Anomaly Detection

Davin Ahn

November 04, 2022

Inhalt

Ausreißererkennung durch Isolation Forest

Hyperparameter einsetzen

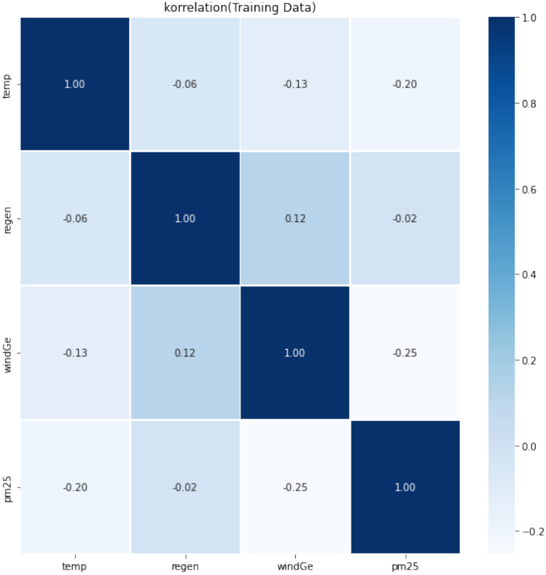
```python
def iForest(ds, n_estimators = 100, max_samples = "auto", contamination = 'auto', max_features = 1.0, random_state = None):

    clf = IsolationForest(n_estimators = n_estimators, max_samples = max_samples,
                          contamination = contamination, max_features = max_features, random_state = random_state)
```

```python
train = iForest(data_train, contamination = float(0.004), random_state = 42)
```

Ausreißererkennung durch Isolation Forest

Hyperparameter einsetzen



Ausreißererkennung durch Isolation Forest

Trainierung

```python
def iForest(ds, n_estimators = 100, max_samples = "auto", contamination = 'auto', max_features = 1.0, random_state = None):

    clf = IsolationForest(n_estimators = n_estimators, max_samples = max_samples,
                          contamination = contamination, max_features = max_features, random_state = random_state)

    # Entfernen die Daten, die keine Features sind
    dataset = ds.drop(columns = ['timestamp', 'sensor'])
    # Algorithmus ausführen
    clf.fit(dataset)
    # Ausreißer erkennen, danach auf jeden Datenpunkt zeichnen
    dataset['anomaly'] = clf.predict(dataset)
    # filtern die Daten, die als Ausreißer gezeigt sind
    outliers = dataset.loc[dataset['anomaly'] == -1]
    # Index von oben gefilterten Daten als Liste speichern
    outlier_index = list(outliers.index)
    # Anzahl der Anomalien und normalen Daten. Daten, die mit -1 klassifiziert sind, sind anomal
    print(dataset['anomaly'].value_counts())
```

```
 1    8482
-1      35
Name: anomaly, dtype: int64
```

Ausreißererkennung durch Isolation Forest

```
#-----Visualization in 3D-----#
# Normalization die Daten
scaler = StandardScaler()
data_scaled = scaler.fit_transform(dataset)

# Reduktion hochdimensionaler Daten in niedrigdimensionale Daten durch PCA(Principal Component Analysis)
pca = PCA(n_components = 3)
data_reduce = pca.fit_transform(data_scaled)

fig = plt.figure()
ax = fig.add_subplot(projection='3d')

# Plot the compressed data points
ax.scatter(data_reduce[:, 0], data_reduce[:, 1], zs = data_reduce[:, 2], s = 4, lw = 1, label = "inliers", c = "green")

# Plot x's for the ground truth outliers
ax.scatter(data_reduce[outlier_index,0], data_reduce[outlier_index,1], data_reduce[outlier_index,2],
           lw = 2, s = 60, marker = "x", c = "red", label = "outliers")
ax.legend()
plt.show()
#----------------------------#

#-----Visualization in 2D-----#
pca = PCA(2)

#Reduction of high-dimensional data into low-dimensional data
data_reduce = pca.fit_transform(data_scaled)

res = pd.DataFrame(data_reduce)
b1 = plt.scatter(res[0], res[1], c = 'green', s = 20,label = "normal points")
b1 = plt.scatter(res.iloc[outlier_index,0], res.iloc[outlier_index,1], c = 'green', s = 20,
                edgecolor = "red",label = "predicted outliers")
plt.legend(loc = "lower right")
plt.show()
#----------------------------#
```

Ausreißererkennung durch Isolation Forest

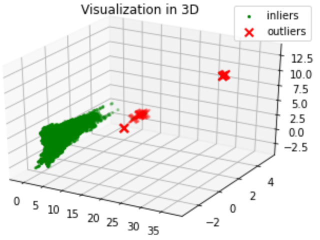
```
# entfernen die Ausreißer, danach das Index initialisieren, danach die column 'index' entfernen
returnData = ds.drop(index=outlier_index).reset_index().drop(columns = ['index'], axis = 1)
```

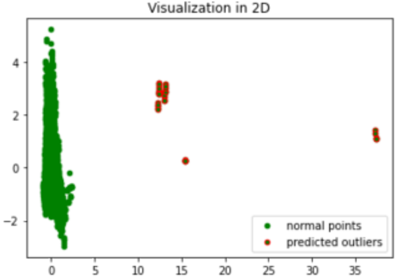
Ausreißererkennung durch Isolation Forest

Resultat

```
1 train = iForest(data_train, contamination = float(0.004), random_state = 42)
```

```
 1    8482
-1      35
Name: anomaly, dtype: int64
```



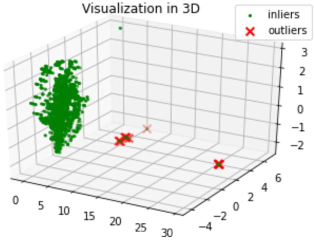


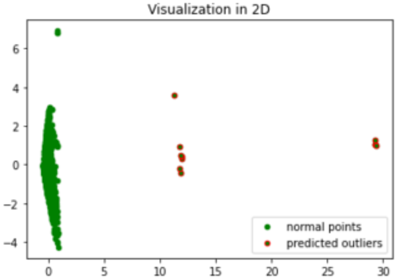
Ausreißererkennung durch Isolation Forest

Resultat

```
test = iForest(data_test, contamination = float(0.004), random_state = 42)
```

```
 1    2725
-1      11
Name: anomaly, dtype: int64
```





Model Trainieren mit anomalen erkannten Daten

Resultat

```
3/3 [==============================] - 0s 4ms/step
1/1 [==============================] - 0s 19ms/step
DEUS: MSE: 25.15/16.22 RMSE: 5.01/4.03 MAE: 3.67/2.86 SMAPE: 41.45%/48.01% R²: 0.64/0.35 RDE: 33.61% in 3.3s
3/3 [==============================] - 0s 4ms/step
1/1 [==============================] - 0s 20ms/step
215: MSE: 25.58/18.53 RMSE: 5.06/4.30 MAE: 3.62/2.83 SMAPE: 38.29%/38.51% R²: 0.67/0.48 RDE: 12.76% in 2.1s
3/3 [==============================] - 0s 5ms/step
1/1 [==============================] - 0s 26ms/step
218: MSE: 28.46/24.45 RMSE: 5.34/4.94 MAE: 3.78/3.28 SMAPE: 38.88%/42.62% R²: 0.67/0.38 RDE: 16.98% in 1.8s
3/3 [==============================] - 0s 4ms/step
1/1 [==============================] - 0s 17ms/step
216: MSE: 25.62/21.94 RMSE: 5.06/4.68 MAE: 3.63/3.20 SMAPE: 37.81%/40.17% R²: 0.64/0.42 RDE: 14.66% in 1.8s
```

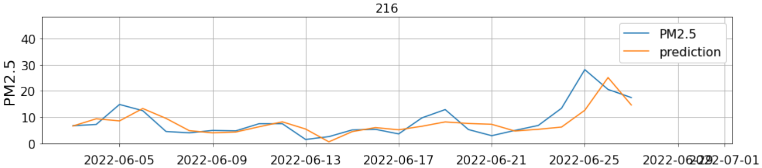
Evaluation mit originalem Datensatz

```
3/3 [==============================] - 0s 5ms/step
1/1 [==============================] - 0s 17ms/step
DEUS: MSE: 22.97/15.35 RMSE: 4.79/3.92 MAE: 3.51/2.75 SMAPE: 40.95%/44.64% R²: 0.67/0.38 RDE: 29.14% in 1.8s
3/3 [==============================] - 0s 4ms/step
1/1 [==============================] - 0s 18ms/step
215: MSE: 23.72/17.90 RMSE: 4.87/4.23 MAE: 3.49/2.75 SMAPE: 38.41%/36.92% R²: 0.70/0.49 RDE: 10.94% in 2.9s
3/3 [==============================] - 0s 4ms/step
1/1 [==============================] - 0s 17ms/step
218: MSE: 25.71/23.60 RMSE: 5.07/4.86 MAE: 3.62/3.19 SMAPE: 39.01%/41.47% R²: 0.70/0.38 RDE: 14.72% in 1.7s
3/3 [==============================] - 0s 5ms/step
1/1 [==============================] - 0s 18ms/step
216: MSE: 23.28/21.16 RMSE: 4.83/4.60 MAE: 3.49/3.12 SMAPE: 37.83%/39.24% R²: 0.67/0.43 RDE: 12.64% in 2.9s
```

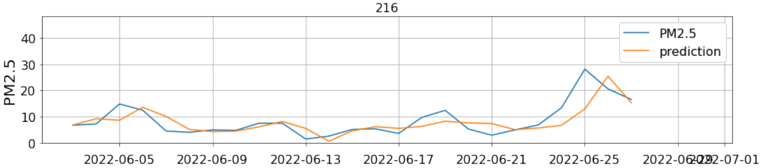
Evaluation mit anomalien erkanntem Datensatz

Model Trainieren mit anomalen erkannten Daten

Resultat



Evaluation mit originalem Datensatz



Evaluation mit anomalien erkanntem Datensatz

Model Trainieren mit anomalen erkannten Daten

Analyse