Chapter 21.

DETECTING OUTLIERS IN CLAIM DATA USING MACHINE LEARNING

In general, outliers are items that are significantly different from the majority of other items they may be compared to. In this chapter, we use the term anomaly synonymously, though also often to indicate a single unusual property of an item: the individual oddities that make an item an outlier.

Outlier detection refers to the process of finding items that are unusual[611]. For tabular data, this usually means identifying unusual rows in a table; for image data, unusual images; for text data, unusual documents, and similarly for other types of data. The specific definitions of "normal" and "unusual" can vary, but at a fundamental level, outlier detection operates on the assumption that the majority of items within a dataset can be considered normal, while those that differ significantly from the majority may be considered unusual, or outliers. For instance, when working with a database of claims, we assume that the majority of claims represent normal behaviour, and our goal would be to locate the claim that stands out as distinct from these.

In statistical theory, outliers can arise due to measurement errors, an error in data transmission, elements outside the population being incorrectly included in the population, a flaw in an assumed theory, or genuine variability. Identifying outliers is crucial, as they can skew statistical analysis and lead to misleading conclusions. Applications of outlier detection include fraud detection, bot detection on social media[612], network security, financial auditing, regulatory oversight of financial markets, medical diagnosis, astronomy, data quality and the development of autonomous vehicles.

It's important to note that not all outliers are necessarily problematic, and in fact, many are not even interesting. Outlier detection can be seen as being not merely interested in removing noise but also in finding interesting database objects deviating in their behaviour considerably from the majority and, as such, providing new insights. Inconsistency can mean that the data object is contaminated from a different distribution than the model considered to describe

[611] Kennedy, B. (2024). *Outlier detection in Python.* Shelter Island: Manning.

[612] Rahman, M. S., Halder, S., Uddin, M. A. et al. (2021). An efficient hybrid system for anomaly detection in social networks. *Cybersecurity, 4(1)*, 10.

the data. But inconsistency could also mean that the pre-supposed model is not describing the data as well as was assumed when selecting the model. Both conclusions can bear rather significant repercussions on the interpretation of the given observations.

There are several ways to find fraudulent records in tabular data. If we have economic or financial data, we can use a group of methods based on Benford's law.[613] However, its weakness is sensitivity to sample size. In this chapter, we will show how the same task can be solved using unsupervised learning techniques for detecting outliers from insurance data. By applying both techniques, the pool of suspicious records would be smaller, which could lead to savings in audit work.

1. OUTLIER DETECTION'S PLACE IN MACHINE LEARNING

Outlier detection is a machine learning technique, which means it learns from data rather than using rules created by people. There are various outlier detection techniques in machine learning, which are categorized as supervised or unsupervised methods.

Supervised learning uses a labelled training dataset to understand the relationships between input and output data[614] (identify outliers in our case). Data scientists manually create training datasets containing input data along with the corresponding labels. Supervised learning trains the model to apply the correct output to new input data in real-world data. In that setting, we are given a set of data, possibly a spreadsheet of records, a collection of text documents, a time sequence of instrument readings, or audio files. What makes a problem supervised is the fact that the data includes what are called labels. As an example, assume that we have a spreadsheet of records from past insurance claims. The labels will in some way describe each record. For example, let's consider a scenario in which we have tabular data with the following fields: date, claim, agent, description and agency, and the labels may refer to the type of record - „fraudulent" or „non-fraudulent". Given this, the goal would be to utilize supervised machine learning in order to create a predictor. This predictor, more

[613] Azdejković, D. (2024). Poboljšanje efikasnosti nekih testova forenzičke analitike. In: *2024 Perspektive razvoja forenzičko-računovodstvenih kapaciteta: izazovi za javni i korporativni sektor*. Belgrade: University of Belgrade, Faculty of Economics and Business, pp. 335-345.

[614] Van Wyk, F., Wang, Y., Khojandi, A., & Masoud, N. (2019). Real-time sensor anomaly detection and identification in automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, *21*(3), pp. 1264-1276.

specifically, is referred to as a *classifier* if the labels represent categories (also known as classes); the classifier will learn to predict the class labels from the record (in this case, predicting the type of claim in a record). And the predictor is referred to as a *regressor* if the labels are numeric values (for example, if we had labels representing the pictures, such as the number of distinct cars in the picture); the regressor will learn to predict these numeric values from the images. The labels here are said to supervise the learning process.

Supervised learning in insurance often revolves around using labelled data to make predictions, assess risk, or enhance customer experience. Here are some examples from the insurance industry:

- *Claim fraud detection* - historical data from past insurance claims, labelled as "fraudulent" or "non-fraudulent," is used to train a machine learning model. By identifying patterns and features associated with fraudulent claims (e.g., unusual claim amounts, repeated claims from the same individual, or inconsistencies in provided information), the model can predict the likelihood of fraud in new claims[615].
- *Premium pricing* - models trained on customer data (age, health, driving records, etc.) labelled with premium amounts can help insurers determine the appropriate premium rates for new customers[616].
- *Risk assessment* - using labelled data about accidents or policy lapses, supervised learning models predict the likelihood of these events for potential clients[617].
- *Customer segmentation* - learning from labelled data about customer behaviour and preferences, insurers group clients into categories for targeted marketing[618].

However, often when we are working with data, there are no labels, just the data itself. So, in *unsupervised learning*, we do not label the data with which we want to train the model. The model learns through training itself from the data. Here are some examples from the insurance field:

---

[615] Debener, J., Heinke, V., & Kriebel, J. (2023). Detecting insurance fraud using supervised and unsupervised machine learning. *Journal of Risk and Insurance*, *90*(3), pp. 743-768.

[616] Gu, S., Kelly, B., & Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*, *33*(5), pp. 2223-2273.

[617] Paltrinieri, N., Comfort, L., & Reniers, G. (2019). Learning about risk: Machine learning for risk assessment. *Safety science*, *118*, pp. 475-486.

[618] Ozan, Ş. (2018). A case study on customer segmentation by using machine learning methods. In: *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, IEEE, pp. 1-6.

- *Customer segmentation*: Using customer data without predefined labels, an unsupervised learning model can group customers based on shared characteristics (e.g., purchasing behaviour, claim history, demographics). These segments can help insurers design targeted marketing strategies, offer personalized policies, or identify underserved customer groups[619].
- *Risk profiling*: Clustering policyholders based on risk levels, which can assist in better pricing and policy design[620].
- *Fraud detection*: Identifying unusual patterns or anomalies in claim data that could indicate potential fraud[621].
- *Policy optimization*: Grouping similar types of insurance policies to identify trends and optimize product offerings[622].

When clustering data, we find sets of items that appear to be similar to each other. A clustering algorithm for a given spreadsheet of records would examine the records and divide them into a small set of clusters, based on perhaps the agent's name, the amount of claim and other properties of the record. This would be done such that each cluster is internally consistent (the records in each cluster are similar to each other) and the different clusters are unlike each other.

Outlier detection, on the other hand, identifies the most unusual items in a dataset. This is based on something quite powerful: learning the fundamental patterns of a given set of data, what constitutes normal for this data, and what items diverge most strongly from these patterns.

Prediction and outlier detection do different things—both are useful, but quite different. When labels are available, it can often be more useful to create a predictive model than to perform unsupervised analysis such as clustering or outlier detection. We can train a classifier to learn what types of records represent fraudulent records. Then, given any new record, the classifier can predict

---

[619] Shen, B. (2021). E-commerce customer segmentation via unsupervised machine learning. *The 2nd international conference on computing and data science*, pp. 1-7.

[620] Bao, W., Lianju, N., & Yue, K. (2019). Integration of unsupervised and supervised machine learning algorithms for credit risk assessment. *Expert Systems with Applications*, *128*, pp. 301-315.

[621] Rai, A. K., & Dwivedi, R. K. (2020). Fraud detection in credit card data using unsupervised machine learning based scheme. In: *2020 international conference on electronics and sustainable communication systems (ICESC)*, IEEE, pp. 421-426.

[622] Shen, J., Zhao, H., Zhang, W., & Yu, Y. (2020). Model-based policy optimization with unsupervised model adaptation. *Advances in Neural Information Processing Systems*, *33*, pp. 2823-2834.

what type of record it is. However, what a classifier cannot do well is recognise when it is given an unusual record, perhaps a record not belonging to one of the classes it was trained to predict. These are problems outlier detectors can solve: recognising which of the data we may be faced with, if any, is unusual.

Definitions of outliers

We will try to more precisely define what outliers are, although it is possible to a degree since outliers are a fuzzy concept. This has some advantages, as it allows us to approach outlier detection from many different fields, but it also carries a lot of vagueness. The fact is that the context makes a great deal of difference. What might be noise in one context may be a signal in another. Here are a few definitions that we often see in academic literature, each useful, but each a little vague.

Grubbs[623] stated "*an outlying observation, or outlier, is one that appears to deviate markedly from the other members of the sample in which it occurs.* "

Barnett and Lewis[624] used the following definition: "*An observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data*." This broadens the concept of outliers as it adds the idea of collective outliers as well as single items (e.g., single credit card purchases) being unusual; a set of several (e.g., a series of credit card purchases over a few hours) may be an outlier, even if each item in a set is typical.

One of the most cited definitions of outliers is from Hawkins[625]: "*An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism.*" This definition is based on an important idea: that there is some process, or set of processes, generating the data and that they operate within certain bounds. Other data, generated by other processes, can become included in a dataset and would then be considered anomalous. For example, we could consider the payment processes in one company that include the purchase of office supplies and the payment of utility services. Each process will generate a set of costs, which will be included in the cost table. However, utility payment processes will be far less frequent and larger, so they could be considered outliers, even though they are completely

[623] Grubbs, F. (1969). Procedures for Detecting Outlying Observations in Samples. *Technometrics, 11*(1), pp. 1-21.

[624] Barnett, V., & Lewis T. (1994). *Outliers in Statistical Data*. New York: John Wiley & Sons.

[625] Hawkins, D. (1980). *Identification of Outliers*. Chapman & Hall.

legitimate. On the other hand, it can result in an employee entering incorrect data, which may lead to possible fraud. As these are different processes, they create records that are fundamentally different.

Despite the common need to identify unusual data points and despite these intuitive definitions, no universally accepted definition of specifically what qualifies as "unusual" exists. This means it's very difficult to say what items should be flagged as outliers. With no labels, there is no systematic way to evaluate any outliers flagged, and without a concrete definition, there is no definite way to create labels. Outlier detection is, then, highly subjective.

Take Figure 1, for example. It shows eight wine glasses. We might ask: Which glasses are unusual? Just by looking at the picture, we can begin to see many ways in which we might consider some of the glasses unusual. They differ in size, shape, the colour of the wine in them, and so on. Different people might come to different conclusions based on the picture. Some might argue that all the glasses are similar, so none of them is unusual. Others might say that the one in the second row on the far right is different from the others, so it could be considered unusual, and so on.

Figure 1. Which glass is an outlier?

Source: https://macyswineshop.com

Trends in identifying outliers

Outlier detection has gone through three major stages, starting with traditional, simple statistical methods (for example, based on *z*-score), going back many decades[626]. These were straightforward and could be computed by hand for small datasets, but were limited to finding unusually small or large values in single sequences of numbers and don't extend well to tables of data.

[626] Aguinis, H., Gottfredson, R. K., & Joo, H. (2013). Best-practice recommendations for defining, identifying, and handling outliers. *Organizational research methods*, *16*(2), pp. 270-301.

These were followed by machine learning-based methods, largely from the last 30 or so years, developed as computer hardware improved, digitised data became common, and machine learning techniques improved. We will refer to these here as traditional machine learning to distinguish them from the third major stage, which has advanced significantly in the last several years: the development of deep learning-based methods—methods based on deep neural networks. This has been crucial in some areas; outlier detection with vision, text, or audio would not be possible without deep learning. All three forms of outlier detection now play a valuable role in analysing data and searching for anomalies.

Where we have the most opportunity to improve traditional machine learning-based outlier detection with tabular data is by improving the collection and sharing of data. As outlier detection is based on data, it benefits from the explosion in data we are now producing and collecting. We now have a wealth of data to compare against, which allows us to develop a better sense of what is normal and what is not, which allows outlier detectors to be much more sensitive and accurate. A great deal of this data, though, is proprietary, owned by single organisations.

2. THE OUTLIER DETECTION PROCESS

The outlier detection process typically involves several key steps:

1. *Understanding the data*: Begin by exploring the dataset to understand its structure, distribution, and context. This includes identifying the type of data (numerical, categorical, etc.) and any domain-specific considerations.
2. *Preprocessing*: Clean the data by handling missing values, removing duplicates, and normalising or scaling features. This ensures that the data is ready for analysis.
3. *Choosing a detection method*: Select an appropriate method based on the dataset and the type of outliers you expect. Common methods include:
    - Statistical techniques like *z*-scores or Interquartile Range (IQR).
    - Machine learning methods like Isolation Forests, Autoencoders, or clustering algorithms.
4. *Applying the method*: Implement the chosen method to identify potential outliers. This may involve setting thresholds or parameters to define what constitutes an outlier.
5. *Validating results*: Review the identified outliers to ensure they are truly anomalous and not just unusual but valid data points. Domain knowledge is crucial here.

6. *Handling outliers*: Decide how to handle the outliers—whether to remove them, transform them, or analyse them separately. The approach depends on the specific goals of the analysis.
7. *Iterative refinement*: Repeat the process as needed, especially if new data is added or the context changes.

This process can be tailored to specific applications, such as fraud detection, quality control, or predictive modeling.

3. SIMPLE OUTLIER DETECTION

Outlier detection has a long history in statistics, and many well-established, simple methods exist, but they are usually univariate and often assume specific data distributions, typically that the data have a Gaussian distribution or approximately a Gaussian distribution. They are designed specifically to find extreme values, the unusually small and large values in sequences of numeric values. These are the easiest outlier tests to understand and provide a good background for the machine learning based approaches we will focus on later. Statistical methods have some significant limitations. They work on single columns of numeric data and often don't extend well to tables.

One-dimensional numeric outliers

Statistical methods to identify outliers in a sequence of numbers include tests based on *z*-scores, interquartile range (IQR), and median absolute deviation (MAD). All tests are based on a simple approach: we calculate the score for value in data and flag values over or below a predefined threshold.

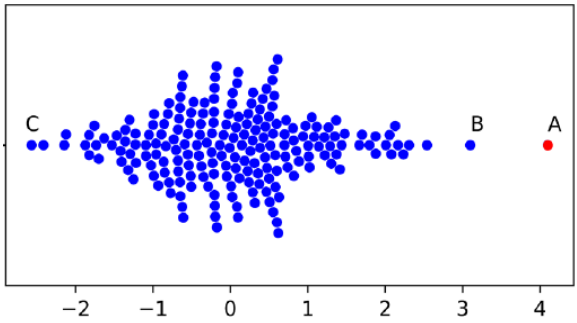
For a given set of numbers, we calculate the z-score for value x_i as

$$z_i = \frac{x_i - \underline{x}}{\bar{s}}$$

Any value with a z-score below -3 and above 3 we flag as an outlier. The threshold is based on the fact that if some data has a Gaussian distribution, then less than 0.3% of the data is above 3 or below -3. The main disadvantage of the *z*-score method is its sensitivity to outliers. The presence of an outlier in the data significantly affects the calculation of the arithmetic mean and standard deviation, and thus the ability to find other outliers. In particular, the *z*-score method allows masking and swamping effects. This is shown in Figure 2: if *A* is not present in the data, then *B* is not an outlier (masking), but if *A* is present in the data, then *C* is an outlier.

A lesson from the z-score method is that we should be very careful when using mean and standard deviation in detecting outliers. Most of the methods in common usage are much more robust to outliers. If we calculate the mean and standard deviation after removing some percentage of the top and bottom values in the data, we can get a more robust method.

Figure 2. Swamping and masking



Source: Kennedy (2024), op. cit., p. 28.

Interquartile range test, as the name suggests, uses the interquartile range $IQR = Q_3 - Q_1$ to define a bottom threshold $Q_1 - \theta \cdot IQR$ and upper threshold $Q_3 + \theta \cdot IQR$, where θ is a parameter. Although John Tukey[627] who developed the test had a long tradition of using $\theta = 1.5$, today's standard is $\theta = 2.2$. The main advantage of the interquartile range test is that it is much more robust to outliers than *z*-scores.

Finally, if we want a much more robust method, we have to use median absolute deviation (MAD). Let $x = (x_1, x_2, \dots, x_n)$ be the set of observed values, then MAD is defined by:

$$MAD(x) = Med\{|x - Med(x)\}.$$

where $Med(x)$ is the median. To determine if a value is an outlier using MAD, we calculate the so-called normalised MAD (MADN):

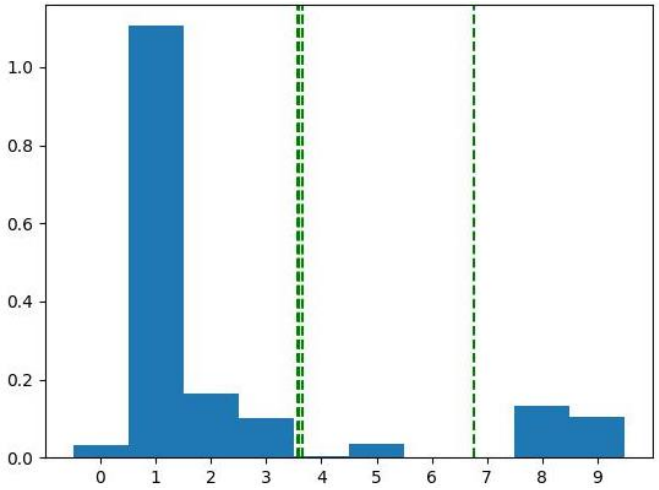
$$MADN(x_i) = \frac{0.6745\big(x_i - Med(x)\big)}{MAD(x)}$$

The constant 0.6745 in this formula is used to make it equivalent to the z-score formula when the data are normally distributed. With MADN, usually 3.0 or 3.5 is used for the threshold.

[627] Tukey, J. W. (1977). *Exploratory Data Analysis*. Massachusetts: Addison-Wesley.

Previous methods are intended to identify extreme values and will miss so-called *internal outliers*. Internal outliers exist in multimodal distributions or distributions with gaps. Histograms are possibly the easiest method to identify internal outliers, especially if we use, for example, the pandas library from Python for constructing a histogram object. The idea of histograms is to divide the space into sets of equal-width bins and calculate the count of each bin. This provides clear insight into the distribution of the same and allows us to flag bins with unusually low counts as outliers. Drawback in using histograms is sensitivity to the number of bins, so we must be very careful in their construction. In Figure 3, we can see the histogram of the hue-mean feature from the segment dataset on the open machine learning site[628]. Segment data is divided into 10 histogram bins. The height of each bar indicates the number of records in the bin. Since our histogram is an object from the pandas library, it is not hard to flag bins which have quite low counts. In our case, we flag bins 5 and 8 with two dashed vertical lines. Points in those bins are internal outliers. Bin 6 has no records, and so, no outliers there.

Figure 3. Internal outlier



Source: Kennedy (2024), op. cit., p. 35.

Kernel density estimation (KDE) works similarly to histograms. KDE estimates the density of data points across feature space as a continuous function. Points that lie in areas of low density (i.e., the tails of the distribution) are considered outliers. For a given set of data points $x_1, x_2, \ldots, x_n$, KDE estimates the density at a point x as:

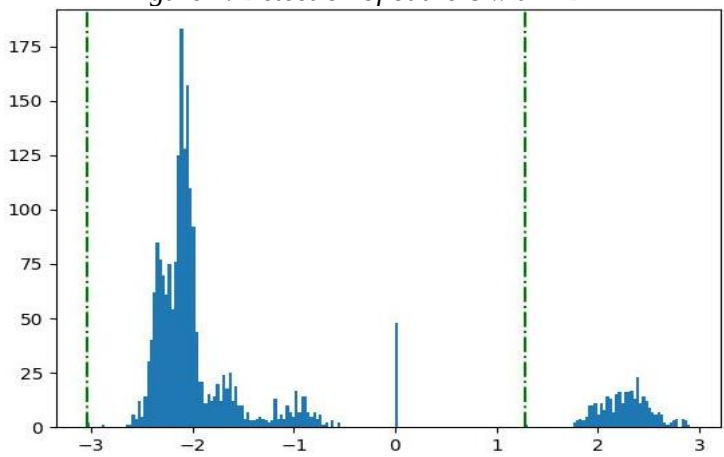
[628] https://www.openml.org/

$$\hat{f} = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

where *K* is a kernel function, and *h* is a smoothing parameter. In other words, an algorithm creates a small kernel, which is shaped as a triangle, rectangle, or Gaussian, where each datapoint is located. This is more robust than histograms, since it is based on many small kernels. Also, the method avoids the need to set the number of bins, but it is sensitive to the width of the kernel.

Using Python library for machine learning, scikit-learn for machine learning, and linear as kernel shape, we can create a KernelDensity object for the dataset. Once we have an estimation of the KDE function, the next challenge is to determine which values are unusually low. Figure 4 shows the points identified (flagged with vertical lines) as outliers in the dataset segment using this method.

Figure 4. Detection of outliers with KDE

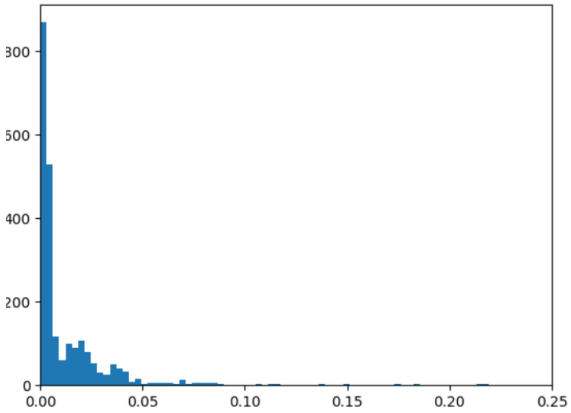


Source: Kennedy (2024), op. cit., p. 38.

Outliers are typically points that are far from other points in the dataset. Since the K-Nearest Neighbors (KNN) method measures the distance from each point to other points, it can be used to detect outliers by examining how far a data point is from its *k* nearest neighbors. Given that the array of distances can be converted into a single number, which will represent the outliers of the point. Usually, converting is done by the mean or the maximum distance. A similar approach to KNN is to cluster the data and flag any points far from the nearest cluster centre. To implement this, we can use the data structure BallTree provided by the scikit-learn package from Python, which can calculate and store the distances between each pair of points in a dataset. The set of scores is not immediately interpretable,

and it is not clear what the cutoff is, so we will obviously focus on just the tail. Figure 5 shows the distribution of max scores using KNN with k=20 for the feature hue-mean from the dataset segment.

Figure 5. Distribution of max scores using KNN with k=20



Source: Author's calculations

One-dimensional categorical outliers

When discussing one-dimensional categorical outliers, we are talking about identifying unusual or rare categories within a single categorical variable. Unlike numerical outliers, which are identified based on their value's distance from the mean or distribution, categorical outliers are identified based on their infrequency or the lack of representation in the dataset.

If a particular category appears very infrequently compared to the other categories, it could be considered an outlier. Similarly, a category that occurs with such overwhelming frequency that it dominates all others could also be flagged as an outlier, although this is less common.

We could identify categorical outliers flagging the *k* rarest values or flagging any value occuring less than some predefined number of times, but that strategy has a weakness since we don't know how many categories we will have, and similarly, we do not know in advance how many instances of each distinct value to expect. We could also flag any value representing less than 1% of rows, but this may overreport, as it could be due to too many outliers.

A better approach is to evaluate the cumulative counts. If we sort unique values from least to most frequent and then calculate the cumulative sum, then we can

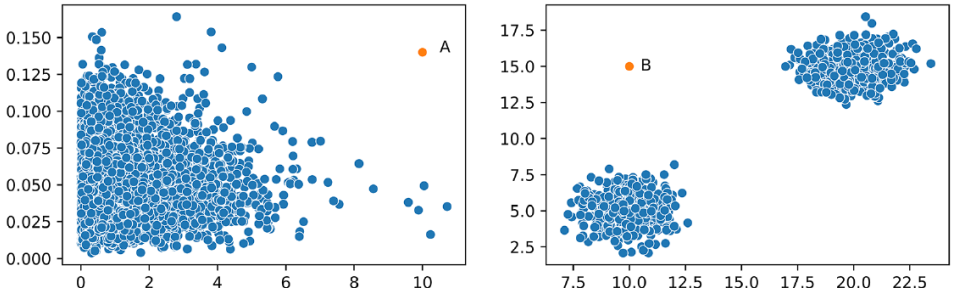
flag values under some threshold, such as 0.5%. This gives us a robust method which works very well in most cases. Another effective method is to apply some statistical methods to the set of counts to flag any unusually small counts.

Multidimensional outliers

In data science, we typically work with data tables and wish to flag unusual rows. Unlike in one-dimensional data, where an outlier is just a value that is too high or too low, multi-dimensional outliers are more complex and can be normal in individual dimensions but still abnormal when considered across all dimensions together. There are only two reasons to flag a row: (1) The row contains one or more unusual values; (2) The row contains one or more unusual combinations of values.

In Figure 6, we can see scatter plots with flagged outliers. Point *A* is an example of an outlier in one dimension (left plot), while point *B* is an example of a rare combination of values (right plot).

Figure 6. Rare combination outlier



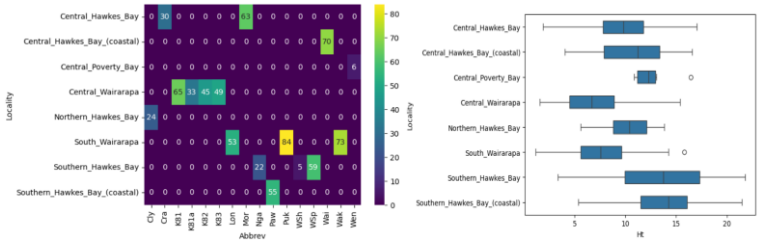
Source: Kennedy (2024), op. cit., p. 45.

Working with tabular data, we may flag rows that have several values or several combinations of values.

A common method for the detection of outliers in table data is visualisation. The general strategy is to look at each feature, use some visualisation method and then also look at each pair of features. Investigating three or more features in a single plot is also possible, but very difficult. For two categorical features, a heatmap is a good choice, for two numeric features, a scatter plot is nice, while for one categorical and one numeric, a box plot could be a very good choice. Figure 7 shows examples of a heatmap and box plot based on two pairs of features from the eucalyptus dataset on the open machine learning site.

In practice, visualisations are used more to understand the data than to flag points as outliers, since, for example, a box plot could be a misleading method if the data is not unimodal.

Figure 7. Examples of visualisation in two dimensions

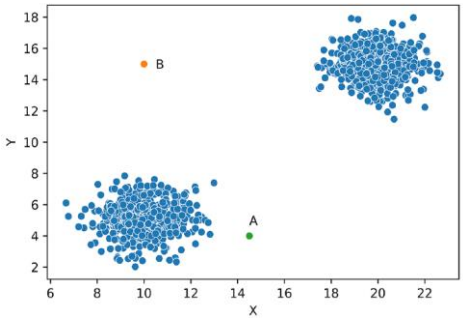


Source: Author's calculations

„Noise“ is a term which we use very often in machine learning. In the context of outlier detection, we use it to refer to points for which it is not clear whether it is are outliers or inliers. Most outlier detectors require setting a threshold or other parameters that will control how many records will be flagged as outliers. This is the way to control how much noise we allow in detecting outliers.

In outlier detection, working with data of at least two dimensions, we must establish the fundamental difference between what we call local and global outliers. The idea is easiest to show on a picture with numeric data.

Figure 8. Local and global outliers



Source: Kennedy (2024), op. cit., p. 51.

In Figure 8, point A is a local outlier since it is likely generated by the same process as points in the nearby cluster. On the other hand, point B is a global outlier, since it is distinct from almost all the data.

4. MACHINE LEARNING BASED OUTLIER DETECTION

While statistical techniques are useful for identifying outliers, they may not always be sufficient due to the following reasons: complex patterns, context dependence, multidimensional data, robustness to noise and dynamic datasets. Machine learning methods have significantly advanced the detection of outliers, offering more robust and adaptive solutions compared to traditional statistical techniques.

Isolation Forest (IF)

The Algorithm Isolation Forest was developed by Fei Tony Liu in 2008[629].This algorithm works by isolating observations using random feature selection and split values, creating decision trees. Outliers, being fewer and more distinct, tend to have shorter path lengths in these trees compared to regular data points.

The algorithm is efficient (linear time complexity $O(n \log \log n)$, scalable for high-dimensional datasets, and requires less memory than other methods[630].

The core idea of Isolation Forest is that anomalies are few and different, and thus, they are easier to isolate in a dataset. If we randomly partition the data, anomalies are likely to be isolated quickly because they differ significantly from normal points. The algorithm of IF has three main parts: Random partitioning of data, path length calculation and anomaly score computation.

In Random partitioning of data, the algorithm builds an ensemble of isolation trees, which are similar to decision trees. Each tree is built by randomly selecting a feature and then randomly selecting a split value between the minimum and maximum value of that feature. This process continues recursively to create branches until: The instance is isolated (i.e., placed alone in a leaf node), or a predefined maximum depth is reached, or the node has only one data point.

The path length is the number of edges (splits) required to isolate a given point. Anomalies tend to have shorter average path lengths in the tree ensemble because they are isolated quickly. Normal points require more splits (i.e., longer paths) to isolate because they are grouped with other similar points.

---

[629] Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. In: *2008 Eighth IEEE International Conference on Data Mining*, Pisa, pp. 413-422.

[630] Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2012). Isolation-Based Anomaly Detection. *ACM Transactions on Knowledge Discovery from Data*, *6*(1), pp. 1-39.

For a point *x*, its anomaly score *s(x,n)* is based on the average path length over all trees:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

where *E(h(x))* is the average path length of point *x* across the forest, *c(n)* is the average path length of unsuccessful searches in a binary search tree, approximated as: $c(n) = 2H(n-1) - \frac{2(n-1)}{n}$ with *H(i)* being the harmonic number: $H(i) \approx ln\ ln\ (i)\ + 0.5772$ (Euler-Mascheroni constant).

Finally, we interpret the score in the following way:

- ≈1: High likelihood of anomaly
- <0.5: Normal
- ≈0.5: Uncertain

The algorithm is implemented in Python using `sklearn.ensemble`.

Local Outlier Factor (LOF)

Local outlier factor is the best-known density-based algorithm used for anomaly detection in datasets, particularly in unsupervised learning contexts, proposed by Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng and Jörg Sander in 2000[631]. The intuition behind this algorithm is that the density around an outlier object is significantly different from the density around its neighbors. It assesses the "local" density deviation of a given data point with respect to its neighbors to determine if it behaves like an outlier.

Here are the steps in the LOF algorithm:

1. For each data point, LOF calculates its local neighborhood using *k*-nearest neighbors (*k-NN*).
2. Find local reachability density (LRD) using the following equation:

$$LRD_k(X) = \frac{N_k(X)}{\sum_{O \in N_k(X)} rd_k(X,O)},$$

[631] Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Identifying Density-based Local Outliers (PDF). In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data (SIGMOD '00)*, New York, NY: Association for Computing Machinery, pp. 93-104.

where $N_k(X)$ is the number of points in the neighborhood of X (could be bigger than k, since multiple objects may have identical distance to X) , $rd_k(X,O) = \{ d_k(O), d(O,X)\}$ is reachable distance from X to O, $d_k(O)$ − k-distance or distance from object to the k-th neighbor and finally d some distance like Euclidean or L_p.

3. The local reachability densities are then compared with those of the neighbors using:

$$LOF_k(A) = \frac{1}{|N_k(A)| \cdot LRD_k(A)} \sum_{B \in N_k(A)} LRD_k(B)$$

which is Local Outlier Factor. A value of approximately 1 indicates that the object is comparable with its neighbors. A value below 1 indicates a denser region, while a value larger than 1 indicates outliers.

LOF detects local outliers rather than just global anomalies and works well in high-dimensional datasets. It can also identify more subtle deviations that traditional distance-based methods may miss. On the other hand, LOF are computationally expensive for large datasets due to k-NN calculations and requires fine-tuning of the k parameter, which affects sensitivity.

The algorithm is implemented in Python using:
`sklearn.neighbours.LocalOutlierFactor.`

5. DETECTING OUTLIERS IN INSURANCE DATA

In this section, we focus on detecting outliers in insurance data and claim incidents. The dataset used for analysis consists of 1000 individual claims, with the primary goal being to present insurance and incident information. The dataset includes 40 different attributes that describe each claim, categorised into four main categories: the insured person, their policy details, incident description, and characteristics of the involved car. The dataset is available on the Machine learning site as well as on GitHub[632]. A Jupyter notebook with all calculations can also be found at the author's GitHub repository[633].

If we want to compare the effectiveness of these two methods, we need a measure. Bearing in mind that the data set was intended for supervised machine learning

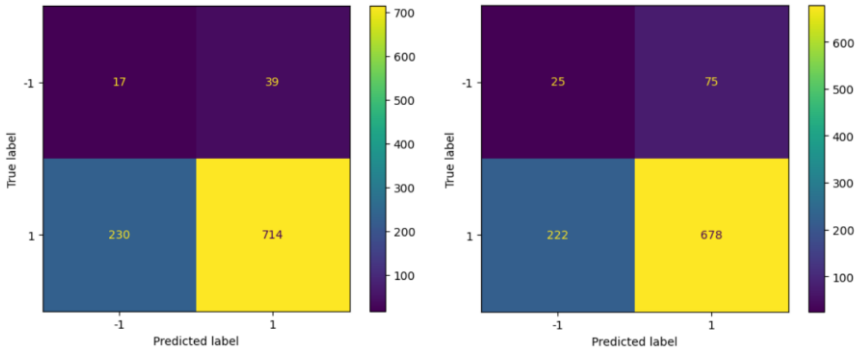
[632] https://github.com/RakeshHansrajani/Insurance_Data_Analysis/blob/main/insurance.csv

[633] https://github.com/dazdejkovic/Outlier-detection-1

for fraud detection, we will use the column labelled ‘fraud’ to construct the confusion matrices.

In the process of data cleaning and preparation, correlated features as well as features with an almost uniform distribution were discarded. Also, only numerical features were used to make the methods comparable. The results in the confusion matrices were obtained by taking the optimal values of the model parameters. For example, for the LOF method, the optimal value of the parameter *k* (number of neighbors) is 60, while for the IF method, the optimal number of estimators (trees) is 500. On this data set, Isolation Forest proved to be a better method than the Local Outlier Factor method (the accuracy of the IF method was 73.6%, and the accuracy of the LOF method was around 71.13%). The next two figures show the confusion matrices for these two methods.

Figure 9. Confusion matrices for IF and LOF methods (-1 indicates anomaly)



Source: Author's calculations

Dealing with real data, an insurance expert should analyse each noted anomaly in detail and make the final judgment as to whether it is an anomaly or not.

* * *

While statistical techniques are useful for identifying outliers, they may not always be sufficient due to a variety of reasons:

1. *Complex patterns*: Statistical methods often rely on assumptions about the data, like normality or linearity. In real-world scenarios, datasets can have complex, non-linear relationships that simple statistical methods might not capture.
2. *Context dependence*: Outliers are highly context-sensitive. A value that is extreme in one context might be perfectly valid in another. Statistical

techniques may fail to incorporate domain knowledge or the context behind the data.

3. *Multidimensional data*: When working with multidimensional datasets, outliers might not be extreme in any single dimension but could still be unusual when considering all dimensions together. Statistical techniques often struggle with such cases.
4. *Robustness to noise*: Statistical methods can be sensitive to noise in the data. Noise might cause false positives, labelling valid data points as outliers, or false negatives, where true outliers go undetected.
5. *Dynamic datasets*: In datasets that evolve over time, such as streaming data, outliers might depend on temporal patterns. Statistical techniques designed for static data may not adapt well to these changes.

Machine learning methods have significantly advanced the detection of outliers, offering more robust and adaptive solutions compared to traditional statistical techniques. Here are some key advancements:

1. *Isolation forest*: This algorithm isolates anomalies by randomly selecting features and split values to create decision trees. Outliers, being rare and different, are isolated in fewer splits, making this method efficient and effective for high-dimensional data.
2. *Autoencoders*: These neural network-based models learn to compress and reconstruct data. Outliers are identified as data points with high reconstruction errors, as they deviate significantly from the learned patterns.
3. *Clustering techniques*: Methods like DBSCAN (Density-Based Spatial Clustering of Applications with Noise) identify outliers as points that do not belong to any cluster. These techniques are particularly useful for datasets with irregular distributions.
4. *One-Class SVM (Support Vector Machine)*: This method learns the boundary of normal data and flags points outside this boundary as outliers. It is effective for datasets with complex structures.
5. *Hybrid Approaches*: Combining machine learning with domain knowledge or statistical methods enhances the accuracy and interpretability of outlier detection. For example, integrating Isolation Forest with feature engineering tailored to the dataset.

These advancements have made outlier detection more scalable, adaptable, and applicable to diverse fields like fraud detection, network security, and healthcare.