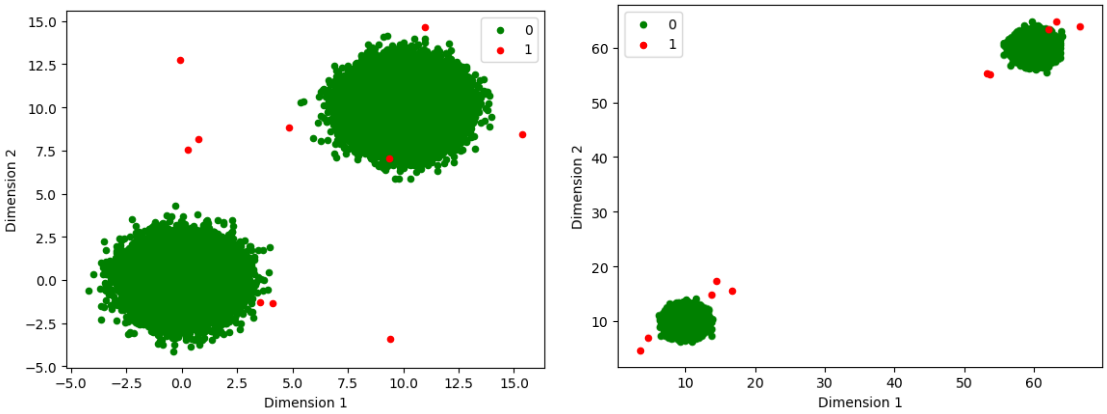
8. Comparing f1 score for outlier class

The algorithms will be compared using global and local outliers separately.

The size of the data used is 90000.
How data is generated
10 points will be anomalies, the rest will be inliers.
For global outliers: Inliers will be sampled from 2 different gaussian distributions randomly, 1 with a mean of 0 and standard deviation of 1 while the other has a mean of 10 and standard deviation of 1. Outliers will be sampled from a gaussian distribution with mean 5 and standard deviation 5.
For local outliers: Inliers will be sampled from 2 different gaussian distributions randomly, 1 with a mean of 10 and standard deviation of 1 while the other has a mean of 60 and standard deviation of 1. For each of the 10 anomalous points, a coin will be flipped to determine if it belongs to the first or second gaussian distribution and a value will be sampled from the randomly selected gaussian distribution. For each anomalous point, noise is sampled from a separate gaussian distribution with mean 5 and standard deviation 1. A coin will be flipped to determine if the noise will add to or subtracted from the value sampled previously for the anomalous point. Hence, the anomalous points will appear either to the left or to the right of the randomly selected gaussian distribution.
Parameters for sklearn: contamination set as 10 anomalous points
C has been tweaked such that clusters can be pruned and the clustering tree is not too big
R = l2 norm of the distance between centre point of data and furthest data point

D = 2/10/20 (depending on dimensionality of dataset)
K = 20, which is the default parameter for Sklearn LOF
N = 10

The plots below show the difference between local and global outliers. The plot on the left shows global outliers while the plot on the right shows local outliers. Green points are inliers whereas red points are outliers.



Local Outliers

Number of dimensions	c	r	Top N LOF	Sklearn LOF
2	0.025	60.7	90%	90%
10	0.0125	373.5	90%	100%
20	0.00625	628.1	80%	100%

Global Outliers

Number of dimensions	c	r	Top N LOF	Sklearn LOF
2	0.1	9.3	70%	70%
10	0.05	68.3	70%	100%
20	0.05	119.2	90%	100%

Tweaking c will affect the f1 score for both local and global outliers. Top N LOF does not perform better than Sklearn LOF.

Real-world datasets

9. ML Matt

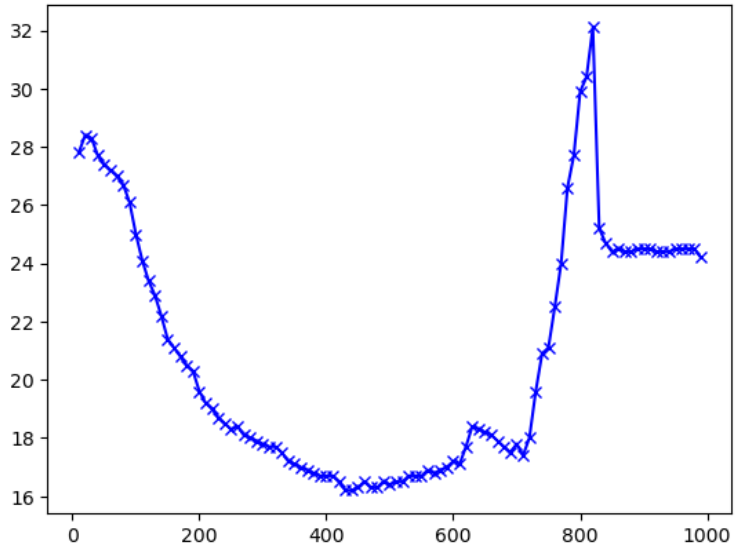
(https://www.kaggle.com/competitions/anomaly-detection-in-cellular-networks/data)
27.6% anomalies, 36815 rows, 44 columns
The algorithms will be compared by runtime and f1 score.

9.1 Vanilla Attempt

Top N LOF with r = 11.5, c = 0.003125, n = 10167, k = 20, d = 44	Sklearn LOF with contamination parameter set as the percentage of anomalous points
Runtime: 265 seconds F1 Score: 27.6%	Runtime: 18.3 seconds F1 Score: 28.1%

9.2 Tuning k

The dataset will be split into a 70:30 ratio, stratified by the class label. Both algorithms (top n lof and sklearn) will tune k separately. They will tune k on the training set and the algorithms will be compared by runtime and f1 score on the test set.
The value of k with the highest f1 score for each algorithm will be chosen.
Parameters for Top N LOF:
C = 0.003125 (found from the vanilla attempt)
D = 44
N = Number of anomalous points in training / test set
R = Determined from training / test set



The plot above shows the tuning process for Top N LOF. X-axis is the value of k while y-axis is the f1 score for the training set. Optimal k is found to be 820.
It turns out that sklearn has the same results as well.

Top N LOF with r =11.6, c = 0.003125, n =3050, k = 820, d = 44	Sklearn LOF with contamination parameter set as the percentage of anomalous points and k set as 820
Runtime: 105 seconds F1 Score: 20.7%	Runtime: 3 seconds F1 Score: 20.6%

10. Access behavior anomaly dataset

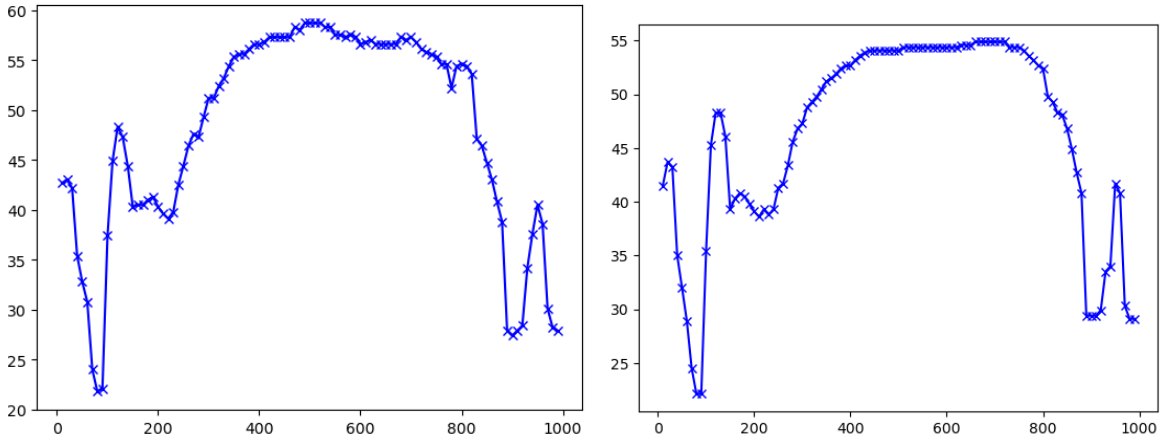
(https://www.kaggle.com/datasets/tangodelta/api-access-behaviour-anomaly-dataset?select=supervised_dataset.csv)
34.7% anomalies, 1695 rows, 9 columns
The algorithms will be compared by runtime and f1 score

10.1 Vanilla Attempt

Top N LOF with r = 4.4, c = 0.003125, n = 589, k = 20, d = 9	Sklearn LOF with contamination parameter set as the percentage of anomalous points
Runtime: 1 second F1 Score: 40.1%	Runtime: < 0.1 second F1 Score: 45.5%

10.2 Tuning k

The dataset will be split into a 70:30 ratio, stratified by the class label. Both algorithms (top n lof and sklearn) will tune k separately. They will tune k on the training set and the algorithms will be compared by runtime and f1 score on the test set.
The value of k with the highest f1 score for each algorithm will be chosen.
Parameters for Top N LOF:
C = 0.003125 (found from the vanilla attempt)
D = 9
N = Number of anomalous points in training / test set
R = Determined from training / test set



The plot on the left shows the tuning process for Top N LOF while the plot on the right shows the tuning process for sklearn. Both plots are largely similar since the LOF calculations for both algorithms are the same.X-axis is the value of k while y-axis is the f1 score for the training set. Optimal k for Top N LOF is found to be 490 while optimal k for sklearn is 660.

Top N LOF with r =4, c = 0.003125, n =177, k = 490, d = 9	Sklearn LOF with contamination parameter set as the percentage of anomalous points and k set as 660
Runtime: < 1 second F1 Score: 52.5%	Runtime: < 1 second F1 Score: 57.6%

11. Energy

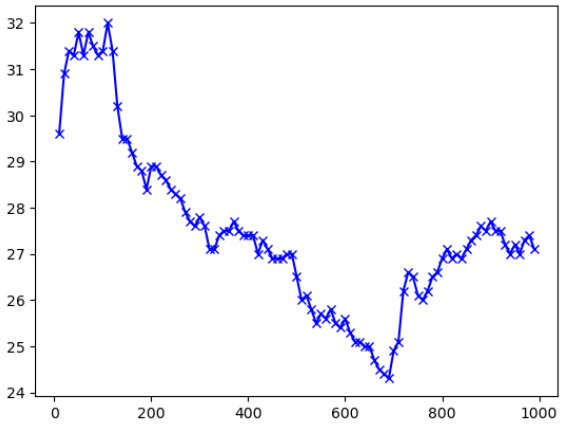
(https://www.kaggle.com/datasets/inIT-OWL/high-storage-system-data-for-energy-optimization)
23% anomalies, 19634 rows, 18 columns
The algorithms will be compared by runtime and f1 score

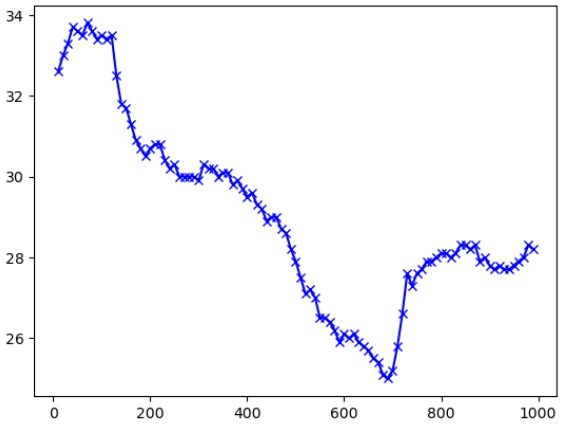
11.1 Vanilla Attempt

Top N LOF with r = 5.8, c = 0.00125, n = 4517, k = 20, d = 18	Sklearn LOF with contamination parameter set as the percentage of anomalous points
Runtime: 29 seconds F1 Score: 32.5%	Runtime: 3.4 seconds F1 Score: 34.2%

11.2 Tuning k

The dataset will be split into a 70:30 ratio, stratified by the class label. Both algorithms (top n lof and sklearn) will tune k separately. They will tune k on the training set and the algorithms will be compared by runtime and f1 score on the test set.
The value of k with the highest f1 score for each algorithm will be chosen.
Parameters for Top N LOF:
C = 0.00125 (found from the vanilla attempt)
D = 18
N = Number of anomalous points in training / test set
R = Determined from training / test set





The plot on the left shows the tuning process for Top N LOF while the plot on the right shows the tuning process for sklearn. Both plots are largely similar since the LOF calculations for both algorithms are the same.X-axis is the value of k while y-axis is the f1 score for the training set. Optimal k for Top N LOF is found to be 110 while optimal k for sklearn is 70.

Top N LOF with r =5.4, c = 0.00125, n =1355, k = 110, d = 18	Sklearn LOF with contamination parameter set as the percentage of anomalous points and k set as 70
Runtime: 12 seconds F1 Score: 27.3%	Runtime: 0.7 second F1 Score: 31.1%

Across all 3 real-world datasets, for both the vanilla attempt and after tuning k, sklearn has consistently performed better and faster than top n lof.

11.3 Alternative approach to hyperparameter tuning

An alternative approach to tune k and c together is bayesian optimisation. The best hyperparameters will be found in the training set and applied on the test set for both Top N LOF and Sklearn separately. The bayesian optimisation process will have 1000 iterations. The hyperparameters tuned for Top N LOF and sklearn will be different. The algorithms will be compared by f1 score and runtime.

For Top N LOF, optimal k is 51 and optimal c is 0.158. For sklearn, optimal k is 26

Top N LOF with r =5.4, c = 0.158, n =1355, k = 51, d = 18	Sklearn LOF with contamination parameter set as the percentage of anomalous points, k set as 26
Runtime: 1 second F1 Score: 32%	Runtime: 0.6 second F1 Score: 33.2%

Disadvantage to tuning Top N LOF: No clusters or points are pruned. The hyperparameters k and c have to be tuned while ensuring that there are clusters pruned.
If the dataset is very large, the data can be loaded in chunks using the chunksize parameter in pd.read_csv and fit to Sklearn LOF incrementally. However, the results will be poorer.
The current code requires a full load of the data so the chunking process cannot be replicated in Top N LOF. Synthetic data needs to be able to be loaded in RAM and those large real world datasets contain image data which is not the correct input for Top N LOF.
Java buffered reader is used for processing large files since it reads the file line by line.

12. Comparing runtime for very large datasets

```
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
        at java.base/jdk.internal.math.FDBigInteger.<init>(FDBigInteger.java:222)
        at java.base/jdk.internal.math.FloatingDecimal$ASCIIToBinaryBuffer.floatValue(FloatingDecimal.java:1554)
        at java.base/jdk.internal.math.FloatingDecimal.parseFloat(FloatingDecimal.java:122)
        at java.base/java.lang.Float.parseFloat(Float.java:555)
        at microcluster.preprocessing.BuildCFTree.startTrainingCFTree(BuildCFTree.java:38)
        at microcluster.topnlof.TopNLOFDetection.preprocessing(TopNLOFDetection.java:68)
        at microcluster.topnlof.TopNLOFDetection.main(TopNLOFDetection.java:243)
```

Top N LOF is still restricted by the amount of RAM that you have, as shown above.(https://stackoverflow.com/questions/73499864/how-to-allocate-more-memory-java-lang-outofmemoryerror-java-heap-space)

For the experiments below, all datasets have 2 dimensions. The runtime of the algorithms will be compared.
For Top N LOF,
N = number of anomalous points in dataset
K = 20
D = 2
C = 0.00125
For sklearn, k = 20

Size of dataset	Top N LOF	Sklearn LOF
1,000,000 rows	119.5 seconds	7.1 seconds
2,000,000 rows	230.1 seconds	16.4 seconds
3,000,000 rows	409.9 seconds	28.3 seconds
4,000,000 rows	534.3 seconds	37.3 seconds
5,000,000 rows	691.2 seconds	48.9 seconds
6,000,000 rows	876.7 seconds	62.2 seconds
7,000,000 rows	1022.3 seconds	76.5 seconds
8,000,000 rows	1270.1 seconds	79 seconds
14,000,000 rows	1241 seconds	161.1 seconds
23,000,000 rows	1726.2 seconds	265.8 seconds
60,000,000 rows	Memory Error	888.8 seconds
70,000,000 rows	Memory Error	1053.7 seconds (but vs code will crash)
80,000,000 rows	Memory Error	1301.7 seconds (but vs code will crash)
90,000,000 rows	Memory Error	Memory Error
100,000,000 rows	Memory Error	Memory Error

It is to note that Top N LOF takes longer than Sklearn for those datasets that are small enough for both algorithms to run.

```
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
        at java.base/java.lang.Float.valueOf(Float.java:537)
        at microcluster.metricobject.MetricObject.MaxDistToMicroClustersNoLocal(MetricObject.java:329)
        at microcluster.metricobject.MicroCluster.computeKdistanceBound(MicroCluster.java:135)
        at microcluster.topnlof.ComputeLOFBound.ComputeKDistanceBoundForMCs(ComputeLOFBound.java:70)
        at microcluster.topnlof.TopNLOFDetection.computeLOFBound(TopNLOFDetection.java:84)
        at microcluster.topnlof.TopNLOFDetection.main(TopNLOFDetection.java:247)
```

The above error could occur either during the building of the tree or after processing. For larger datasets, the error occurred during the building of the tree. For smaller datasets, the error occurred after processing.

13. Sklearn clustering + LOF

Instead of using Top N LOF in Java, clustering will be done followed by LOF. Both steps will be implemented using sklearn packages.

The motivation of this experiment is the long runtime of a single sklearn LOF. It is hypothesised that by performing clustering first and then sklearn LOF on each cluster, there will be huge savings in the runtime.

The objective of this experiment is to compare the runtime and accuracy of the aforementioned 2 step process against the 1 step process of sklearn LOF.

13.1 60 million rows of synthetic dataset

Anomalies will be added to this synthetic dataset.

13.1.1 Global Outliers

How data is generated: The dataset will be split into generating sets of 10000 points. Referring to Experiment 8, every 10000 points will have the same 2 clusters with outliers pattern. For example, the first set of 10000 points will have the 2 clusters with means x1 and x2. The next set of 10000 points will have the 2 clusters with means x1 + 50 and x2 + 50 (translation of the clusters). This is to ensure that all clusters will still be distinct from 1 another.

By setting the number of clusters to be 20, naive K-means took 757.9 seconds. Mini batch K-means took only 93.1 seconds. The difference between the 2 algorithms is that Mini batch K-means takes a subset of the dataset for each iteration instead of the entire dataset, thus reducing runtime.
Replicating BIRCH clustering in python gives Memory Error.
As such, the chosen clustering algorithm (for the subsequent experiments as well) will be Mini batch K-means. The percentage of anomalies is 0.1%.
Ground truth of the number of clusters cannot be used. For naive k-means clustering, the runtime is too long. For mini-batch k means, some clusters will only have 1 point, causing errors when doing LOF for those clusters later (no neighbor). The number of clusters is set as 50.

The contamination parameter for the LOF for each cluster will be set as 0.1%.

Metric	2 step process (Clustering + Sklearn LOF)	Sklearn LOF
Runtime	Clustering: 32.7 seconds LOF: 349.2 seconds Total: **381.9 seconds**	**317.5 seconds**
F1 Score	74.7%	74.7%

Setting k=20 for mini batch k-means will give the same f1 score as a single pass of LOF, though predictions are not exactly the same. Total runtime is slightly slower for the 2 step process as compared to a single pass of LOF.

13.1.2 Local Outliers

How data is generated: The dataset will be split into generating sets of 10000 points. Referring to Experiment 8, every 10000 points will have the same 2 clusters with outliers pattern. For example, the first set of 10000 points will have the 2 clusters with means x1 and x2. The next set of 10000 points will have the 2 clusters with means x1 + 50 and x2 + 50 (translation of the clusters). This is to ensure that all clusters will still be distinct from 1 another.

The chosen clustering algorithm will be Mini Batch K-means. The percentage of anomalies is 0.1%. The number of clusters will be set as 50.

The contamination parameter for the LOF for each cluster will be set as 0.1%.

Metric	2 step process (Clustering + Sklearn LOF)	Sklearn LOF
Runtime	Clustering: 83.7 seconds LOF: 246.6 seconds Total: **330.3 seconds**	**291.4 seconds**
F1 Score	95.7%	95.6%

Setting k=20 for mini batch k-means will give the same f1 score as a single pass of LOF, though predictions are not exactly the same. Total runtime is slightly slower for the 2 step process as compared to a single pass of LOF.

The next few subsections will compare the 2 step process and a single pass of LOF against the real world datasets. Referring to Experiments 9, 10 & 11, the contamination parameter will be based on the percentage of anomalies in the training/test sets (does not matter since it is a stratified split). The algorithms will be compared by runtime and f1 score on the test set. The clustering algorithm will be Mini batch k-means.

13.2 ML Matt dataset

The number of clusters is set as 50.

Metric	2 step process (Clustering + Sklearn LOF)	Sklearn LOF
Runtime	Clustering: 0.2 second LOF: 0.1 second Total: **0.3 second**	**2.8 seconds**
F1 Score	28.3%	28.5%

13.3 Access behavior anomaly dataset

The number of clusters is set as 50.

Metric	2 step process (Clustering + Sklearn LOF)	Sklearn LOF
Runtime	Clustering: 0.2 second LOF: < 0.1 second Total: **< 0.3 second**	**< 0.1 second**
F1 Score	32.8%	28.2%

13.4 Energy

The number of clusters is set as 50.

Metric	2 step process (Clustering + Sklearn LOF)	Sklearn LOF
Runtime	Clustering: 0.2 second LOF: 0.1 second Total: **0.3 second**	**0.6 second**
F1 Score	32.3%	34.3%

From the experiments done above, it is inconclusive to determine if the runtime for the 2 step process is slower than a single pass of LOF. F1 scores are largely similar. A single pass of LOF is preferred as there are less codes to run.

13.5 Increasing size of synthetic dataset

From Experiment 13.1, it has been noted that there might be room for a larger synthetic dataset. This experiment is to test the largest dataset that python can handle. The parameters follow Experiment 13.1.

Global Outliers

Size of dataset	2 step process (Clustering + Sklearn LOF)	Sklearn LOF
70 million rows	Runtime: 95.9 seconds + 376.9 seconds = **472.8 seconds** F1 Score: 74.6%	Runtime: **353.9 seconds** F1 Score: 74.6%
80 million rows	Runtime: 95.9 seconds + 376.9 seconds = **472.8 seconds** F1 Score: 74.6%	Runtime: **353.9 seconds** F1 Score: 74.6%