anomaly_detection

November 13, 2022

0.1 Anomaly detection in time series using ADTK Interquartile Range Methodology and Isolation Forest Technique

Anomaly (or outlier) detection is the data analysis task of detecting instances/observations in a sample that deviate strongly from the norm (e.g. assuming a Normal/Gaussian distribution, an anomaly is any data point residing more than 3 standard deviations from the population mean).

0.1.1 Load libraries

[1]:
```
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rcParams['figure.figsize'] = (20,5)
import seaborn as sns
sns.set_style("whitegrid")

import warnings
warnings.filterwarnings("ignore")
```

0.1.2 Load data

Time series chemical manufacturing data.

Useful for proactive identification of abnormalities in monitored trends. Early detection can enhance process understanding, eliminate downtimes related to malfunctioning equipment, operations' shutdown, and reduce associated costs.

[2]:
```
df = pd.read_csv('../data/chemical_manufacturing_time_series.csv')
df.shape
```

[2]:
```
(2212, 8)
```

[3]:
```
df.head()
```

[3]:
```
  Date (DD/MM/YYYY)  Pressure_Sensor_1  Pressure_Sensor_2  \
0   14/03/2019 4:00          30.702663          29.010453
1   14/03/2019 4:01          30.384615          28.278746
2   14/03/2019 4:02          32.233728          27.993031
```

```
3   14/03/2019 4:03          31.545858         30.919861
4   14/03/2019 4:04          32.233728         30.954704

   Motor_Revolutions_Per_Minute  Motor_Temperature  Motor_Power  \
0                            94              45.62         2.93
1                            97              55.93         3.30
2                           103              45.40         3.94
3                           100              55.23         4.14
4                           108              54.51         4.37

   Motor_Efficiency  Motor Trip Failure
0             75.69                   0
1             73.78                   0
2             73.04                   0
3             77.55                   0
4             79.24                   0
```

[4]:
```python
df.set_index("Date (DD/MM/YYYY)", inplace=True)
df.index.names = ["Date"]
df.head()
```

[4]:
```
                 Pressure_Sensor_1  Pressure_Sensor_2  \
Date
14/03/2019 4:00          30.702663          29.010453
14/03/2019 4:01          30.384615          28.278746
14/03/2019 4:02          32.233728          27.993031
14/03/2019 4:03          31.545858          30.919861
14/03/2019 4:04          32.233728          30.954704

                 Motor_Revolutions_Per_Minute  Motor_Temperature  Motor_Power  \
Date
14/03/2019 4:00                            94              45.62         2.93
14/03/2019 4:01                            97              55.93         3.30
14/03/2019 4:02                           103              45.40         3.94
14/03/2019 4:03                           100              55.23         4.14
14/03/2019 4:04                           108              54.51         4.37

                 Motor_Efficiency  Motor Trip Failure
Date
14/03/2019 4:00             75.69                   0
14/03/2019 4:01             73.78                   0
14/03/2019 4:02             73.04                   0
14/03/2019 4:03             77.55                   0
14/03/2019 4:04             79.24                   0
```

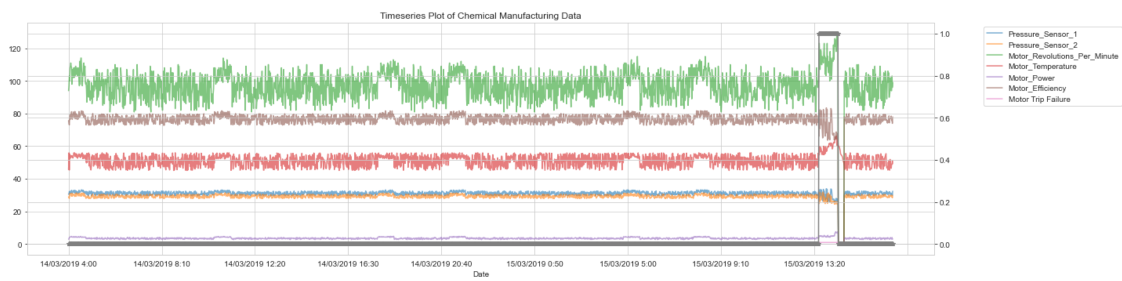
0.1.3 Exploratory data analysis

Box plot data distributions

[5]:
```
# !pip install plotly
#! pip install plotly_express==0.4.0
import plotly.express as px
fig = px.box(df, title="Chemical Manufacturing Data", template="gridon")
fig.show()
```

Time series line plot

[6]:
```
ax1 = df.plot(alpha=0.6)
ax1.xaxis.set_major_locator(plt.MaxNLocator(10))
ax2 = ax1.twinx()
ax2.plot(df['Motor Trip Failure'], color='grey', marker='*')
ax2.xaxis.set_major_locator(plt.MaxNLocator(10))
ax1.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.title("Timeseries Plot of Chemical Manufacturing Data")
plt.tight_layout()
plt.show()
```



The plots show several outliers/abnormalities in monitored continuous variables as well as the `Motor Trip Failure` binary variable, with largest deviations in various variables observed when `Motor Trip Failure` is True (=1)

[7]:
```
df = pd.read_csv('../data/chemical_manufacturing_time_series.csv',
                                  index_col="Date (DD/MM/YYYY)", parse_dates=True,␣
 ↪dayfirst=True)
column_names = df.drop('Motor Trip Failure', axis=1).columns
column_names
```

[7]:
```
Index(['Pressure_Sensor_1', 'Pressure_Sensor_2',
       'Motor_Revolutions_Per_Minute', 'Motor_Temperature', 'Motor_Power',
       'Motor_Efficiency'],
      dtype='object')
```

0.1.4 Anomaly detection using ADTK Interquartile Range Methodology

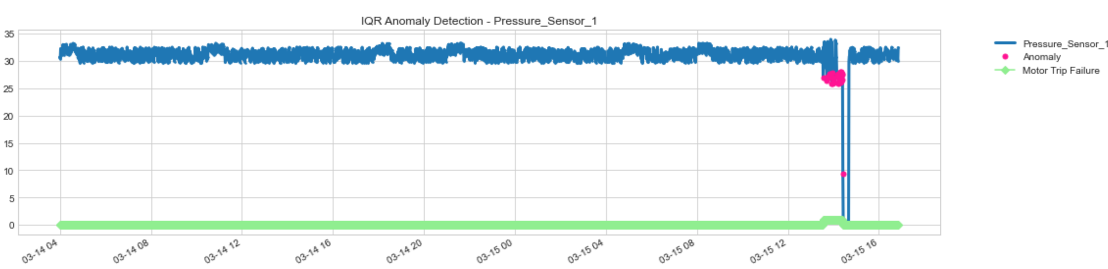
ADTK is a open-source Python package for unsupervised/rule-based models of time series anomaly detection. ADTK uses a `detector` component in the model to scan time series and return anomalous time points.

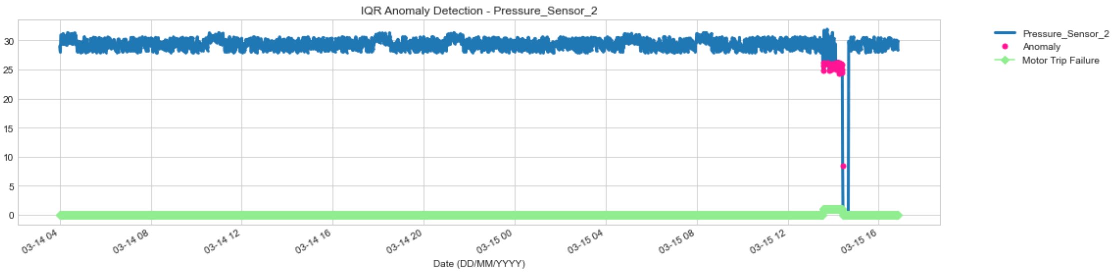
To detect outliers, a detector needs to learn the *normal range* of time series values.`adtk.detector.InterQuartileRangeAD` is a detector that learns the normal range and detects anomaly based on inter-quartile range of historical data. It compares time series values with 1st and 3rd quartiles of historical data, and identifies time points as anomalous when differences are beyond the inter-quartile range (IQR) times a user-given factor c - used to determine the bound of normal range (betweeen Q1-c∗IQR and Q3+c∗IQR). `fit_detect` method trains the detector and detects anomalies from the time series used for training; classifies data points returning binary series indicating normal/anomalous for each column of a DataDrame.

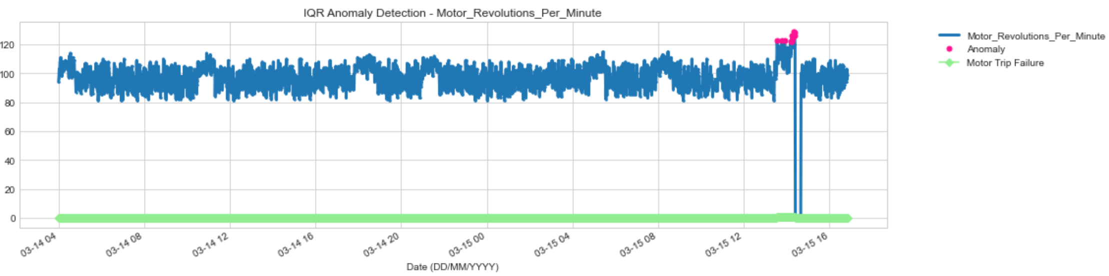
ADTK

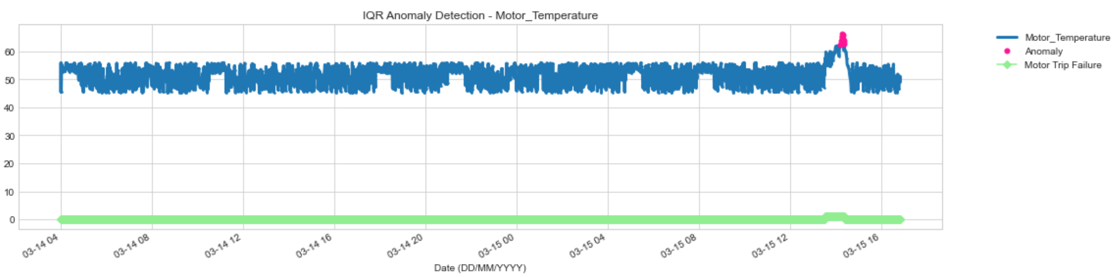
[8]:
```python
#! pip install adtk

from adtk.detector import InterQuartileRangeAD
from adtk.visualization import plot

def detect_anomalies_adtk(df, features, title = "IQR Anomaly Detection - "):
    iqr_ad = InterQuartileRangeAD(c=1.5)

    for feature in features:
        anomalies = iqr_ad.fit_detect(df[feature])

        plot(df[feature], anomaly=anomalies, ts_linewidth=3, ts_markersize=3,
             anomaly_markersize=5, anomaly_color='deeppink', anomaly_tag="marker")
        df['Motor Trip Failure'].plot(color='lightgreen', marker="D", linewidth=1.5)
        plt.title(title + feature)
        plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
        plt.tight_layout()
```

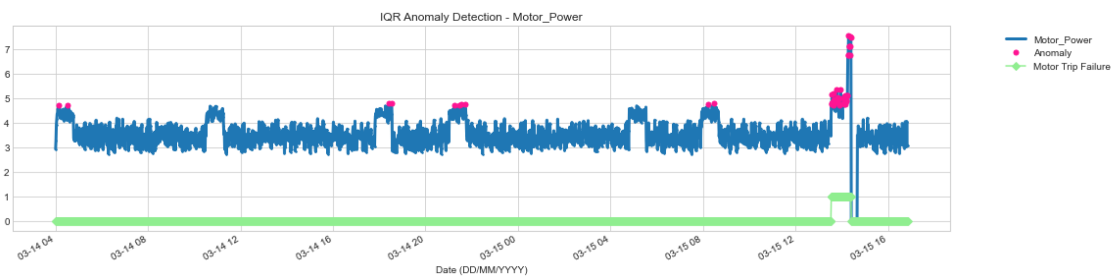
[9]:
```python
detect_anomalies_adtk(df, column_names)
```

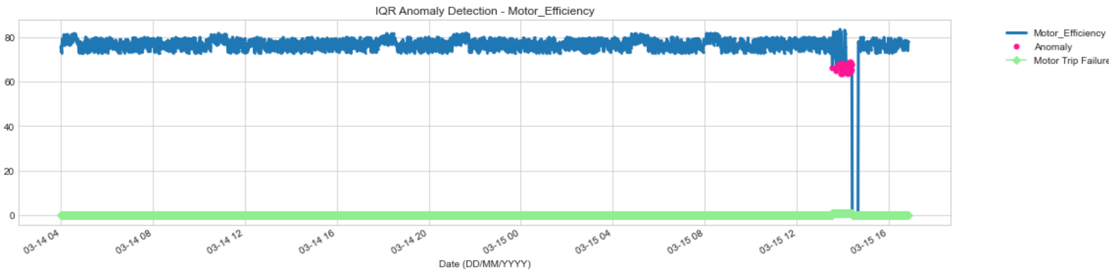



IQR Anomaly Detection - Pressure_Sensor_2
Pressure_Sensor_2
Anomaly
Motor Trip Failure
Date (DD/MM/YYYY)


IQR Anomaly Detection - Motor_Revolutions_Per_Minute
Motor_Revolutions_Per_Minute
Anomaly
Motor Trip Failure
Date (DD/MM/YYYY)


IQR Anomaly Detection - Motor_Temperature
Motor_Temperature
Anomaly
Motor Trip Failure
Date (DD/MM/YYYY)


IQR Anomaly Detection - Motor_Power
Motor_Power
Anomaly
Motor Trip Failure
Date (DD/MM/YYYY)



The visualization illustrates that the largest concentration of outliers occurs immediately before the Motor Trip Failure event. Moreover, Motor Power

[10]:
```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1

lower_limit = Q1 - 1.5 * IQR
upper_limit = Q3 + 1.5 * IQR

df_no_outliers = df[~((df < lower_limit) | ((df > upper_limit ))).any(axis=1)]
df_no_outliers.shape
```

[10]:
```
(2134, 7)
```

With detected outliers removed and `Motor Trip Failure` = True, there is no data left suggesting that the outliers are actually the events associated with Motor Failure and should not be removed as they contain valuable information that could be used for proactive analysis and improved identification of anomalous behavior. `Motor Power` trend is an indication for early preventive analysis.

[11]:
```
fig = px.box(df_no_outliers, title="No outliers Chemical Manufacturing Data", template="gridon")
fig.show()
```

[12]:
```
fig = px.box(df_no_outliers[~df_no_outliers['Motor Trip Failure']==1],
             title="No outliers Chemical Manufacturing Data having Motor Trip Failure",
             template="gridon")
fig.show()
```

[13]:
```
motor_power = (df['Motor_Power'].values.reshape(-1,1))
motor_power.shape
```

[13]:
```
(2212, 1)
```

0.1.5 Anomaly detection using Isolation Forest Technique

Isolation Forest is an efficient `unsupervised machine learning algorithm for anomaly detection`, especially in high-dimensional datasets, that identifies anomaly by isolating outliers in the data. The algorithm builds a Random Forest in which each Decision Tree is grown randomly: at each node it picks a feature randomly over all features, then it picks a random threshold value (between the min and max value of the selected feature) to split the dataset in half. The dataset is gradually divided this way, until all instances end up isolated from the other instances. Anomalies are usually far from other instances, so on average (across all Decision Trees) they tend to get isolated in fewer steps than normal instances.

As a parenthesis, Random Forest is a supervised technique used for profiling normal data points, which at each node picks a feature from a random subset of features based on the criteria of maximum reduction of impurity.

Since recursive partitioning can be represented by a tree structure, the number of splittings required to isolate a sample is equivalent to the path length from the root node to the terminating node. This path length, averaged over a forest of such random trees, is a measure of normality and the decision function. Random partitioning produces noticeable shorter paths for anomalies. Hence, when a forest of random trees collectively produce shorter path lengths for particular samples, they are highly likely to be anomalies.

Isolation Forest's `score_samples` method returns an anomaly score for each sample. The anomaly score of an input sample is computed as the mean anomaly score of the trees in the forest. The measure of normality of an observation given a tree is the depth of the leaf containing this observation, which is equivalent to the number of splittings required to isolate this point. Values that are < -0.5 are more "normal" and those that exceed the -0.5 threshold are more likely to be Anomalous. So, the lower the more abnormal. The `predict` method tells, for each observation, whether or not (+1 or -1) it should be considered as an inlier according to the fitted model.

IsolationForest

[14]:
```
#import sklearn
#print(sklearn.__version__)

from sklearn.ensemble import IsolationForest

def detect_anomalies_isoforest(features):
    model = IsolationForest()
    model.fit(features)
    scores = model.score_samples(features)
    predictions = model.predict(features)

    return scores, predictions
```

[15]:
```
scores, predictions = detect_anomalies_isoforest(motor_power)
```

[16]:
```
df_motor_power = df[['Motor_Power']]
df_motor_power['Anomaly_Scores'] = scores
df_motor_power['Anomaly_Classification'] = predictions
```

```
df_motor_power['Anomaly_Classification_Cutoff'] = np.
 ↪where(df_motor_power['Anomaly_Scores'] < -0.5, 1, 0)
df_motor_power.head()
```

[16]:
```
                     Motor_Power  Anomaly_Scores  Anomaly_Classification  \
Date (DD/MM/YYYY)
2019-03-14 04:00:00         2.93       -0.501221                      -1
2019-03-14 04:01:00         3.30       -0.416089                       1
2019-03-14 04:02:00         3.94       -0.457214                       1
2019-03-14 04:03:00         4.14       -0.489200                       1
2019-03-14 04:04:00         4.37       -0.504221                      -1

                     Anomaly_Classification_Cutoff
Date (DD/MM/YYYY)
2019-03-14 04:00:00                              1
2019-03-14 04:01:00                              0
2019-03-14 04:02:00                              0
2019-03-14 04:03:00                              0
2019-03-14 04:04:00                              1
```

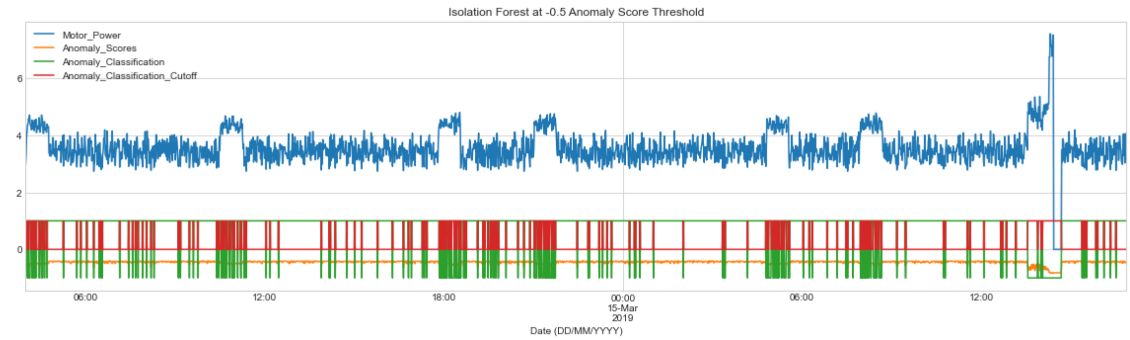
[17]:
```
df_motor_power.describe()
```

[17]:
```
       Motor_Power  Anomaly_Scores  Anomaly_Classification  \
count  2212.000000     2212.000000             2212.000000
mean      3.555705       -0.455850                0.720615
std       0.600558        0.063362                0.693492
min       0.000000       -0.831910               -1.000000
25%       3.220000       -0.470300                1.000000
50%       3.490000       -0.435395                1.000000
75%       3.810000       -0.419209                1.000000
max       7.560000       -0.402997                1.000000

       Anomaly_Classification_Cutoff
count                    2212.000000
mean                        0.139693
std                         0.346746
min                         0.000000
25%                         0.000000
50%                         0.000000
75%                         0.000000
max                         1.000000
```

Visualize the anomalies generated from the Isolation Forest for the Motor_Power observable, selected as the key variable showcasing several anomalous behaviors with ADTK.

[18]:
```
df_motor_power[['Motor_Power', 'Anomaly_Scores', 'Anomaly_Classification',␣
 ↪'Anomaly_Classification_Cutoff']].plot()
```

```
plt.title("Isolation Forest at -0.5 Anomaly Score Threshold")
plt.show()
```



The plot suggests a number of false positives where periodic peaks occur and are flagged as anomalies but they not be necessarily points of concern. However, the Isolation forest has captured the critical point near Failure where scores have the largest negative values, the calculated anomaly cutoff is 1 and all predictions are -1 (i.e. Anomalous).

Setting a anomaly score threshold of -0.75 improves the sensitivity of Isolation Forest to capture only the critical point when the motor is overloaded, its power shoots up and motor trips. Minimum anomality score, cutoff 1 and prediction -1.

[19]:
```
df_motor_power['Anomaly_Classification_Cutoff'] = np.
 ↪where(df_motor_power['Anomaly_Scores'] < -0.75, 1, 0)
df_motor_power.head()
```

[19]:
```
                     Motor_Power  Anomaly_Scores  Anomaly_Classification  \
Date (DD/MM/YYYY)
2019-03-14 04:00:00         2.93       -0.501221                      -1
2019-03-14 04:01:00         3.30       -0.416089                       1
2019-03-14 04:02:00         3.94       -0.457214                       1
2019-03-14 04:03:00         4.14       -0.489200                       1
2019-03-14 04:04:00         4.37       -0.504221                      -1

                     Anomaly_Classification_Cutoff
Date (DD/MM/YYYY)
2019-03-14 04:00:00                              0
2019-03-14 04:01:00                              0
2019-03-14 04:02:00                              0
2019-03-14 04:03:00                              0
2019-03-14 04:04:00                              0
```

[20]:
```
df_motor_power[['Motor_Power','Anomaly_Scores', 'Anomaly_Classification',␣
 ↪'Anomaly_Classification_Cutoff']].plot()
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
```

```python
plt.title("Isolation Forest at 0.75 Anomaly Score Threshold - Motor_Power")

detect_anomalies_adtk(df, ['Motor_Power'])
```