Analytics

Forecasting

Model 1: Linear regression

Model 2: Long short-term memory

Overview

- implemented with PyTorch
- provides training and prediction

Hyperparameters

- number of features: 5
  - lag1_count: predecessor count of the current datapoint
  - dT: time between predecessor datapoint and current datapoint
  - minute of day: [0, 1339]
  - day of week: [0, 6]
  - month of year: [0, 11]
  - Note: all data will be scaled to [0, 1]
- number of layers: 2
  - two-stacked LSTM based on recommendations in the internet
- learning rate: 0.0001
  - found by probing
- batch size: 32
  - found by probing
- sequence length: 16
  - important parameter to indicate how many data points the LSTM must look back to make conclusion for current value
  - if too low, the LSTM cannot identify patterns
  - if too high, computation effort gets very high
- number of epochs: 20
  - important parameter that defines the number of training rounds
  - with 20 we get a model that fits to the training dataset well (could be even a bit higher with the tradeoff of needing longer timer and more computation effort)

Add Training Result Diagram?

Model 3: SARIMAX

Model Selection for Forecasting

- all three models are used on the edge to forecast values
- for bestOnline forecasting we use the strategy of “Selecting the Most Accurate” using the accuracy values to find the winner model and send its forecast also to the ESP device
- “Majority Rule” is implemented as well but only tested and not used in production

Model Comparison:

TODO: Add kibana visualization here next Tuesday → linreg should be best

TODO: Add accuracy values → visualize first in kibana

Linear Regression	Long short-term memory	SARIMAX
- fast and lightweight training - easy to implement	- heavy training - extremely flexible - helpful documentation in the internet	- heavy training

Conclusion: In the context of predicting the value in fifteen minutes, linear regression is the preferred one. It achieved best results in the forecasting. Also it was the only one we could train on the IoT platform in the FaaS environment.

Anomaly Detection in Time Series Data

Used dataset

- the data was fetched from sensor 1276 (Index: 48_122_1276)
- the sensor is used as a receiver of the virtualized student room events
- recorded data goes from 1622805315 to 1624644558
  → *4th June 13:15:15* to *25th June 20:09:18* (MESZ)

	t	count	hour_of_day	day_of_week	month_of_year
count	4.794000e+03	4794.000000	4794.000000	4794.000000	4794.0
mean	1.623776e+09	14.295160	13.833751	1.993951	6.0
std	5.119805e+05	12.975232	4.664959	1.603565	0.0
min	1.622805e+09	0.000000	0.000000	0.000000	6.0
25%	1.623258e+09	1.000000	11.000000	1.000000	6.0
50%	1.623791e+09	13.000000	14.000000	2.000000	6.0
75%	1.624287e+09	24.000000	17.000000	3.000000	6.0
max	1.624645e+09	64.000000	23.000000	6.000000	6.0

Data table with 4794 data points gives rough idea over their time-related distribution, i.e. in which time frames happen most of the events

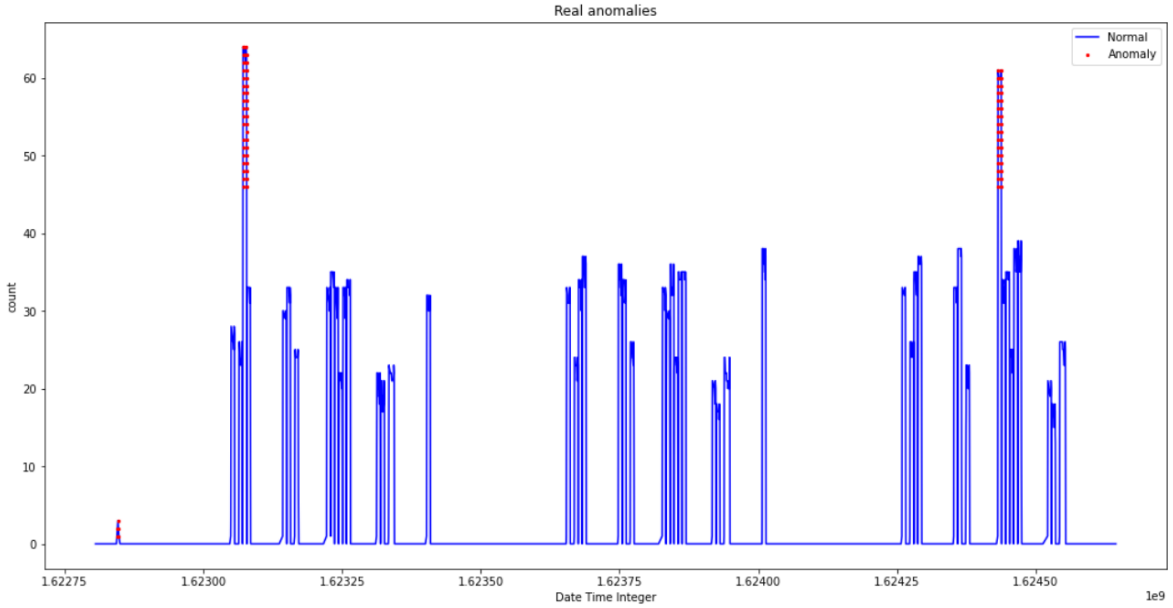
- for the weekly schedule we see a tendency towards within the week: counting events happen mainly during the week and counts play a less significant role
- for the daily schedule we see a tendency towards late morning until afternoon, whereas events in the early morning and in the evening have a smaller share

Basic anomaly detection and visualization of the dataset

To label a data point as an anomaly in the dataset (at least) one of the two conditions must hold

- any count bigger than 0 at the weekends is an anomaly
- any count over 45 is an anomaly

Applying these conditions on our dataset, in which we artificially add some >0 counts on the first weekend, we get the following line diagram:



Interpretation:

- we see the artificial weekend anomaly as a first red dotted peak at the beginning
- two high peaks are red dotted since they exceed the maximum room count of 45 → these events are real anomalies in this time period
- we can compute the **outliers fraction θ** which is necessary in the following detection methods

KMeans based anomaly detection

General

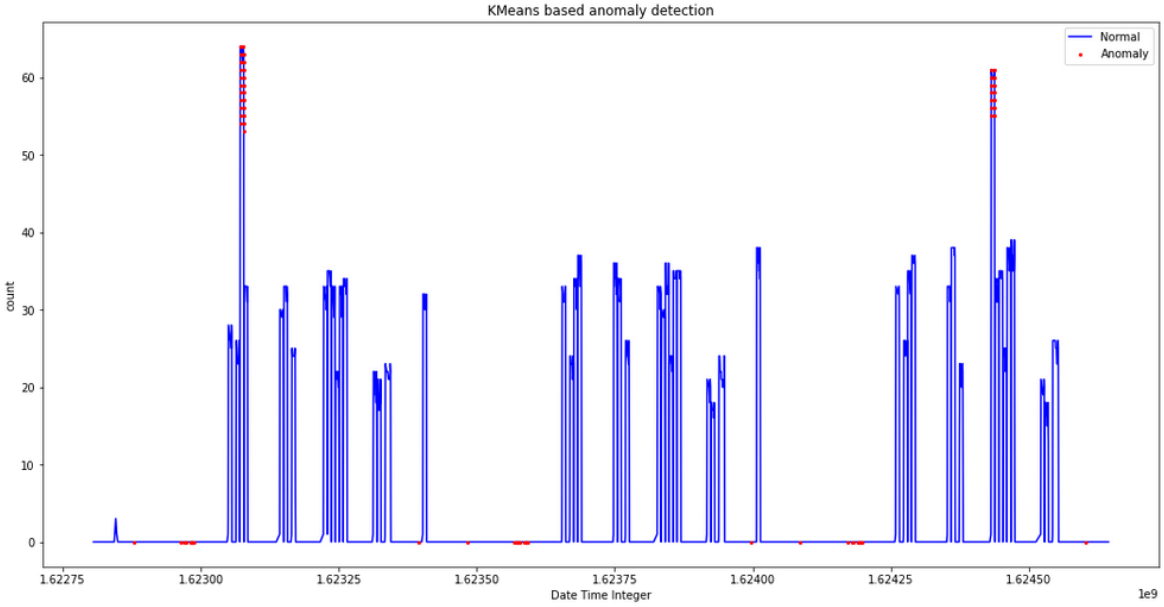
- generate clusters: each datapoint is assigned to the cluster with the nearest centroid
- compute distance between each point and its nearest centroid
- compute outlier threshold by using the outliers fraction **θ** → distances bigger than threshold are considered as anomalies

Parameterization

- external parameters
  - outliers fraction **θ**
- internal parameters
  - number of clusters
    → using elbow curve to find best number
    - run k-means multiple times with an increasing number of clusters
    - sum of squared distances from each point to centroid as objective function
    - take cluster number that is “bend of elbow”

Result

- based on 4 clusters



Interpretation

- KMeans is able to detect the room count anomaly when threshold of 45 is exceeded. However, not as precise as in the basic visualization, as we see some false negatives that exceed the threshold but are not detected as anomalies.
- False positives are dotted that come in some kind of schedule. Apparently, KMeans expects students in the room for these timestamps.

IsolationForest based anomaly detection

General

- each data point is finally isolated on the isolation tree and has a path beginning at the tree's root
- if the path for a data point is long the point is regarded as non-anomalous, if the path is short it was easy to isolate the point, thus the point is regarded as anomalous

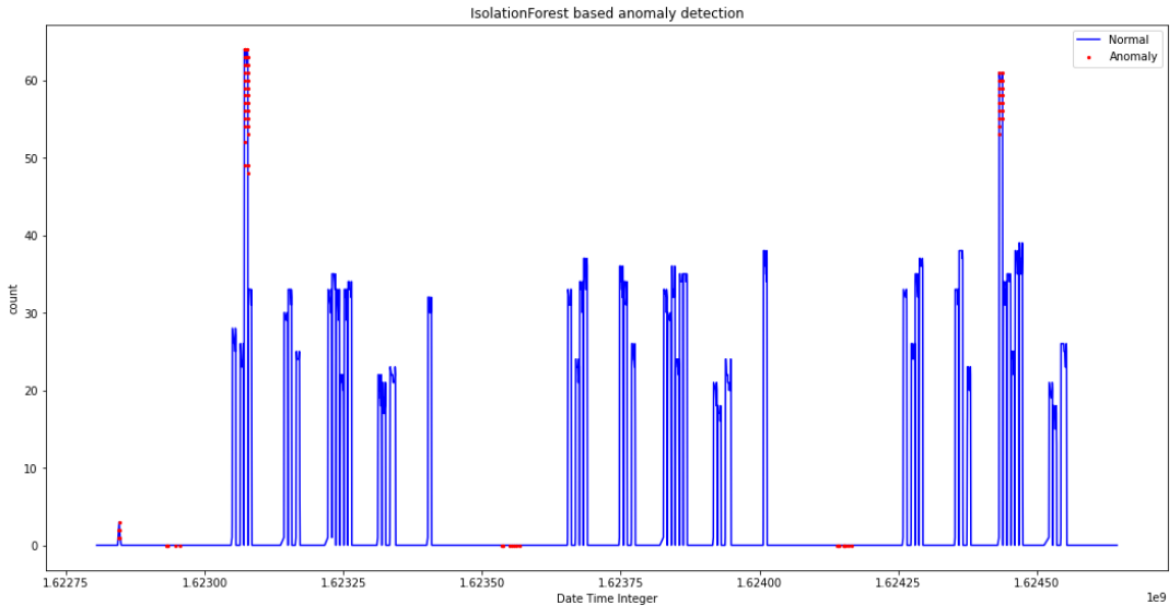
Parameterization

- external parameters
  - Outlier fraction θ
- internal parameters
  - #estimators
  - max samples
  - max features
  - bootstrap
  - warm start

  → parameter values are found by doing a parameter study using Optuna

Result

- based on: {'n_estimators': 100, 'max_samples': 0.5932156212744166, 'max_features': 0.7148931876176408, 'bootstrap': True, 'warm_start': True}



Interpretation

- IsolationForest is able to detect the room count anomaly when threshold of 45 is exceeded. We see some false negatives that exceed the threshold but are not detected as anomalies. Performs better than KMeans.
- IsolationForest is able to detect the anomaly of students in the room on the first weekend.
- False positives are dotted that come in some kind of schedule. Apparently, IsolationForest expects students in the room for these timestamps.

SVM based anomaly detection

General

- uses the concept of support vector machine: separate data points into classes by computing the linear separator with the largest margin possible
- use a one class support vector machine to find a function that is able to separate points in those lying in the area of high density and those lying outside of it → data points of the last are regarded as anomalies

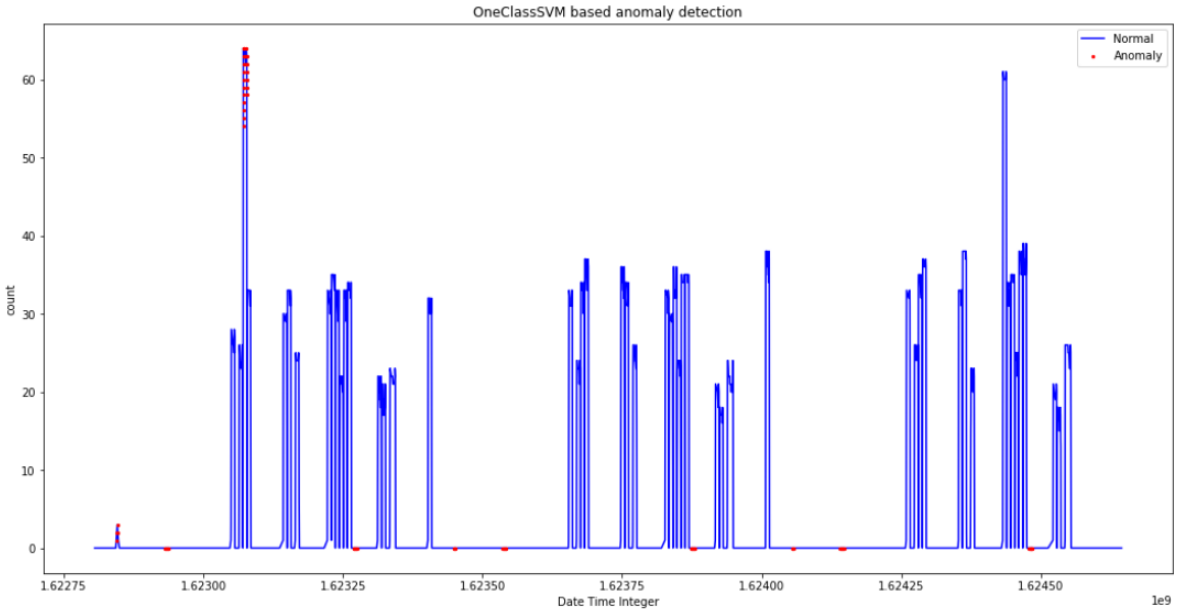
Parameterization

- external parameters
  - outliers fraction θ
- internal parameters
  - kernel: kernel type to transform the data
    - Selection: linear, polynomial, sigmoid, radial basis function
  - degree (if kernel=polynomial): Range [1, 5]
  - shrinking:

→ parameter values are found by doing a parameter study using Optuna

Result

- based on 'kernel': 'sigmoid', 'shrinking': False



Interpretation

- detects the first weekend with students as anomaly correctly
- detects the first peek that exceeds the threshold of 45 as anomaly, however with some false negatives similar to the other approaches
- does not detect the second peek that exceeds the threshold as anomaly

Comparison

Anomaly Detection Method	Advantages	Limitations
KMeans	- easy to understand - #clusters as single internal variable - easy to compute for single feature data	-
IsolationForest	- no distance or density computations → tree can be computed fast - low memory footprint: total number of tree nodes is (n + n -1) = 2n -1 where n is the number of data points → tree grows linearly with number of data points	- knowledge about proportion of outliers crucial for training
OneClassSVM		

Quick note on anomaly detection on virtual testbed

- because of evaluation results and performance advantages we use IsolationForest
- IsolationForest model can be trained using FaaS on the IoT platform or doing it locally
- anomaly detector in the edge uses the model and periodically evaluates the virtual room count data for anomalies

Sources:

https://towardsdatascience.com/time-series-of-price-anomaly-detection-13586cd5ff46
https://towardsdatascience.com/clustering-metrics-better-than-the-elbow-method-6926e1f723a6
https://towardsdatascience.com/isolation-forest-is-the-best-anomaly-detection-algorithm-for-big-data-right-now-e1a18ec0f94f