

Outliers Detection and Removal

Abstract

Detecting outliers in data is an important problem due to the fact that outliers can cause models to make incorrect predictions or classifications, therefore identifying outliers and removing them are important tasks when dealing with data. In this paper we will propose two distance-based algorithms to detect outliers in a given dataset.

In our experiments we expected that after cleaning the outliers the model will be more accurate on all the datasets that we used. But not as we expected, the outlier removal did not help the machine learning model to improve the prediction. We also expected to find a decrease in the standard deviation. In this case, the removal did cause a decrease in the std as we wanted. Moreover, the process reduces the range in most of the numerical features, we expected that because of the same reasons as in the decreasing std scenario.

Problem Description

As mentioned, detecting outliers is an important task that has interesting applications ranging from data cleaning to financial fraud detection and from network intrusion detection to clinical diagnosis of diseases. In our research of the problem, we found that distance-based algorithms are not the common way of detecting algorithms at all and while reading about the subject we gathered that there is no distance-based algorithm that could even remotely replace the current methods, therefore we wanted to try and create a tool that helps data scientists understand the data better through distance-based algorithms.

Solution Overview

We decided to make two algorithms, the first depends (almost) only on the size of the dataset and not k (the number of neighbors to consider) nor n (the numbers of outliers to return) however it will not work efficiently on large datasets due to the fact that it computes the distances matrix for the dataset. The second algorithm uses a special data structure that fits this problem called a KD-Tree, using this data structure we managed to expand our tool for large datasets, but on the other hand it is depended on k , meaning the more neighbors we want to consider the more time the algorithm will take.

Algorithm 1

This algorithm computes the distance matrix for a given data set, then it sorts the matrix by value for each row and finally it sorts the matrix by the k^{th} column in descending order and return the top n indices in the matrix.

Algorithm 1: MatrixCleaningAlgorithm

1. *Compute a distance matrix from all vectors to all vectors in the dataset*
2. *Sort the distance matrix by value for each row*
3. *Sort the sorted distance matrix by the k^{th} column*
4. *Return the top n outliers indices in the distance matrix based on the distance to their k^{th} nearest neighbor*

The optimization in this algorithm is the use of the *Pandas* module which allows us to handle the data in an effective manner and the *Scipy* module which allows us to calculate the distance matrix very fast.

As we can see it does not matter which k and n will be given, the run time will be the same but if the dataset will be too large, we wouldn't be able to store the distance matrix in memory.

Algorithm 2

This algorithm creates a KD-Tree with the data and a specific metric, then it queries the tree, finding the distances of each vector to their k^{th} nearest neighbor and finally it returns the top n distance indices in the distances list.

Algorithm 2: TreeCleaningAlgorithm

1. *Creates a KD – Tree to store the dataset with a specific metric*
2. *Queries the tree – finds the distances of each vector to their k^{th} nearest neighbor*
3. *Returns the top n distance indices in the distances list*

The optimization in this algorithm is the use of the KD-Tree which is a binary search tree where data in each node is a K-Dimensional point in space.

The construction of the tree is as follows - divide the set of points into two groups by the first dimension where the middle is the vector that its value is the median value

for that dimension and it will be the root of the tree. In the next iteration divide the set by the second dimension and the medians will be the root's children and so on.

Once the tree is constructed, we can query it, meaning we can find the nearest neighbors to a vector, as follows – start from the root of the tree and check if the value of the first dimension is greater or less than it and choose accordingly until it reaches a leaf node. Once it reached a leaf node it needs to recursively go up the tree while noticing the distance between the vector and the dividing line to the section that was not visited. If the distance is less than the distance found to the nearest neighbor then that section of the tree will be traversed too.

As we can see the algorithm will work well on large data sets due to the use of the KD-Tree but if the k is too large the query will take time.

Related Work

There are a lot of methods for outlier detection such as:

- Z-Score
- Probabilistic and Statistical Modeling
- Linear Regression Models
- Proximity Based Models
- High Dimensional Outlier Detection Methods

However not many of them are distance-based.

While reviewing the articles (attached to the project's proposal) and more online we found that the widely accepted approach is in a matter of fact the naïve approach with different optimizations, therefore we concluded that we should also try and find an optimization to the naïve approach.

The most popular optimizations that are used in the articles are:

1. Pruning – using mostly ANNS (Approximate Nearest Neighbor Search) to detect if the current vector can eventually be an outlier and if it does not it would be pruned.
2. Ranking – to improve the pruning process. Ranking the vectors such that the ANNS rule will be triggered earlier in the computation.

As for us, we tried to use a data structure that will allow the algorithm to skip on as many vectors as possible and in that way, we will achieve the same result in hopefully less time.

Experimental evaluation

In purpose to prove that our process detects and cleans outliers, we used four datasets with different sizes and different numbers of features. For each dataset, we split the data into a train set and a test set at a ratio of 75-25 percent. We train a linear regression model on the train set and computed the accuracy on the test set. After that, we ran the outlier cleaning process on the train set, and train the model again. In addition, we computed the R square rating. We also compared each dataset with its cleaned dataset by pandas describe table, to look at the std (standard deviation) and the min-max values, and print boxplots for some features.

By checking the std – if the algorithm works as we expect, the std should be lower on the cleaned datasets than the std on the original datasets. Moreover, we expect that in most cases the range between the minimum and maximum values would be lower in the cleaned dataset.

This inspection is proper since the outliers (when we look at numerical values), will have a feature with high or low values that does not match the variability of the data. So, an outlier, as we describe, will increase the std. In addition, a very high or low value could harm the learning of the model, therefore, looking at the minimum and maximum values of a feature is a measure of the algorithm's success.

In all cases of numerical features, as we expected, there was a decrease in the std and a reduction in the min-max range. In cases where the feature is Boolean, we didn't get a significant decrease in the std. Also, after averaging the results of a hundred iterations on each data set, we saw that the accuracy and the R square rating were the same (with two percent deviation). In conclusion, the cleaning of the four datasets didn't influence the learning model. But there is an effect on the statistical power. We assume that maybe the datasets have a similar process as our cleaning data process before they were uploaded online.

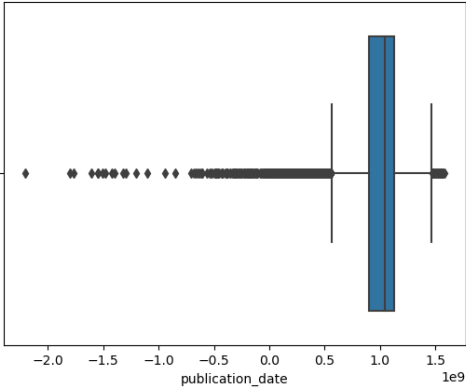
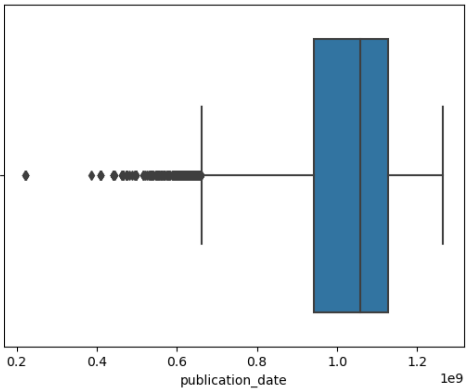
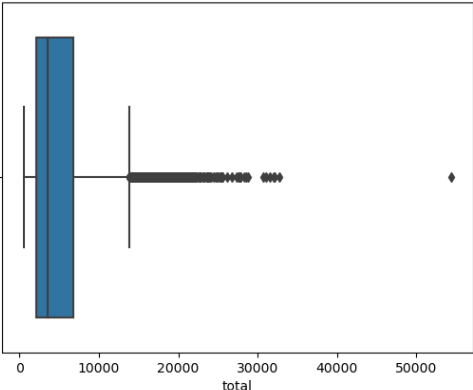
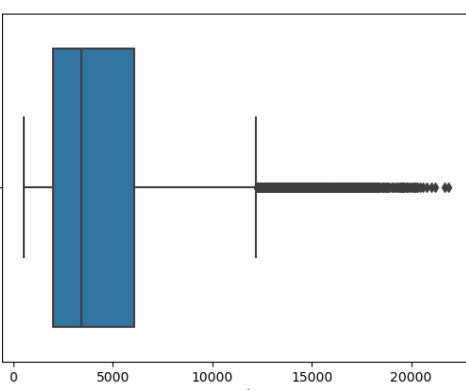
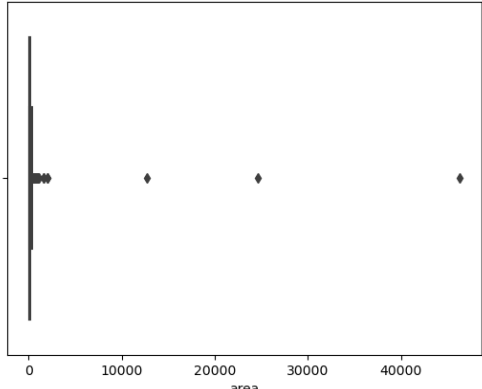
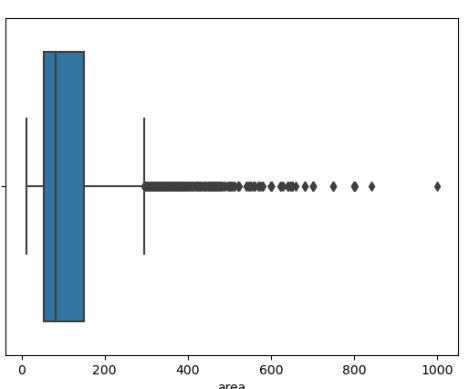
To visually see the quality of the process we attached some boxplots diagrams and the pandas describe table. We conclude that the dropping in std and value range is caused by our data cleaning process. More than that we can infer that the algorithm works as we expect.

```
houses
Average accuracy: 86.69622147399927
Average clean accuracy: 84.33557800224469
R Square: 0.9999991765061828
R Square clean: 0.9999991376847296
```

```
books
Average accuracy: 91.65168943206325
Average clean accuracy: 91.65168943206325
R Square: -0.030209538777861145
R Square clean: -0.030209538777861145
```

```
flights
Average accuracy: 93.20589786205839
Average clean accuracy: 93.20589786205839
R Square: 0.41693231787974105
R Square clean: 0.41693231787974105
```

```
covid
Average accuracy: 100.0
Average clean accuracy: 100.0
R Square: 1.0
R Square clean: 1.0
```

Dataset Name	Original	Cleaned
Books		
Houses		
Houses		

Conclusion

In this project we proposed a new way to detect outliers in datasets using special data structures in distance-based algorithms. We used various datasets which we cleaned from outliers using our new tool and then compared the cleaned datasets with the originals and found that although the models were not improved as much as we liked the statistic results did improve, meaning our approach did have an impact over the datasets even if it was not as much as we expected.