Outlier (Anomaly) Detection Modelling in PMML

Jaroslav Kuchař[1,2] and Adam Ashenfelter[3] and Tomáš Kliegr[2]

[1] Web Intelligence Research Group, Faculty of Information Technology,
Czech Technical University in Prague, Czech Republic
jaroslav.kuchar@fit.cvut.cz
[2] Department of Information and Knowledge Engineering,
Faculty of Informatics and Statistics, University of Economics Prague
Czech Republic
tomas.kliegr@vse.cz
[3] BigML Inc.
Corvallis, Oregon
United States
ashenfelter@bigml.com

Abstract. PMML is an industry-standard XML-based open format for representing statistical and data mining models. Since PMML does not yet support outlier (anomaly) detection, in this paper we propose a new outlier detection model to foster interoperability in this emerging field. Our proposal is included in the PMML RoadMap for PMML 4.4.
We demonstrate the proposed format on one supervised and two unsupervised outlier detection approaches: association rule-based classifier CBA, frequent-pattern based method FPOF and isolation forests.

Keywords: outlier detection, anomaly detection, PMML, frequent pattern mining, rule-based classifiers, isolation forests

1 Introduction

Outliers (also called anomalies) are observations that differ from other observations to the extent that they arouse suspicion that they were generated by a different mechanism than the rest of the data. Algorithms that can detect outliers have a growing list of applications, including fraud detection, intrusion detection, medical diagnosis and sensor events [1, 3].

There are many existing approaches that can be used to detect outliers. Selection of the proper method depends on the character of the input data and goals, level of the supervision, dimensionality of input data, algorithmic approach (proximity-based or clustering-based techniques), and type of outliers detected (point, contextual or collective outliers). In all their variety, all approaches generally provide output value for each input instance that represents the level of *anomality*. This is either a class label (usually a binary flag) or a numerical score.

Despite the growing need for standard approach for handling outlier detection models generated by different approaches and software tools implementing

them, there has been so far little standardization effort that would foster interoperability between the individual components handling these models in the analytics tool chain.

PMML[4] is an XML-based open standard for representing statistical and data mining models. It supports many existing models including association rules, classification, regression or clustering models and also neural networks. Many existing tools and data mining solutions support this standard [2]. Since PMML does not yet support outlier (anomaly) detection, in this paper we propose a new outlier detection model to foster interoperability in this emerging field. Our proposal is included in the PMML RoadMap for PMML 4.4.

The paper is organized as follows: In Section 2 we use XML schema fragments to describe the proposed PMML extension. Section 3 demonstrates the versatility of the proposed specification on three different types of models. In Section 4 we compare the proposed specification with another proposed PMML extension. Finally, the conclusions present a brief summary and outlook.

2 Specification

Since PMML is an XML-based standard, the proposed specification for the outlier detection model is in the form of an XML Schema model. Figure 1 depicts the main structure of PMML. Our extension adds a new model to the list of available models.

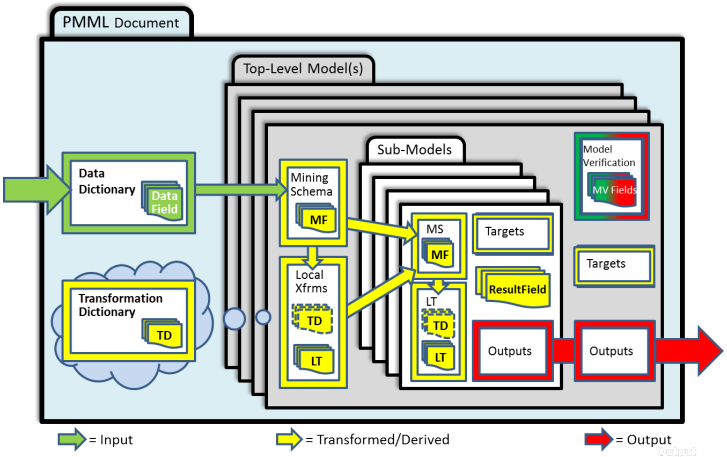

Fig. 1. Main structure of the PMML document. Source: `http://dmg.org/pmml/v4-3/FieldScope.html`.

[4] `http://dmg.org/pmml/pmml-v4-3.html`

2.1 Outlier Detection Model

Listing 1.1 shows the main element of the proposed model: `OutlierDetectionModel`. It contains required standard elements from the PMML specification - `Extension` and `Mining Schema`.

Listing 1.1. Outlier Detection Model

```
<xs:element name="OutlierDetectionModel">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="pmml:Extension" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="pmml:MiningSchema" minOccurs="0"/>
            <xs:element ref="ParameterList" minOccurs="0"/>
            <xs:choice>
                <xs:element ref="pmml:AssociationModel" minOccurs="0"/>
                <xs:element ref="pmml:Segmentation" minOccurs="0"/>
                <!-- The rest of other possible models are skipped for the demonstration
                     purpose -->
            </xs:choice>
            <xs:element ref="LabeledInstances" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="modelName" type="xs:string"/>
        <xs:attribute name="algorithmName" type="ALGORITHM-TYPE" use="required"/>
        <xs:attribute name="typeOfOutliers" type="OUTLIERS-TYPE" use="required"/>
        <xs:attribute name="numberOfOutliers" type="xs:positiveInteger" />
        <xs:attribute name="output" type="OUTLIERS-OUTPUT-TYPE" use="required"/>
    </xs:complexType>
</xs:element>
```

Remaining elements and attributes were newly added. `ParameterList` is an optional structure containing specific parameters for each supported algorithm/approach. PMML currently supports a variety of existing machine learning algorithms such as decision trees or regression, which can serve as basis for outlier detection algorithms. However, these existing PMML models cannot be directly reused, because the adaptation of existing generic machine learning model for outlier detection typically implies introduction of new parameters and/or amendments of the existing ones. We therefore decided that `ParameterList` will be a generic structure offering a configurable list of key-value pairs as parameters (cf. Listing 1.2). The list can contain generic parameters of the underlying model or any proprietary configurations of each algorithm, which are important to compute the output value.

Listing 1.2. Model Parameters

```
<xs:element name="ParameterList">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="pmml:Extension" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="Parameter" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="Parameter">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="pmml:Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="value" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>
```

The model specifies a set of attributes for description of the type of the outlier detection model. Similarly to other PMML models, there is an optional `modelName` attribute and the following required attributes:

- `algorithmName` – specification of algorithm type. Currently supported and demonstrated algorithms are isolation forests, frequent pattern mining outliers and a rule based classifier. The list of the allowed algorithm names is extensible and currently defined as the `ALGORITHM-TYPE` in Listing 1.3.
- `output` – defines the output of the outlier detection algorithm. Supported options are *label* or numeric *score* (see Listing 1.3).

In addition, there are the following required attributes specific for outlier detection:

- `typeOfOutliers` – defines the type of the outlier the model is able to handle. Supported types are *point*, *collective* and *contextual* (see Listing 1.3).
- `numberOfOutliers` – the attribute specifies the number of outliers that should be returned as the output of the task.

Listing 1.3. Model Types

```
<xs:simpleType name="OUTLIERS-TYPE">
    <xs:restriction base="xs:string">
        <xs:enumeration value="point"/>
        <xs:enumeration value="collective"/>
        <xs:enumeration value="contextual"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="OUTLIERS-OUTPUT-TYPE">
    <xs:restriction base="xs:string">
        <xs:enumeration value="score"/>
        <xs:enumeration value="label"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="ALGORITHM-TYPE">
    <xs:restriction base="xs:string">
        <xs:enumeration value="iforest"/>
        <xs:enumeration value="fpof"/>
        <xs:enumeration value="cba"/>
        <!-- The rest of possible algorithm types are skipped for the demonstration
             purpose -->
    </xs:restriction>
</xs:simpleType>
```

This specification also allows to provide detailed description of detected outliers (cf. Listing 1.4). The output is in form of the set of top labeled instances. The specification is similar to training instances of KNN model in PMML[5]. The format is a table, where each row contains elements with the original data. The required attributes `id` and `output` represent the original id of the row in the data and the output value (label or score) assigned by the algorithm respectively.

[5] `http://dmg.org/pmml/v4-3/KNN.html#xsdElement_TrainingInstances`

Listing 1.4. Labelled instances

```
<xs:element name="LabeledInstances">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="pmml:Extension" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="pmml:InstanceFields"/>
            <xs:element ref="InlineTable"/>
        </xs:sequence>
        <xs:attribute name="recordCount" type="xs:positiveInteger" use="optional"/>
        <xs:attribute name="fieldCount" type="xs:positiveInteger" use="optional"/>
    </xs:complexType>
</xs:element>
<xs:element name="InlineTable">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="pmml:Extension" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="Row" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="Row">
    <xs:complexType>
        <xs:complexContent mixed="true">
            <xs:restriction base="xs:anyType">
                <xs:sequence>
                    <xs:any processContents="skip" minOccurs="1" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="id" type="xs:string" use="required"/>
                <xs:attribute name="output" type="xs:string" use="required"/>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
```

An alternative way would be to use the `Output` element already defined in PMML instead of introducing a new structure consisting of multiple rows. While using the `Output` element would be more in line with common practice for existing PMML models, there are two important limitations. First, this would imply that the existing set of features/operations defined in PMML is sufficient to describe how the output for a specific anomaly detection model is obtained. Second, this would not support the use case when output for limited number of detected outliers should be returned.

3 Examples

As examples we use three algorithms: frequent-pattern mining algorithm, isolation forest and rule-based classifier: The first two approaches are unsupervised, rule-based classifier is an example of the standard supervised approach.

– *Outlier detection based on frequent pattern mining* – the FPOF (Frequent Pattern Contradiction Outlier Factor) method [4]. Reference implementation is available as an R package [6]. This package also already exports to the proposed PMML extension.

[6] `https://github.com/jaroslav-kuchar/fpmoutliers`

– *Isolation forest* – well-known algorithm with good quality/complexity ratio. Represented as an ensemble of trees. Reference implementation is provided by BigML API [7] or scikit-learn (v 0.18)[8].
– *Rule based classifier* – standard supervised classification algorithm based on rules. We use reference implementation available of the Classification By Associations (CBA) algorithm, which is available as an R package [9].

Table 1 presents a simple dataset with customer information, which we will use for demonstration purposes. This dataset was originally proposed by [4].

Table 1. Example dataset

Index	Age-Range	Car	Salary-Level	Class
1	Middle	Sedan	Low	Normal
2	Middle	Sedan	High	Normal
3	Young	Sedan	High	Normal
4	Middle	Sedan	Low	Normal
5	Young	Sports	High	Outlier
6	Young	Sports	Low	Normal
7	Middle	Sedan	High	Normal
8	Young	Sports	Low	Normal
9	Middle	Sedan	High	Normal
10	Young	Sports	Low	Normal

3.1 Frequent-pattern mining

Listing 1.5 describes an example of output of the frequent pattern based unsupervised method. The algorithm detects point outliers and provides scores as the output (line 9). Since the method is built on top of frequent patterns, the association model is included[10]. The final score is computed proportionally to the number of matching frequent itemsets and their support (cf. [4] for details).

To represent outliers based on frequent itemsets our proposal reuses the complete `AssociationModel`. What is actually needed is a way to express frequent itemsets, which is only a part of it. An alternate more complex version of the schema, which we considered, would introduce `FrequentItemset` model as a standalone model, and then reuse it in the `OutlierDetectionModel`.

Listing 1.5. Frequent-pattern mining example

```
1 <?xml version="1.0"?>
2 <PMML version="4.3" xmlns="http://www.dmg.org/PMML-4_3" xmlns:xsi="http://www.w3.org
      /2001/XMLSchema-instance" xmlns:od="http://www.example.com/od" xsi:schemaLocation="
      http://www.dmg.org/PMML-4_3 pmml-4-3+od-0-1.xsd">
```

[7] `https://bigml.com/api/anomalies`
[8] `http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html`
[9] `https://cran.r-project.org/web/packages/rCBA/index.html`
[10] `http://dmg.org/pmml/v4-3/AssociationRules.html`

```
3  <Header copyright="" description="">
4   <Timestamp>2017-04-30 07:01:05</Timestamp>
5  </Header>
6  <DataDictionary>
7   <!-- Data Dictionary is skipped for the demonstration purpose -->
8  </DataDictionary>
9  <od:OutlierDetectionModel xmlns="http://www.example.com/od" algorithmName="fpof"
       modelName="FPI OD model" typeOfOutliers="point" numberOfOutliers="10" output="
       score">
10
11  <MiningSchema xmlns="http://www.dmg.org/PMML-4_3">
12   <MiningField name="Age.Range"/>
13   <MiningField name="Car"/>
14   <MiningField name="Salary.Level"/>
15  </MiningSchema>
16
17  <ParameterList>
18   <Parameter name="minSupport" value="0.1"/>
19  </ParameterList>
20
21  <AssociationModel functionName="associationRules" numberOfItems="6" minimumSupport="
       0.1" minimumConfidence="0" numberOfItemsets="22" numberOfRules="0" xmlns="http://
       www.dmg.org/PMML-4_3">
22   <MiningSchema>
23    <MiningField name="transaction" usageType="group"/>
24    <MiningField name="item" usageType="active"/>
25   </MiningSchema>
26   <Item id="1" value="Age.Range=Middle"/>
27   <Item id="2" value="Age.Range=Young"/>
28   <Item id="3" value="Car=Sedan"/>
29   <Item id="4" value="Car=Sports"/>
30   <Item id="5" value="Salary.Level=High"/>
31   <Item id="6" value="Salary.Level=Low"/>
32   <Itemset id="1" numberOfItems="1" support="0.4">
33    <ItemRef itemRef="4"/>
34   <!-- The rest of the association model is skipped for the demonstration purpose -->
35  </AssociationModel>
36
37  <LabeledInstances>
38   <InlineTable>
39    <Row id="5" output="2.654762">
40     <Age.Range>Young</Age.Range>
41     <Car>Sports</Car>
42     <Salary.Level>High</Salary.Level>
43    </Row>
44      <!-- The rest of the table is skipped for the demonstration purpose -->
45   </InlineTable>
46  </LabeledInstances>
47 </od:OutlierDetectionModel>
48 </PMML>
```

3.2 Isolation forest

As second example of representing output of an unsupervised method we selected isolation forests as implemented in `bigml.com`. Since isolation forests are built from several trees, the model uses the Segmentation[11] specification to combine multiple models and build the final model from multiple models.

Listing 1.6 describes an example of isolation forest as implemented in `bigml.com`. There is only one required parameter specifying number of trees that should be composed – two trees for this example (line 22). The algorithm also detects

[11] `http://dmg.org/pmml/v4-3/MultipleModels.html`

point outliers and provides scores as the output (line 13). The final output score for each instance (e.g. as on line 59) is derived from the combination of available trees and depth of relevant branches/predicates matching the instance (cf. [5] for details).

Listing 1.6. Frequent-pattern mining example

```
1  <?xml version="1.0"?>
2  <PMML version="4.3" xmlns="http://www.dmg.org/PMML-4_3"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:od="http://www.example.com/
         od"
4   xsi:schemaLocation="http://www.dmg.org/PMML-4_3 pmml-4-3+od-0-1.xsd">
5
6   <Header copyright="" description="">
7    <Timestamp>2017-04-30 09:02:01</Timestamp>
8   </Header>
9   <DataDictionary>
10   <!-- Data Dictionary is skipped for the demonstration purpose -->
11  </DataDictionary>
12  <od:OutlierDetectionModel xmlns="http://www.example.com/od" algorithmName="iforest"
         modelName="BigML Isolation Forests"
13   typeOfOutliers="point" numberOfOutliers="10" output="score">
14
15   <MiningSchema xmlns="http://www.dmg.org/PMML-4_3">
16    <MiningField name="Age.Range"/>
17    <MiningField name="Car"/>
18    <MiningField name="Salary.Level"/>
19   </MiningSchema>
20
21   <ParameterList>
22    <Parameter name="forest_size" value="2"/>
23   </ParameterList>
24
25   <Segmentation multipleModelMethod="selectAll" xmlns="http://www.dmg.org/PMML-4_3">
26
27    <Segment id="1">
28     <True/>
29     <TreeModel modelName="exampleDataset" functionName="mixed">
30      <MiningSchema>
31       <MiningField name="Age.Range"/>
32       <MiningField name="Car"/>
33       <MiningField name="Salary.Level"/>
34      </MiningSchema>
35      <Node recordCount="6">
36       <True/>
37       <Node recordCount="2">
38        <CompoundPredicate booleanOperator="and">
39         <SimplePredicate field="Age.Range" operator="equal" value="Middle"/>
40         <SimplePredicate field="Car" operator="equal" value="Sedan"/>
41        </CompoundPredicate>
42        <!-- The rest of the model is skipped for the demonstration purpose -->
43       </Node>
44      </Node>
45     </TreeModel>
46    </Segment>
47
48    <Segment id="2">
49     <True/>
50     <TreeModel modelName="exampleDataset" functionName="mixed">
51      <!-- The rest of the model is skipped for the demonstration purpose -->
52     </TreeModel>
53    </Segment>
54
55   </Segmentation>
56
57   <LabeledInstances>
58    <InlineTable>
```

```
    <Row id="5" output="0.52717">
     <Age.Range>Young</Age.Range>
     <Car>Sports</Car>
     <Salary.Level>High</Salary.Level>
    </Row>
    <!-- The rest of the table is skipped for the demonstration purpose -->
   </InlineTable>
  </LabeledInstances>
 </od:OutlierDetectionModel>
</PMML>
```

3.3 Rule-based classifier

The rule-based classifier is a representative of a supervised method – standard classification algorithm applied on the outlier detection problem. Let assume that we have the fifth instance annotated as the outlier using the Class attribute (See Table 1). The rule-based classifier (here CBA) can learn rules that label specific instances as outliers and the rest as normal instances.

A simplified output of the rule-based classier can look as follows:

- { Car=Sports & Salary-Level=High } → { Class=Outlier })
- {} → { Class=Normal })

The structure of the model in PMML (Listing 1.7) is similar to the unsupervised frequent pattern based model. The difference is in setting of the algorithm name (line 9), output type and parameters (starting from line 18). The model also reuses `AssociationModel` to represent rules.

Listing 1.7. Rule-based classifier example

```
1  <?xml version="1.0"?>
2  <PMML version="4.3" xmlns="http://www.dmg.org/PMML-4_3" xmlns:xsi="http://www.w3.org
       /2001/XMLSchema-instance" xmlns:od="http://www.example.com/od" xsi:schemaLocation="
       http://www.dmg.org/PMML-4_3 pmml-4-3+od-0-1.xsd">
3   <Header copyright="" description="">
4    <Timestamp>2017-04-30 11:39:17</Timestamp>
5   </Header>
6   <DataDictionary>
7    <!-- Data Dictionary is skipped for the demonstration purpose -->
8   </DataDictionary>
9   <od:OutlierDetectionModel xmlns="http://www.example.com/od" algorithmName="cba"
        modelName="CBA OD model" typeOfOutliers="point" numberOfOutliers="10" output="
        label">
10
11   <MiningSchema xmlns="http://www.dmg.org/PMML-4_3">
12    <MiningField name="Age.Range"/>
13    <MiningField name="Car"/>
14    <MiningField name="Salary.Level"/>
15    <MiningField name="Class"/>
16   </MiningSchema>
17
18   <ParameterList>
19    <Parameter name="minSupport" value="0.1"/>
20    <Parameter name="minConfidence" value="0.1"/>
21    <Parameter name="label" value="Class"/>
22   </ParameterList>
23
24   <AssociationModel functionName="associationRules" numberOfItems="6" minimumSupport="
        0.1" minimumConfidence="0.1" numberOfItemsets="29" numberOfRules="12" xmlns="http
        ://www.dmg.org/PMML-4_3">
```

```
   <MiningSchema>
    <MiningField name="transaction" usageType="group"/>
    <MiningField name="item" usageType="active"/>
   </MiningSchema>
   <Item id="1" value="Age.Range=Middle"/>
   <Item id="2" value="Age.Range=Young"/>
   <Item id="3" value="Car=Sedan"/>
   <Item id="4" value="Car=Sports"/>
   <Item id="5" value="Salary.Level=High"/>
   <Item id="6" value="Salary.Level=Low"/>
   <Item id="7" value="Class=Normal"/>
   <Item id="8" value="Class=Outlier"/>
   <Itemset id="1" numberOfItems="1" support="0.4">
    <ItemRef itemRef="4"/>
    <AssociationRule support="0.4" confidence="0.7" antecedent="7" consequent="10"/>
   <!-- The rest of the association model is skipped for the demonstration purpose -->
  </AssociationModel>

  <LabeledInstances>
   <InlineTable>
    <Row id="5" output="Outlier">
     <Age.Range>Young</Age.Range>
     <Car>Sports</Car>
     <Salary.Level>High</Salary.Level>
    </Row>
      <!-- The rest of the table is skipped for the demonstration purpose -->
   </InlineTable>
  </LabeledInstances>
 </od:OutlierDetectionModel>
</PMML>
```

4 Related Work

We have identified one existing approach to represent outlier detection models as an extension of PMML, which is used by the R-based implementation of isolation forests[12]. This specification implemented as part of the jpmml package[13] is based on the `regression` mining function of underlying models from PMML.

Basing model on regression implies supervised learning. Isolation forests do produce numeric scores, but they are generally considered as an unsupervised model. Furthermore, the regression framework is not suitable for other types of outlier detection algorithms.

Our model proposal fundamentally differs from `jpmml` in that it is not based on a particular existing PMML model, but fosters reuse of fragments from `AssociationModel` and `Segmentation` PMML models, which as we demonstrated, allows support for a broader range of outlier detection algorithms, including isolation forests.

5 Conclusions

Designing an anomaly detection model for PMML is particularly hard, because, in principle, nearly all data mining models can produce information about outliers. The goal of our work was to design modular solution that would support

[12] `https://r-forge.r-project.org/R/?group_id=479`
[13] `https://github.com/jpmml/r2pmml`

broader range of anomaly detection algorithms. We demonstrated the proposed format on three algorithms. Reference implementation of the export is available as an R package for frequent pattern mining outlier detection[14]. Including the `OutlierDetection` model is on the roadmap for the next release of the PMML specification.

Acknowledgements. The authors would like to thank the anonymous reviewers for their insightful comments. This research was supported by the European Union's H2020 EU research and innovation programme via the OpenBudgets.eu project (under grant agreement No 645833). Tomáš Kliegr was supported by long term institutional support of research activities by Faculty of Informatics and Statistics, University of Economics, Prague.

References

1. Aggarwal, C.C.: An Introduction to Outlier Analysis, pp. 1–34. Springer International Publishing, Cham (2017), http://dx.doi.org/10.1007/978-3-319-47578-3_1
2. Guazzelli, A., Zeller, M., Lin, W.C., Williams, G.: PMML: An Open Standard for Sharing Models. The R Journal 1(1), 60–65 (2009), https://journal.r-project.org/archive/2009/RJ-2009-010/index.html
3. Hawkins, D.: Identification of Outliers. Monographs on applied probability and statistics, Chapman and Hall (1980)
4. He, Z., Xu, X., Huang, Z., Deng, S.: FP-outlier: Frequent pattern based outlier detection. Computer Science and Information Systems/ComSIS 2(1), 103–118 (2005)
5. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: Proceedings of the 8th IEEE International Conference on Data Mining (ICDM'08). pp. 413–422 (2008)

[14] https://github.com/jaroslav-kuchar/fpmoutliers