See the forest for the trees

How to detect outliers in your data using the Isolation Forest algorithm

Hyunsu Kim, FSA
Michael Leitschkis, CERA, DAV

Section 1: Introduction

Outlier detection is the process of identifying data points that drastically differ from so-called “normal instances” in a given data set.

Outlier detection delivers critical information across many different domains in finance, such as financial reporting, fraud detection, and portfolio risk management. Recognizing anomaly patterns not only helps us detect potential errors at early stages but also enables us to uncover potential insights on the underlying data.

Most of the existing machine learning algorithms that can be used for the outlier detection hinge on profiling models based on normal instances.

Hence, in this paper we consider a technique called the Isolation Forest, which overcomes the shortcomings of classic anomaly detection algorithms. It has already been successfully applied across other disciplines, ranging from astrophysics to private wealth management. In astrophysics, one application has attempted to discover a new star by detecting an anomaly image compared to those of existing stars.[1] In private wealth management, the Isolation Forest approach has been helpful in detecting fraud and money laundering practices.[2] In this paper, we will apply the Isolation Forest approach to the life insurance context.

The proposed methodology offers the following advantages over traditional approaches:[3]

- Explicit isolation of anomalies without profiling normal instances
- Linear time complexity at a low memory requirement
- Scalability allowing the use of large data sets and high-dimensional data involving large numbers of attributes
- The ability to work without knowledge about outliers (labels) in the existing data set, as it is an unsupervised learning method, unlike alternative supervised techniques requiring expert judgment or known labels.

The purpose of this paper is to discuss an algorithm for an efficient automated detection on outliers in both small and large data sets.

The contents of this paper are organized as follows:

- **Section 2:** Isolation Forest algorithm walk-through.
- **Section 3:** Case studies (univariate and multivariate) that look at the underlying market movements of sub-funds for a contrived variable annuity portfolio.
- **Section 4:** Practical recommendations and conclusions

1 Hariri, S. & Kind M.C. (June 21, 2018). Isolation Forest for Anomaly Detection. Retrieved November 15, 2019, from http://www.ncsa.illinois.edu/Conferences/LSST18/assets/pdfs/hariri_forest.pdf.

2 Sharova, E. (May 27, 2018). Video: Unsupervised Anomaly Detection With Isolation Forest. PyData. Retrieved November 15, 2019, from https://www.youtube.com/watch?v=5p8B2Ikcw-k.

3 Liu, F.T., Ting, K.M., & Zhou, Z.H. (December 2008), Isolation Forest, in 8th IEEE International Conference on Data Mining (pp. 413-422). IEEE.

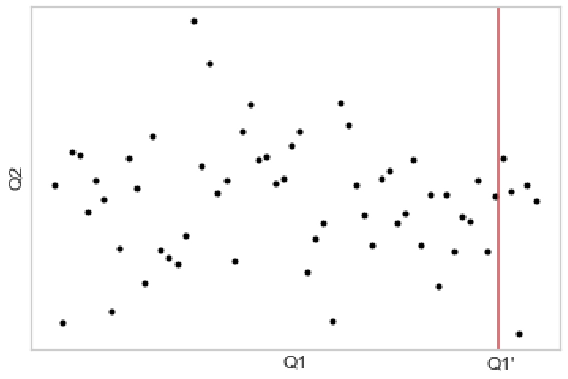
Section 2: Isolation Forest algorithm walk-through

The main purpose of this section is to discuss the algorithm step by step, explaining it in layman's terms.[4]

STEP 1A: BINARY DECISION SPLIT

In this step, we split our data sample. To do so, we randomly select one of the attributes (in a simple two-dimensional illustration, let us call them Q1 and Q2), and then we randomly choose a splitting value for the attribute just selected—anywhere between the minimal and the maximal value of that attribute. Figure 1 illustrates how Q1 has been selected and then a splitting value Q1' has been chosen.

FIGURE 1: BINARY DECISION SPLIT

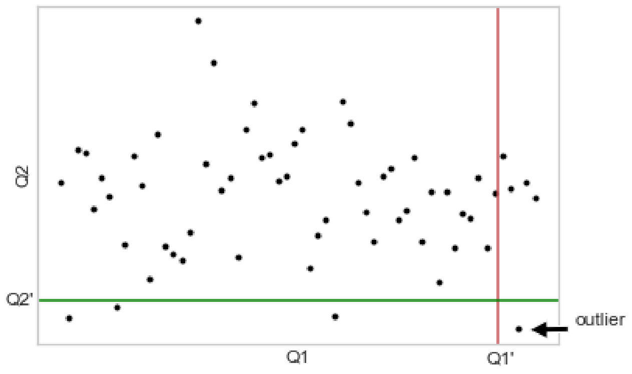


STEP 1B: GENERATION OF AN ISOLATION TREE

In this step, we iterate the binary decision split carried out in Step 1a. It takes fewer iterations to isolate anomalous data points than normal ones—see the data point in the lower-right corner in Figure 2, for which two iterations have been sufficient. This collection of binary splits is called an *isolation tree*. Each binary split can be thought of as an *internal node* of the tree. One split value (test) and exactly two branches (*daughter nodes*) emanate from an internal node: one containing points less than the split value (such as Q1' above) and the other comprised of points greater than the split value. When a point has been isolated, this corresponds to an *external node*.

Under the *path length* of a point, we understand the number of edges the point passes until it is isolated. For example, the path length for the aforementioned point in the lower-right corner in Figure 2 equals 2.

FIGURE 2: GENERATION OF AN ISOLATION TREE



Note that we iterate the binary decision split until we reach a tree height, which can be either user-defined or set by the algorithm. In the latter case, a standard choice is the average tree height, which can be shown to be proportional to the logarithm of the sample size.[5] Setting such a limit allows us to save computational resources and is natural because points with path lengths shorter than the average are of interest (most likely outliers).

STEP 2: GENERATION OF AN ISOLATION FOREST

Next, we repeat all of the steps above in order to generate another isolation tree. We continue our iterations until we have created a large enough collection of isolation trees, which is called an *Isolation Forest*.

As the reader may expect, there is no precise mathematical definition as to how many trees make up a forest. For our purposes, the reader might think of 15 to 100 trees as a sensible size for an Isolation Forest.

STEP 3: ANOMALY SCORE CALCULATION FOR EACH DATA POINT

For each data point in our original data set, we now execute the following process:

Feed a data point into one of the trees in the forest. Find its position in the tree by successively applying the binary splits at each internal node being passed by the point. Once the position of the point in the tree has been found, the anomaly score of a given data point is calculated as $S = 2^{-E/c(n)}$, where E is the average path length for this data point and $c(n)$ is a universal renormalizing constant, which is a function of the sample size n and measures the expected number of splits to isolate a given

4 For a more technical exposition, including relevant model validation results, please refer to Liu, F.T. et al., op cit.

5 Detailed in Liu, F.T. et al., op cit.

6 Ibid.

point within this sample size. More precisely, consider a forest made of N trees. For each tree i, denote the number of binary splits needed to find the data point of interest by M_i , and the number of remaining points in the final node by k_i (which can be two or more because a tree height limit has been set). Then the average path length is calculated as:

$$E = \frac{1}{N}\sum_{i=1}^{N} M_i + 1_{k_i \geq 2} c(k_i)$$

Note that the additive adjustment representing the expected number of splits when multiple points $(k_i \geq 2)$ remains in the external node in order to recover an unbiased expected path length.

- We calculate this anomaly score for each tree and average them out across many different trees. The average anomaly score across the trees will be then the final anomaly score for a given data point in question.
- Numerically, an outlier will feature an anomaly score around 0.6, while a normal instance will typically produce an anomaly score below 0.5.[6]

Section 3: Simple case study

In this section, we expand on the example from the previous section and develop it step by step to be more fully illustrative of typical Isolation Forest use cases.

To be more precise, we analyze a contrived variable annuity portfolio where the policyholders have the flexibility to allocate their assets to up to 14 funds. For the sake of simplicity, we look into just one policy and examine the evolution of the policyholder's overall account value (AV) over time in order to understand which daily data constitute normal instances and which could be outliers.

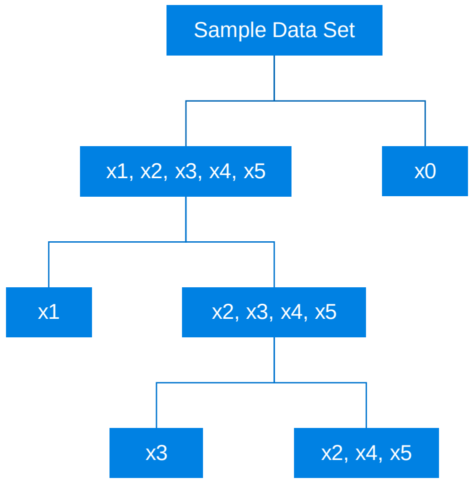
UNIVARIATE CASE

In order to formulate this problem as a univariate one, we ignore the evolution of the 14 funds mentioned above over time and only consider the overall account value for the time being. Revisiting the data shown at the end of Section 2, we look into the daily AV evolution over six days.

AV in this example (the first field of Figure 4) stands for a policyholder's Account Value and x0,…,x5 represent six different time points.

For the sake of visual simplicity, here we produce a single isolation tree in Figure 3.

FIGURE 3: ISOLATION TREE FOR UNIVARIATE CASE



Note: we built the tree up to the depth (tree height limit) of RoundUp (log2(sample size)) = 3

Applying Step 3 of the Isolation Forest algorithm described above, we produce the anomaly scores for each data point as listed in Figure 4. For the sake of clarity, we also provide calculation details for the first data point (a potential outlier) and the last data point (a normal instance):

FIGURE 4: ANOMALY SCORE FOR UNIVARIATE CASE

	AV	ANOMALY SCORE	PATH	NUMBER OF DATA POINTS PER LEAF
x0	**325,380**	**0.7741**	**'R'**	**6, 1**
x1	306,293	0.5992	'L', 'L'	6, 5, 1
x2	310,501	0.3405	'L', 'R', 'R'	6, 5, 4, 3
x3	308,657	0.4638	'L', 'R', 'L'	6, 5, 4, 1
x4	310,050	0.3405	'L', 'R', 'R'	6, 5, 4, 3
x5	**310,698**	**0.3405**	**'L', 'R', 'R'**	**6, 5, 4, 3**

For the first data point:

- c(6) equals 2.7066
- E(h) equals 1, because x0 is the only data point in its final node and only one binary split has been needed to separate x0 in our sample tree
- Anomaly score S becomes **0.7741**

For the last data point:

- c(6) equals 2.7066
- Because x5 is one of three data points in its external node (the tree has not been built any further due to the height limit parameter allowing reductions of computational complexity), we are going to need the c(3) adjustment term in order to calculate E(h):
  - c(3) equals 1.2074
  - E(h) equals 3 + c(3) = 4.2074
- Anomaly score S becomes **0.3405**.

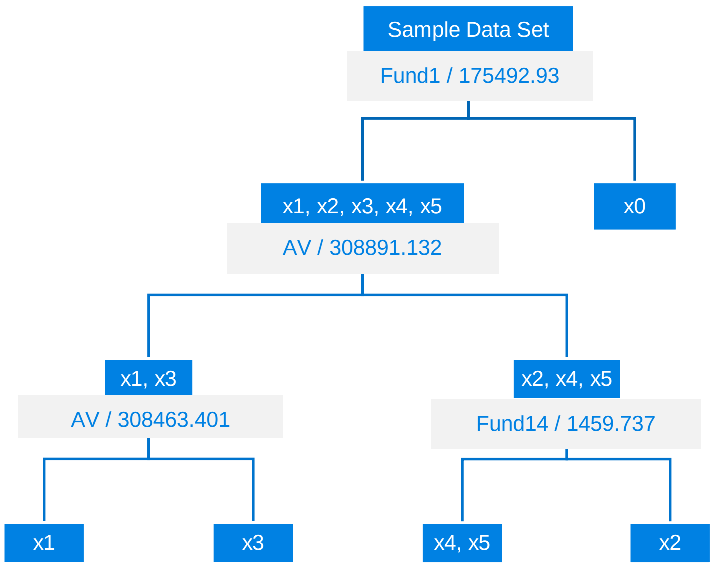
Note that, as mentioned above, we have merely considered one isolation tree in this initial example. Another sampling would typically lead to a different isolation tree, and this is why the algorithm considers a whole Isolation Forest, in order to produce stable average estimates for anomaly scores. In other words, the purpose of this initial example has been merely to illustrate how the formulae in Step 3 of the algorithm works. In the following section, we expand this little example further in order to obtain a more realistic view of the Isolation Forest algorithm.

MULTIVARIATE CASE

Let us now extend the univariate case study above by also considering the evolution of sub-funds—Fund1 to Fund14—over time, not just the evolution of the overall account value. While our multivariate problem is of dimension 15, let us visualize the approach in dimension 3, restricting ourselves to AV, Fund1, and Fund14 only, before returning to the actual 15-dimensional problem.

Once again, we apply the Isolation Forest algorithm described in Section 2 above. Note that, this time, binary splits are carried out in any of the three dimensions (AV, Fund1, and Fund14), rather than in just the one dimension in the univariate case above. See Figure 5 for an illustration via a sample isolation tree, where we denote for each node *in which randomly sampled dimension* and at which randomly sampled value the corresponding binary split has taken place:

FIGURE 5: ISOLATION TREE FOR MULTIVARIATE CASE



Next, we can calculate the anomaly scores for this tree—for the results, see Figure 6.

FIGURE 6: ANOMALY SCORES FOR MULTIVARIATE CASE

	AV	FUND 1	FUND14	ANOMALY SCORES	PATH	NUMBER OF DATA POINTS PER LEAF
x0	**325,380**	**181,679**	**1,439**	**0.7741**	**'R'**	**6, 1**
x1	306,293	172,234	1,442	0.5992	'L', 'L'	6, 5, 1
x2	310,501	175,110	1,464	0.3405	'L', 'R', 'R'	6, 5, 4, 3
x3	308,657	173,885	1,453	0.4638	'L', 'R', 'L'	6, 5, 4, 1
x4	310,050	174,831	1,459	0.3405	'L', 'R', 'R'	6, 5, 4, 3
x5	310,698	175,227	1,460	0.3405	'L', 'R', 'R'	6, 5, 4, 3

Please remember that the Isolation Forest algorithm would require us to repeat this calculation over, say, 15 trees and produce average anomaly scores. However, let us skip this step and instead focus on the question how we could enhance the analysis performed so far in order to make a more informed decision as to whether or not the "outlier candidate" data point x0 would indeed constitute an outlier.

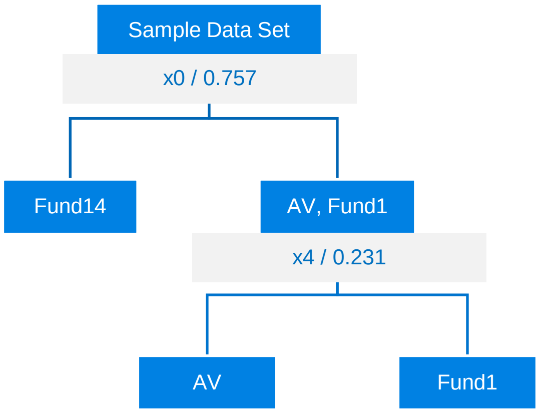
It turns out that we could "transpose" our multivariate data set by considering a multivariate Isolation Forest problem in the dimensions x0,…,x5 with data points being AV, Fund 1,…,Fund 14. However, in order to infer something useful from this angle, we should reflect the fact that the values of different sub-funds are of different orders of magnitude. Therefore, we perform a minimum/maximum normalization on these data before applying the Isolation Forest algorithm. Under this normalization, the minimal value of a variable is replaced by 0, the maximal value is replaced by 1 and intermediate values are replaced by appropriate values between 0 and 1, reflecting the original distances to the minimum and the maximum, as shown in Figure 7.

FIGURE 7: TRANSPOSED DATA AFTER NORMALIZATION

	AV	FUND1	FUND14
x0	**1.0000**	**1.0000**	**0.0000**
x1	0.0000	0.0000	0.1216
x2	0.2204	0.3045	1.0000
x3	0.1238	0.1748	0.5834
x4	0.1968	0.2749	0.7947
x5	0.2308	0.3169	0.8410

Now we can meaningfully apply the Isolation Forest algorithm to our “transposed” data points of AV, Fund1, and Fund14 in the multivariate world of dimensions x0.,…,x5. A sample isolation tree for this problem is displayed in Figure 8.

FIGURE 8: ISOLATION TREE FOR TRANSPOSED DATA



Next, we can calculate the anomaly scores for this isolation tree via the usual approach and obtain the results shown in Figure 9.

FIGURE 9: ANOMALY SCORES FOR TRANSPOSED DATA

	X0	X1	X2	X3	X4	X5	ANOMALY SCORES	PATH
AV	1	0	0.2204	0.1238	0.1968	0.2308	0.3172	‘R’, ‘L’
Fund1	1	0	0.3045	0.1748	0.2749	0.3167	0.3172	‘R’, ‘R’
Fund14	**0**	**0.1216**	**1**	**0.5834**	**0.7947**	**0.8410**	**0.5632**	**‘L’**

In this simple three-dimensional example, Fund14 has been assigned a rather high anomaly score compared to AV and Fund1. This makes intuitive sense, as both AV and Fund1 attain their maximal values at the time period x0 while Fund14 does not do so. Of course, the reader might raise a question whether or not the anomaly score of 0.5632 were high enough to be called an outlier. We will return to this question in the next section.

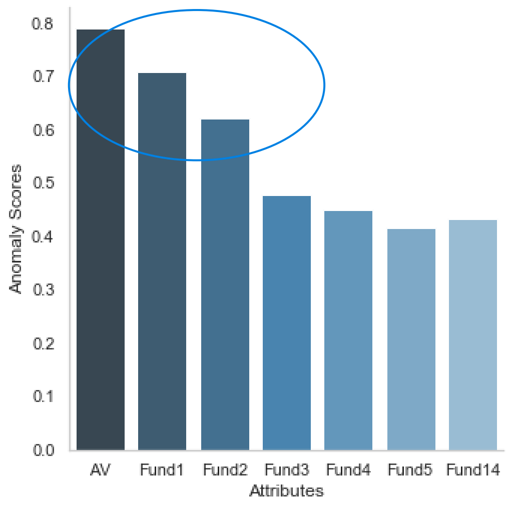
If we apply the procedures explained above in dimension 3 to our 15-dimensional problem featuring AV and all 14 sub-funds from Fund1 to Fund14, consider the six-dimensional transposed problem in dimensions x0-x5 as above and obtain the anomaly scores for the data points AV,…,Fund14 shown in Figure 10.

FIGURE 10: ANOMALY SCORES FOR 15-DIMENSIONAL DATA

	AV	FUND1	FUND2	FUND3	FUND4	FUND5	FUND..	FUND14
x0	325,380	181,679	62,365	2,396	20,623	784	..	1,439
x1	306,293	172,234	57,567	2,397	20,688	787	..	1,442
x2	310,501	175,110	58,572	2,457	20,966	799	..	1,464
x3	308,657	173,885	58,159	2,418	20,883	796	..	1,453
x4	310,050	174,831	58,558	2,430	20,975	801	..	1,459
x5	310,698	175,227	58,747	2,433	20,999	802	..	1,460
Anomaly Score	**0.7909**	**0.7080**	**0.6219**	**0.4771**	**0.4488**	**0.4162**	**..**	**0.4336**

We can visualize these anomaly scores as shown in Figure 11. Please note that we are omitting the Funds 6-13 from this graph for the ease of readability.

FIGURE 11: ANOMALY SCORES FOR 15-DIMENSIONAL DATA



In this example, we can clearly see that the corresponding anomaly scores for AV, Fund1, and Fund2 are larger than the ones of the other funds. In other words, given historical movements of the AV and all its funds, the historical movements of Fund1 and Fund2 seem to be off the trend displayed by all the other funds.

Section 4: Practical recommendations and conclusion

SUBSAMPLING

As proposed in Liu, F.T. et al., in dealing with large data sets, the subsampling technique is highly recommended. By subsampling, we are simply choosing a rather smaller subset of the initial data set and generating our Isolation Forest from the subsample.

This subsampling technique alleviates the following two issues:

- Swamping: There are just too many data points so that normal instances can be “close” to anomalies. In other words, it is expected in this case that a large number of partitions are needed to isolate anomalies. It's much harder to differentiate the anomalies against normal instances.
- Masking: There are so many anomalies they form an anomaly cluster, so to speak. In other words, again, a large number of partitions to isolate anomalies is expected due to the rather dense and large anomaly cluster.

DIMENSIONALITY REDUCTION

A real-life outlier detection problem may often be a high-dimensional one. In that case, it may be possible to use some dimensionality reduction techniques prior to applying the Isolation Forest algorithm. For example:

- Principal component analysis (PCA): A widely used unsupervised linear dimensionality reduction technique. It essentially reduces the initial dimension to a few factors that have significant explanatory power
- Autoencoder (AE): A rather modern unsupervised linear/nonlinear dimensionality reduction technique. The gist of this technique is ultimately the same as that of the PCA. However, AE uses artificial neural net structures. This inherently allows the model to take into account the nonlinearity across initial factors.

STATISTICAL CRITERION

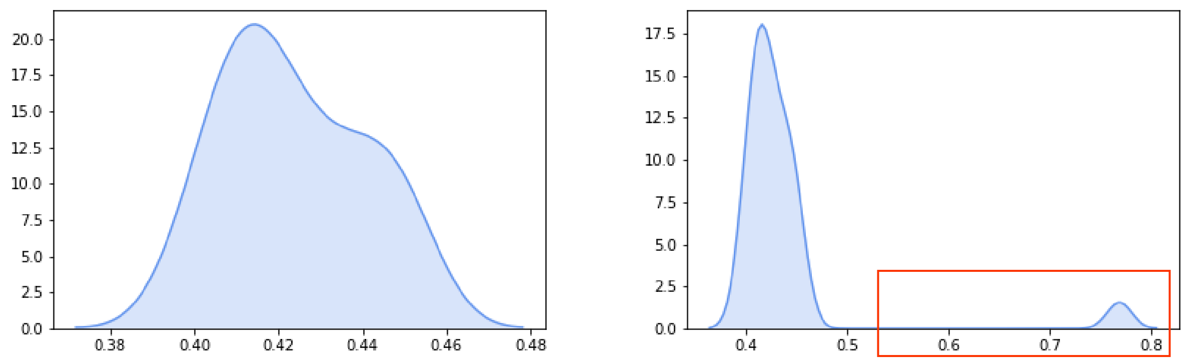
For a practitioner interested in implementing the Isolation Forest algorithm, the main question is how to embed this algorithm in an automated reporting solution. While human eyes can spot some outliers in an “intuitive” way, an automated solution must rely on quantitative ways of determining for each data point whether or not it is an outlier.

The most obvious approach would be to implement a “point threshold”—a data point would be considered an outlier if and only if its anomaly score is above a certain threshold, such as 0.75. A more refined approach would be to consider a combination of a “point threshold” and a statistical one. For example, we could consider the distribution of the anomaly scores and examine for each data point how the deletion of this data point would impact the distribution. A possible statistical criterion could be based on the kurtosis:

$$kurtosis(S[T]) > kurtosis(S[t]) + Significance\ Gap$$

Here, $S[T]$ is a vector of *all* anomaly scores including a potential outlier while $S[t]$ is a vector of anomaly scores for all but excluding the potential outlier and Significance Gap is a positive user-defined parameter. In Figure 12, we can see how the kurtosis on the right diagram is significantly higher than that on the left diagram, because a data point featuring a high anomaly score is included in the former but not in the latter.

FIGURE 12: DIFFERENT KURTOSIS



Note: The underlying data for this visualization is not relevant to the data used in the previous sections. The data was used merely to illustrate the concept of kurtosis.

EXPERT JUDGMENT

Another important practical consideration is the degree of expert judgment required to use the Isolation Forest algorithm. As discussed in Section 2 above, the user would have to specify a few parameters such as the total number of trees or the tree depth. A possible approach is to set the total number of trees in the forest at 20 to 30 and to set the simulated tree length to the logarithm of the sample size:

- According to Liu, F.T. et al., the algorithm converges well before the forest size of 100 and we typically produce good results using a forest size of 20 to 30 or even 15 trees.
- Generation of long trees should not be necessary to detect *outliers*.

That said, the user should be aware of the following caveat: If the movements of the underlying data in a given vector space are volatile and/or there are known data quality issues, then the Isolation Forest algorithm would not necessarily be able to reliably detect outliers.

DATA SETS CONTAINING CATEGORICAL VARIABLES

So far, we have only considered continuous data in the sections above, but real-life problems often contain mixed data sets featuring both continuous and categorical variables.

Therefore, we would also like to indicate how to apply the Isolation Forest algorithm to a problem that contains both continuous variables and categorical ones. Please refer to the appendix for a rather detailed example.

CONCLUSION

The Isolation Forest algorithm explicitly isolates anomalies instead of profiling normal instances, unlike more traditional outlier detection algorithms. The gist of the algorithm is that the “fewer and different” data points will display significantly shorter isolation tree paths, closer to the root of each tree, as opposed to the “more and common” data points.

The algorithm works well with both small and large data sets. It performs well on smaller data sets because it does not need to train a model based on a large number of normal instances.

Equally, it is capable of efficiently scaling for good results and performance on larger data sets, due to the subsampling step. Last but not least, it has been shown that the Isolation Forest algorithm features a linear run-time complexity and performs better on high-dimensional problems than typical distance/density-based approaches.

We conclude our paper by listing the main practical benefits of using the Isolation Forest algorithm:

- The Isolation Forest unlocks insights from large and small underlying data
- It can deal with both continuous and categorical data
- The algorithm only requires a moderate degree of expert judgment and hence lends itself well to an automated application

We have seen the Isolation Forest improve the robustness of outsourcing solutions. More generally speaking, this algorithm can help enhance any financial reporting process by detecting bad input data in an automated way.

Appendix

For the sake of simplicity, we would like to illustrate our suggested treatment of categorical variables on a simple two-dimensional example rather than in theoretical terms.

Figure 13 displays a contrived weather data set featuring just two dimensions, namely a continuous variable ***temp*** (temperature) and a categorical variable ***type*** with four possible choices, namely “cloudy,” “rainy,” “snowy,” and “sunny.”

FIGURE 13: WEATHER DATA FEATURING TWO DIMENSIONS

	TEMP	TYPE
x0	20	cloudy
x1	28	sunny
x2	22	rainy
x3	33	snowy
x4	32	sunny
x5	16	cloudy

In order to apply the algorithm discussed above to this problem, we would have to somehow assign numerical values to the weather types. A naïve way of doing this would be via the alphabetical order—e.g. we might assign 0 to “cloudy,” 1 to “rainy,” 2 to “snowy,” and 3 to “sunny.” The reader might already realize why this would not be a great idea—indeed, many binary splits in the dimension “type,” for example a split at 1.5—would not discern between the weather types “snowy” and “sunny,” although these weather types are very different from one another. So exactly how does this problem differ from the purely continuous ones discussed above? If the variable “type” had been a continuous one, we would have been able to implicitly assume that its values are all logically ordered, so that, e.g., a value of 2 would be genuinely “closer” to a value of 3 than, say, a value of 0 in the Euclidean sense. However, this implicit assumption is apparently not true in the nominal categorical case. In other words, one artificial numerical ordering of the nominal categorical variable would not be genuinely “better” than the others.

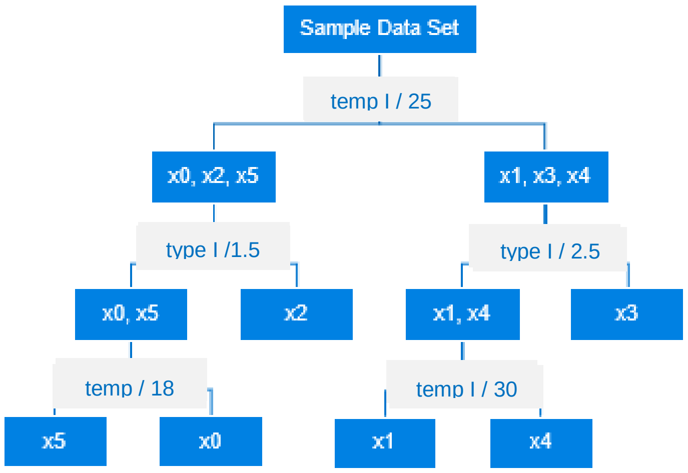
Therefore, we are facing the question of what would happen if we tried out two different mappings assigning numerical values to our nominal categorical weather types. Figure 14 shows one such artificial labelling for our nominal categorical variable—label encoding.

FIGURE 14: LABEL ENCODING FOR WEATHER DATA

	TEMP	TYPE	RANDOM LABELLING OF WEATHER TYPE
x0	20	cloudy	0
x1	28	sunny	1
x2	22	rainy	2
x3	33	snowy	3
x4	32	sunny	1
x5	16	cloudy	0

Making use of this labeling, we can generate the isolation tree shown in Figure 15.

FIGURE 15: ISOLATION TREE FOR WEATHER DATA



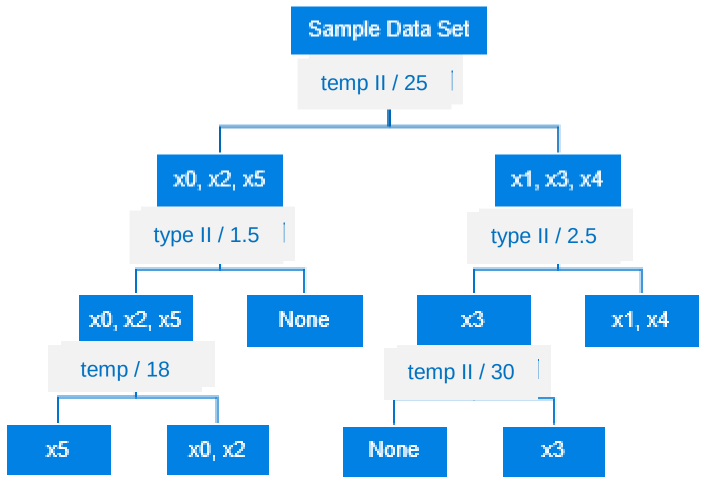
So far, so good, it might seem. But what would happen if we used a different artificial labeling for our nominal categorical variable? Figure 16 shows such an alternative labeling.

FIGURE 16: ALTERNATIVE LABELING FOR WEATHER DATA

	TEMP	TYPE	RANDOM LABELLING OF WEATHER TYPE
x0	20	cloudy	0
x1	28	sunny	3
x2	22	rainy	1
x3	33	snowy	2
x4	32	sunny	3
x5	16	cloudy	0

Applying this labeling, we can generate the isolation tree shown in Figure 17.

FIGURE 17: ISOLATION TREE FOR WEATHER DATA APPLYING ALTERNATIVE LABELING



As we can see, we have now obtained an isolation tree that differs from that shown in Figure 15. There is no “canonical” assignment of numerical values to the nominal categorical variable that would be “better” than the others, and an application of a particular assignment throughout the process would introduce spurious bias.

Therefore, our suggestion is to randomly reshuffle the assignment of numerical values to the nominal categorical variable at each child node of the tree. This additional randomization step will enable the algorithm to properly deal with categorical variables.

Milliman

Milliman is among the world’s largest providers of actuarial and related products and services. The firm has consulting practices in life insurance and financial services, property & casualty insurance, healthcare, and employee benefits. Founded in 1947, Milliman is an independent firm with offices in major cities around the globe.

milliman.com

CONTACT

Hyunsu Kim
hyunsu.kim@milliman.com

Michael Leitschkis
michael.leitschkis@milliman.com