Technical University of Munich

Informatics 10 – Chair of Computer Architecture and Parallel Systems (Prof. Schulz)

Master Praktikum: IoT (Internet of Things) (IN2106, IN4224)

MILESTONE 2

Anomalies detection, report

Group: 2

Students and matriculation number:

Nadija Borovina (03742897)

Nedžad Hadžiosmanović (03742896)

Lecturers:

Gerndt, Hans Michael, Prof. Dr.

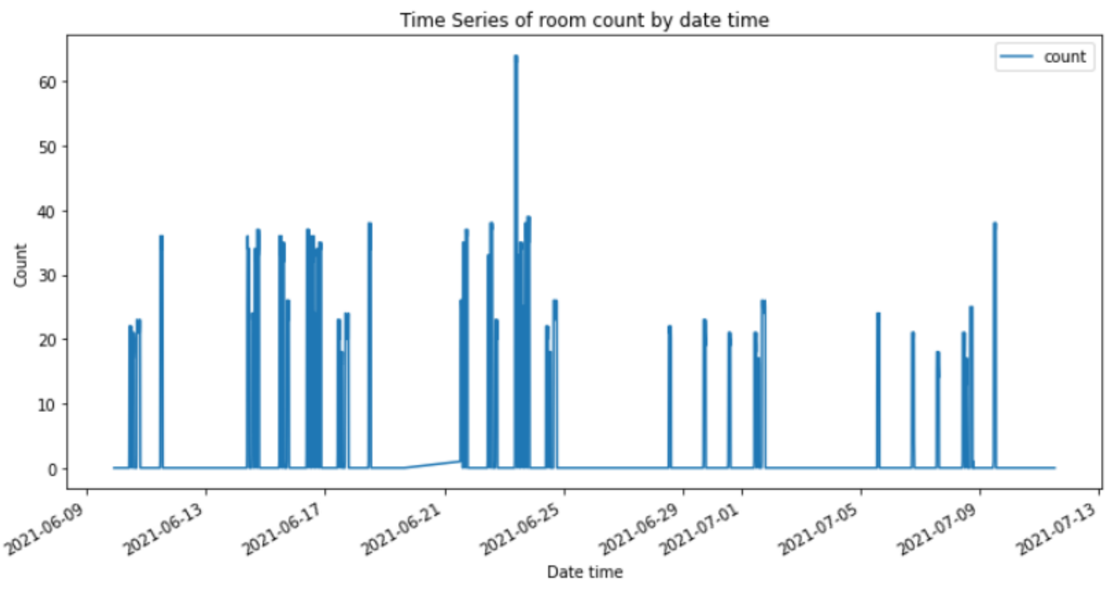
Podolski Vladimir, M. Sc.

Chenyuan Zhao

Munich, July 2021.

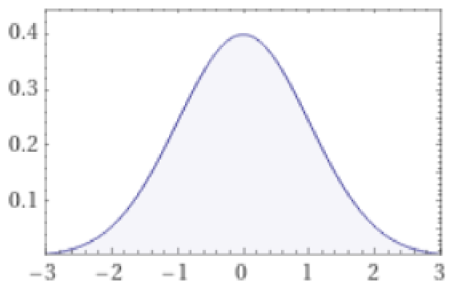
1. Plotting the data

As a first step to anomaly detection, we plotted our data (all we had, data since June, since as already mentioned on one of our meetings, data from before that were somehow deleted from our sensor). We can already see a potential outliers and anomalies that occurred during this time span, for example on 23. of June where we had more than 60 people in the room, while max allowed was 45.

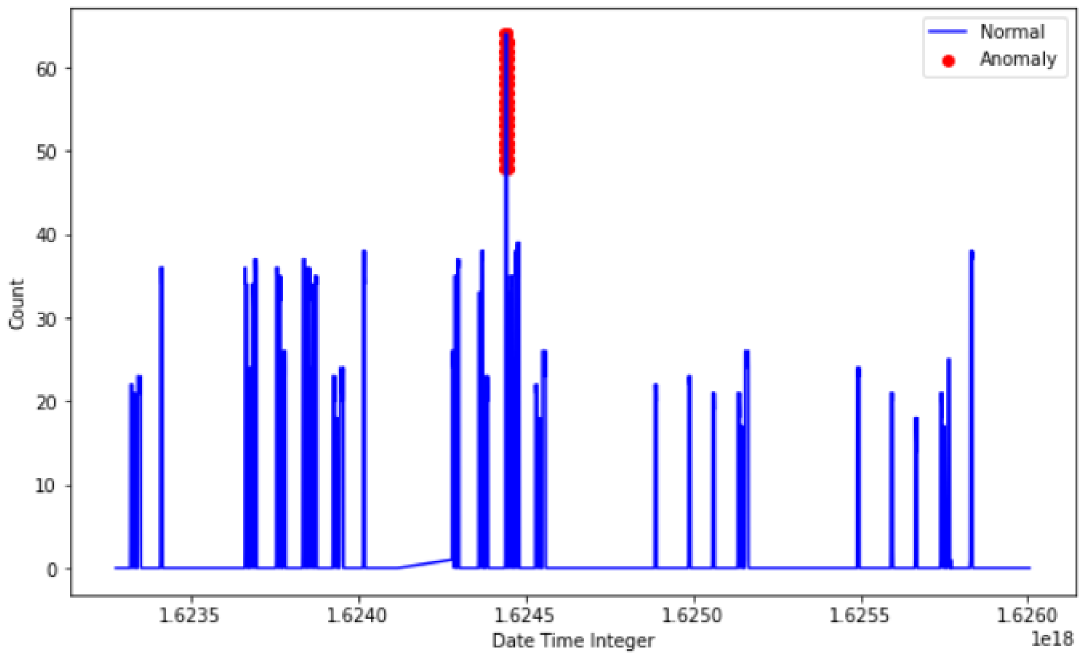


2. Z-score

After visualising our data, we first decided to use one of the simplest algorithms for anomaly detection, which would be z-score. Z-score tells how many standard deviations away a given observation is from the mean. It's a simple statistics that uses variance and mean. For example, if a z-score is equal to +1, it is 1 standard deviation above the mean, and since it standardizes the values (for which is also known as standard score) by looking at standard normal distribution graph, we can see that reasonable values for threshold should be between 2 and 3:

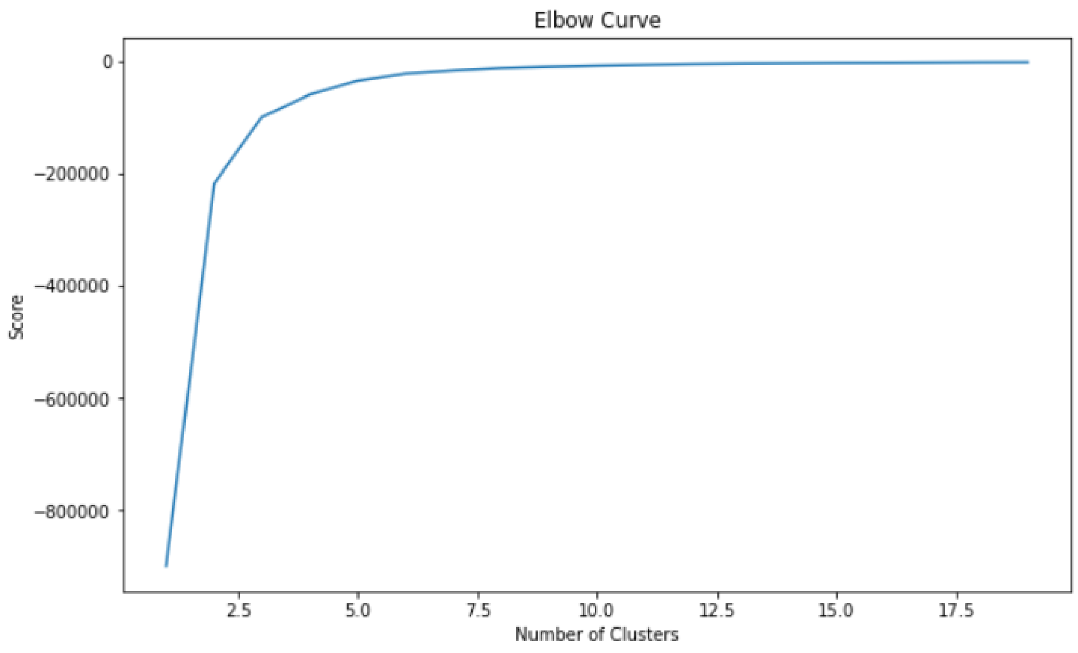


For our threshold we used 3 as it means that for any value that has z-score higher than that, will be considered anomaly. For such settings, we get the results as in the next graph. Our results we stored in a separate column 'anomaly' of our dataframe, which will be used later.

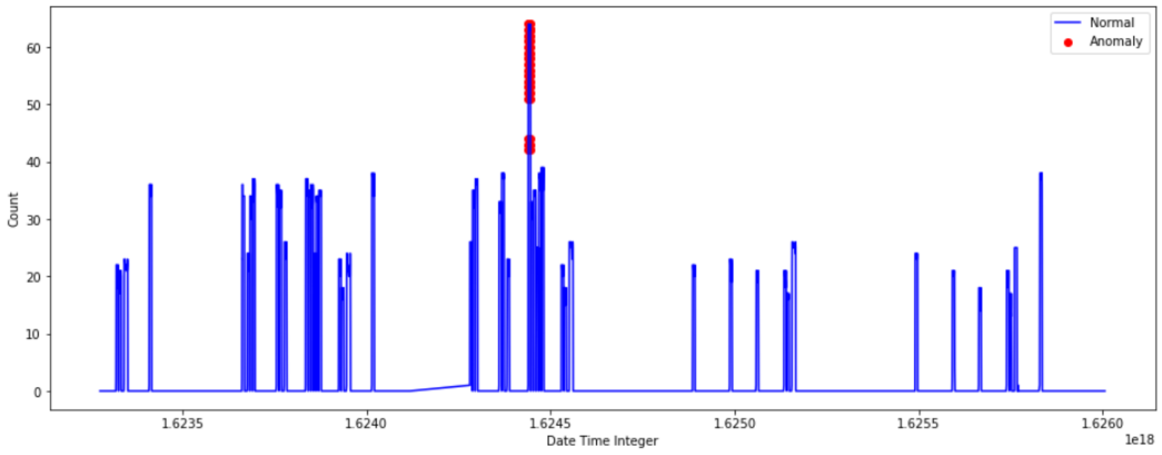


3. K-means

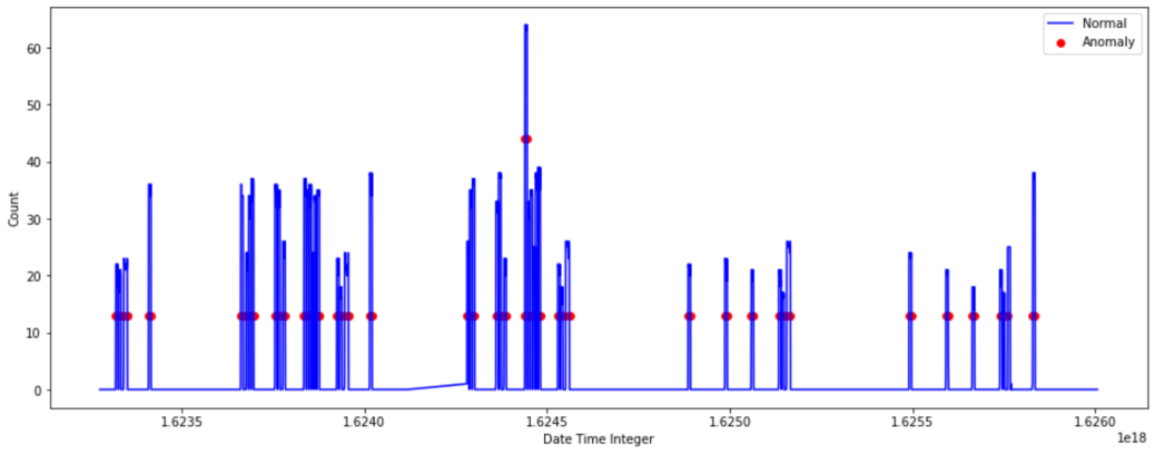
After calculating z-score, we decided to try another method called k-means. K-means is cluster based anomaly detection. It creates ‘k’ similar clusters of data points. Data instances that fall outside of these groups are marked as an anomalies. Before we starting k-means clustering, we use elbow curve to determine the optimal number of clusters.



We can see that the graph levels off after 5 clusters, which means that adding more clusters won’t explain much more of the variance in data. To proceed with our anomaly detection using k-means, we first must set a few parameter values such as outliers fraction and threshold. Outliers fraction specifies the proportion of the outliers in our data. We set out outlier fraction to be percentage of the observations that fall over the abs value of 3 in z-score, which we previously intentionally calculated and which is usually a practice. Our threshold we set to the minimum distance of these outliers. After plotting we get the result as in picture:

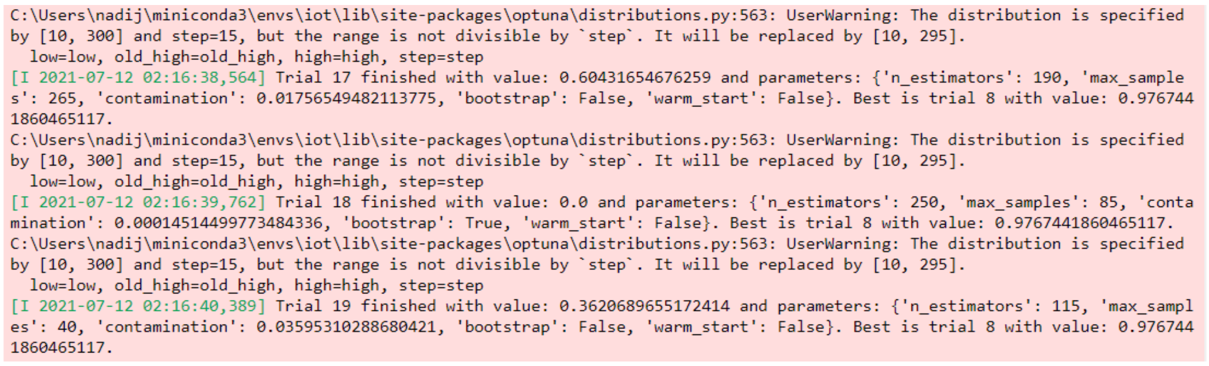


As we can see, we get indeed an interesting result, very different from z-score. After rerunning our notebook, we realised that with each run and same parameters, we obtain different results, which can be seen in the next picture. This is due to k-means instability, namely, with each run we obtain the same number of different clusters due to different centroid initialization.



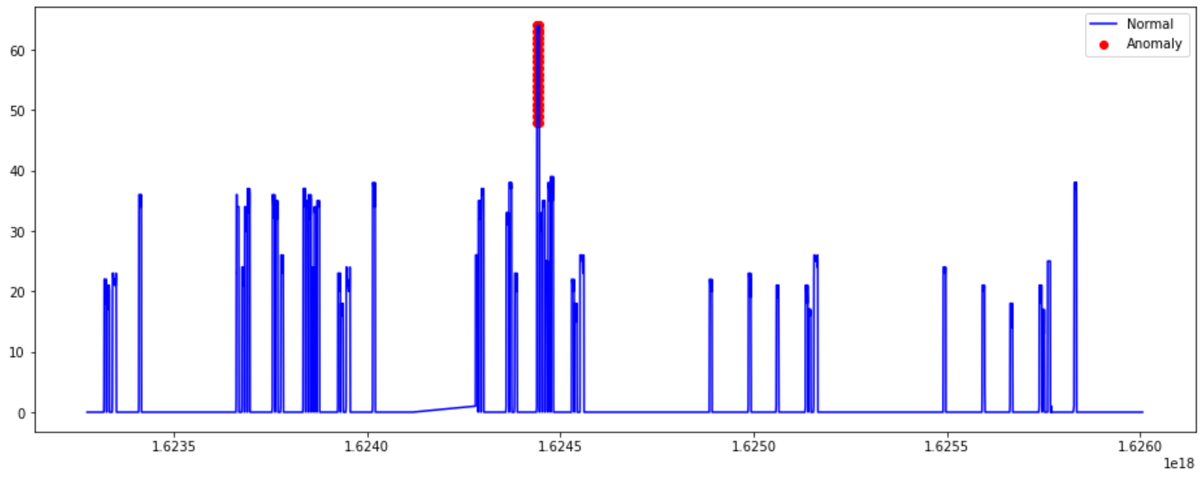
4. Isolation Forest

After k-means, we decided to try another approach, isolation forest. For this purpose, to find best parameter we decided to use Optuna, which is hyperparameter optimization framework. We specified some of the parameter ranges and values and with optuna evaluated 20 of their combinations. To use optuna, data must be labeled, so that we can compare real and predicted anomalies and based on that give score to each model. To calculate model score we used F1 = 2 * (precision * recall) / (precision + recall), and as labels used “anomaly” column already mentioned in z-score chapter, since labelling this amount of data manually would be extremely time consuming and anomalies just on z-score seemed pretty accurate (if we were to discard values bigger than 45 and less than 0). Optuna trials can be seen on the next photo:



```
C:\Users\nadij\miniconda3\envs\iot\lib\site-packages\optuna\distributions.py:563: UserWarning: The distribution is specified
by [10, 300] and step=15, but the range is not divisible by `step`. It will be replaced by [10, 295].
  low=low, old_high=old_high, high=high, step=step
[I 2021-07-12 02:16:38,564] Trial 17 finished with value: 0.60431654676259 and parameters: {'n_estimators': 190, 'max_sample
s': 265, 'contamination': 0.01756549482113775, 'bootstrap': False, 'warm_start': False}. Best is trial 8 with value: 0.976744
1860465117.
C:\Users\nadij\miniconda3\envs\iot\lib\site-packages\optuna\distributions.py:563: UserWarning: The distribution is specified
by [10, 300] and step=15, but the range is not divisible by `step`. It will be replaced by [10, 295].
  low=low, old_high=old_high, high=high, step=step
[I 2021-07-12 02:16:39,762] Trial 18 finished with value: 0.0 and parameters: {'n_estimators': 250, 'max_samples': 85, 'conta
mination': 0.00014514499773484336, 'bootstrap': True, 'warm_start': False}. Best is trial 8 with value: 0.9767441860465117.
C:\Users\nadij\miniconda3\envs\iot\lib\site-packages\optuna\distributions.py:563: UserWarning: The distribution is specified
by [10, 300] and step=15, but the range is not divisible by `step`. It will be replaced by [10, 295].
  low=low, old_high=old_high, high=high, step=step
[I 2021-07-12 02:16:40,389] Trial 19 finished with value: 0.3620689655172414 and parameters: {'n_estimators': 115, 'max_sampl
es': 40, 'contamination': 0.03595310288680421, 'bootstrap': False, 'warm_start': False}. Best is trial 8 with value: 0.976744
1860465117.
```

And the results:



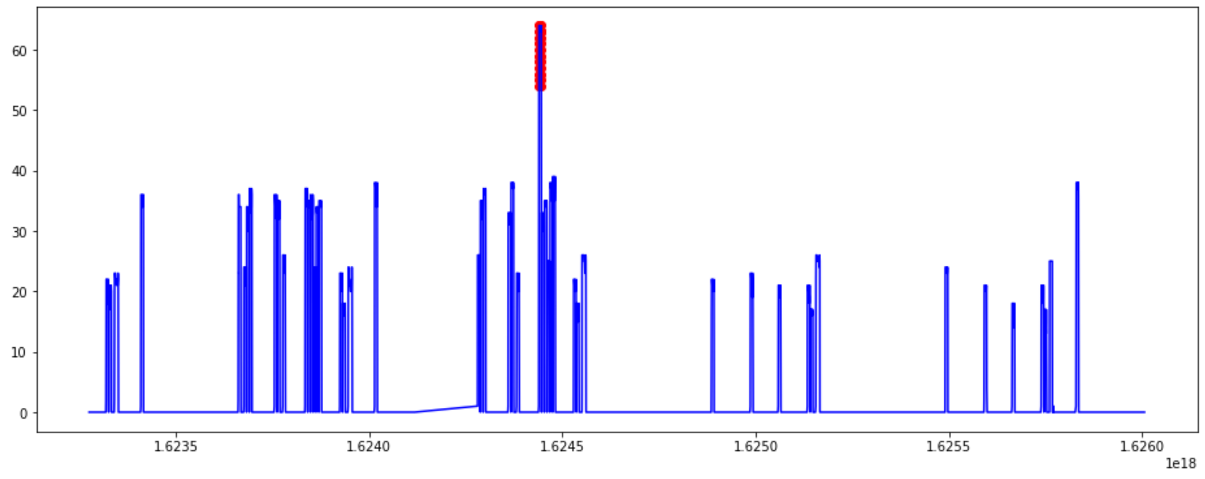
Now, the best trial is the 8th: `Trial 8 finished with value: 0.9767441860465117 and parameters: {'n_estimators': 130, 'max_samples': 130, 'contamination': 0.007471064343230549, 'bootstrap': False, 'warm_start': False}. Best is trial 8 with value: 0.9767441860465117.`

Number of estimators in fact represents number of isolation trees in isolation forest. Max samples is the number of random samples it will pick from the original dataset for creating isolation trees and another very important parameter would be contamination. As we can see it is very close to outliers fraction we previously calculated in z-score (0.007282816022195249).

5. SVM

Lastly, we used one of Support Vector Machine-based anomaly detection and that is OneClassSVM. SVM is usually associated with supervised learning, but OneClassSVM can be used to identify anomalies as an unsupervised problems. The idea of OneClassSVM is finding a function that is positive for regions with high density of points, and negative for small densities.

To use OneClassSVM we had to choose a few parameters. One of them being outliers fraction, which we set to already calculated fraction using z-score. Then, we had to choose kernel, and for that we picked most commonly used, non-linear one – radial basis function (RBF). Third parameter we had to set was gamma. Gamma parameter of RBF controls the distance of the influence of a single training point. Low values of gamma indicate a large similarity radius and results in more points grouped together. For high values of gamma, the points need to be very close to each other to be considered in the same group. Therefore, models with very large gamma values tend to overfit, so we set our gamma to a very small value and the results we get can be seen in the next picture:



From all the above, we can say, that isolation forest worked the best in our case (besides z-score, which we used for further calculations and labeling) and almost perfectly predicted the labeled anomalies, with accuracy of almost 0.98, therefore we used isolation forest as our final model, and pushed anomalies it detected.