Project Report for Outlier Detection on Credit Card Fraud and Boston Datasets

Sharon Saronian

In this project, the goal is to conduct Outlier Detection on certain datasets. There are different types of Outlier Detection methods. In this project we will perform statistical, supervised, and unsupervised Outlier Detection. Statistical methods will include outlier detection using non-parametric visualization techniques such as boxplots and scatter plots, and multivariate parametric methods such as X^2-test (Z-Score or F1-Score), and IQR score on the Boston Dataset. Supervised methods will include the Isolation Forest algorithm, and unsupervised methods will include the Local Outlier Factor (LOF) and the One Class SVM algorithms on the Credit Card Fraud Detection Dataset. For each dataset, outputs from the mentioned algorithms will be compared to each other.

1. Dataset Information

1.1. Boston Housing Dataset

The Boston dataset contains information that was gathered by the U.S Census Service. It concerns housing in the area of Boston. It is originally obtained from the StatLib archive (http://lib.stat.cmu.edu/datasets/boston).

This dataset is widely used in classification tasks in machine learning, but is also a very good case to study outlier detection. The reason this dataset was chosen is that it isn't imbalanced, and many of its attributes follow the Gaussian distribution, which makes it a suitable case for statistical outlier detection algorithms.

There are 13 attributes, all numerical, as stated below:

1. CRIM - per capita crime rate by town
2. ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS - proportion of non-retail business acres per town.

4. CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
5. NOX - nitric oxides concentration (parts per 10 million)
6. RM - average number of rooms per dwelling
7. AGE - proportion of owner-occupied units built prior to 1940
8. DIS - weighted distances to five Boston employment centres
9. RAD - index of accessibility to radial highways
10. TAX - full-value property-tax rate per $10,000
11. PTRATIO - pupil-teacher ratio by town
12. B - 1000(Bk - 0.63) ^2 where Bk is the proportion of blacks by town
13. LSTAT - % lower status of the population

And there is one class attribute, which is the Median value of owner-occupied homes in $1000's and is numerical and continuous.

There are 506 instances, so this is a rather small dataset.

The Boston dataset is available online, but python's Sklearn package also has it built-in. Therefore, we will just load it from there and use it.

1.2. Credit Card Fraud Detection Dataset

This dataset is downloaded from Kaggle (https://www.kaggle.com/mlg-ulb/creditcardfraud). It has been collected during a research on fraud detection and big data by the Machine Learning and Worldline group of ULB university.

The dataset contains credit card transactions made in September 2013 by European cardholders. The transactions occurred in two days, and there are 284,807 transactions (instances) in all. This makes it a rather huge dataset, with many instances. **That's why the size of my compressed project folder would be more than 50 MBs, and LMS wouldn't let me upload the folder. So, I had to not include it in the project folder. But it can be downloaded from the link I provided above.**

There are 28 numerical attributes, namely V1, V2, ... V28. These are not original attributes, but are results of a PCA transformation. The original features and more information about them weren't provided by the owners due to confidentiality problems. So, the transformed features were provided instead. There are also two untransformed features, namely 'Time' and 'Amount'. 'Time' contains the seconds elapsed between the first transaction in the dataset and each transaction. 'Amount' is simply the transaction amount. So, there are 30 features in total.

There is also one Class label feature, which is binary. It has two values, 0 and 1. 0 means that the transaction wasn't a fraud and 1 means that it was.

This dataset is highly imbalanced, with only 0.172% of the labels being fraud. Therefore, typical statistical outlier detection methods can't be used on it.

Instead, we will use supervised and unsupervised algorithms as stated above, to detect the outliers in this dataset.

2. Outlier Detection

2.1. Statistical Outlier Detection

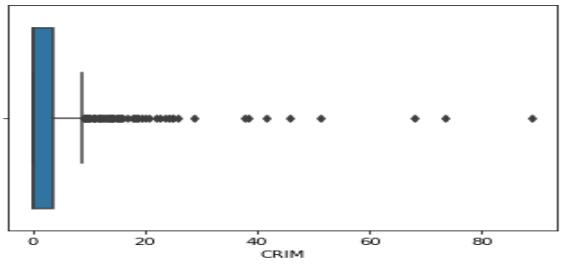
In this step we will do statistical outlier detection on the Boston dataset. First, we load the Boston dataset, and separate it's features from the class labels. Here are the features (5 first instances):

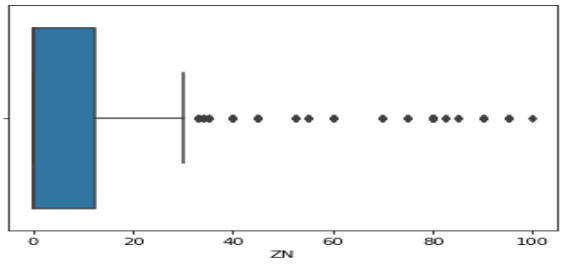
	CRIM	ZN	INDUS	CHAS	NOX	...	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	...	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	...	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	...	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	...	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	...	3.0	222.0	18.7	396.90	5.33

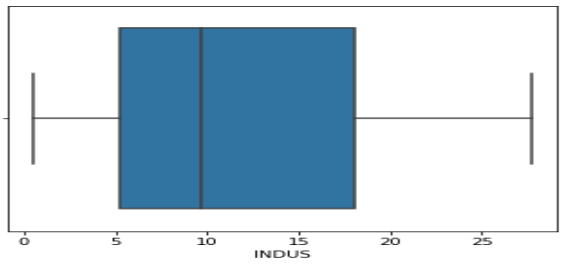
And here are the class labels (house prices-again 5 first instances):

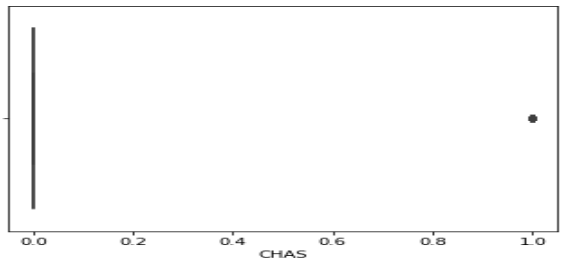
	0
0	24.0
1	21.6
2	34.7
3	33.4
4	36.2

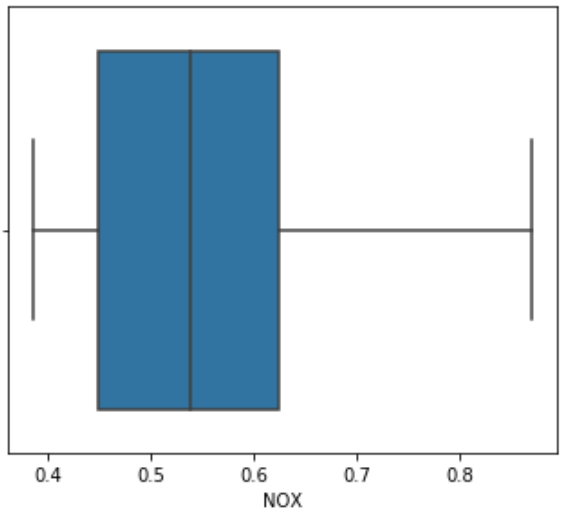
The first thing we do is univariate outlier analysis with boxplots. For each of the 13 features, we will plot the boxplot. We know that the points lying outside the boxplot indicate outliers, while those that are normal points are all grouped together and visualized by boxes, so that’s a good start. Here are the boxplots for each of the features:

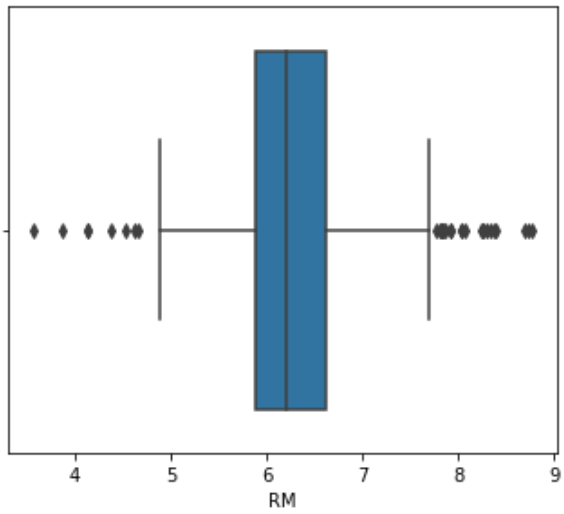

0
20
40
60
80
CRIM

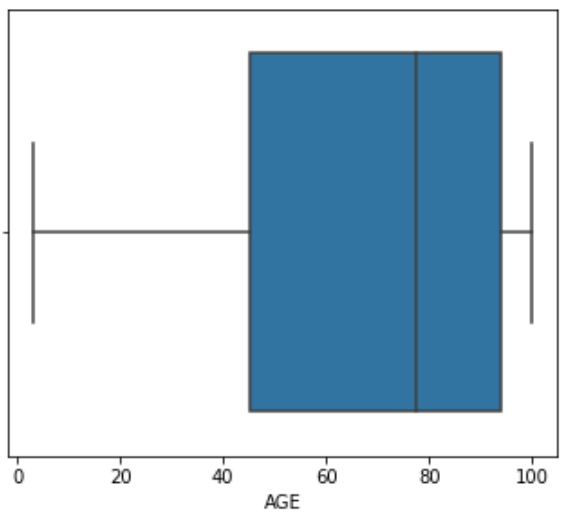

0
20
40
60
80
100
ZN

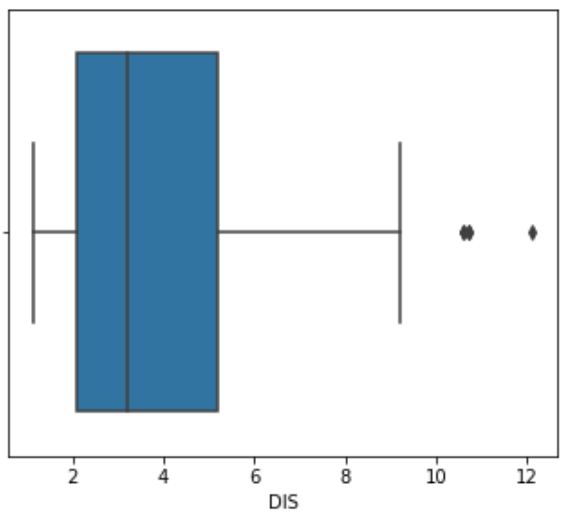

0
5
10
15
20
25
INDUS

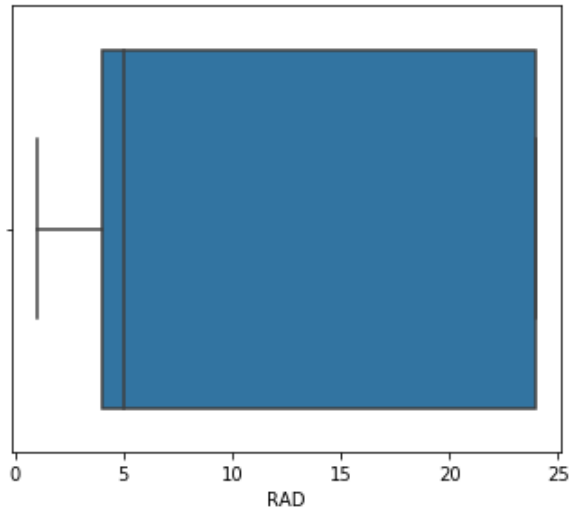

0.0
0.2
0.4
0.6
0.8
1.0
CHAS

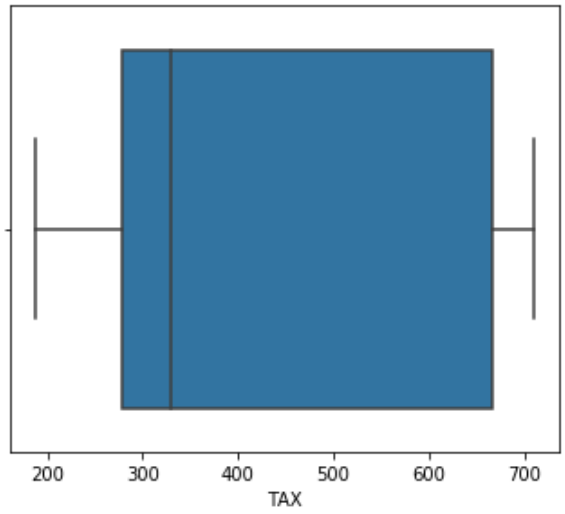

0.4
0.5
0.6
0.7
0.8
NOX

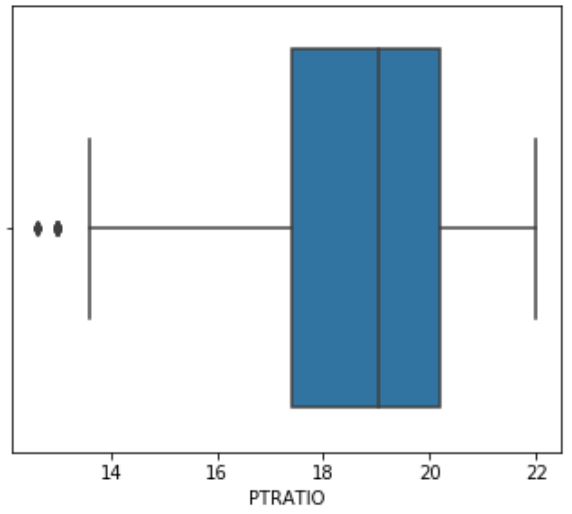

4
5
6
7
8
9
RM

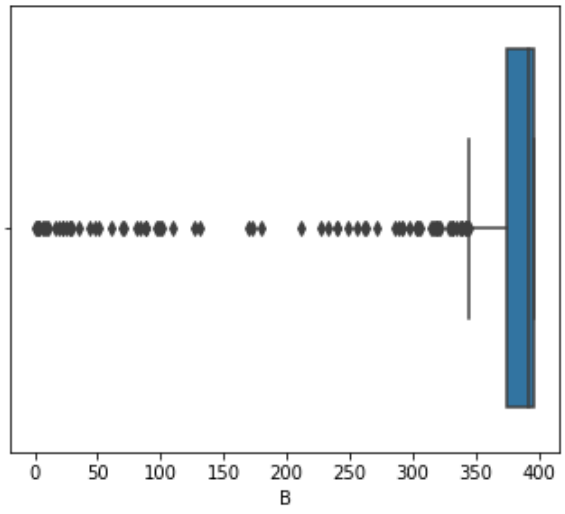

0
20
40
60
80
100
AGE

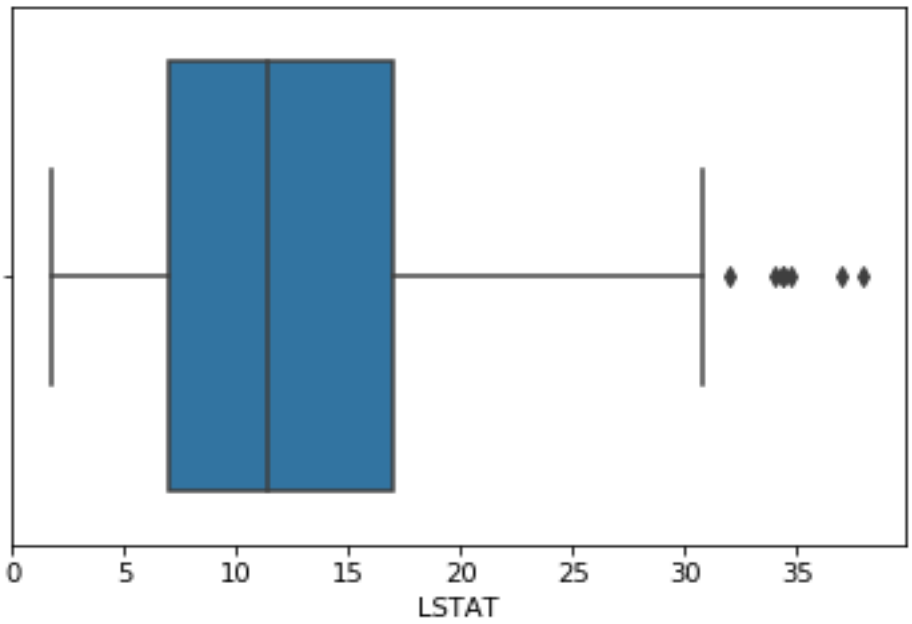

2
4
6
8
10
12
DIS


0
5
10
15
20
25
RAD


200
300
400
500
600
700
TAX

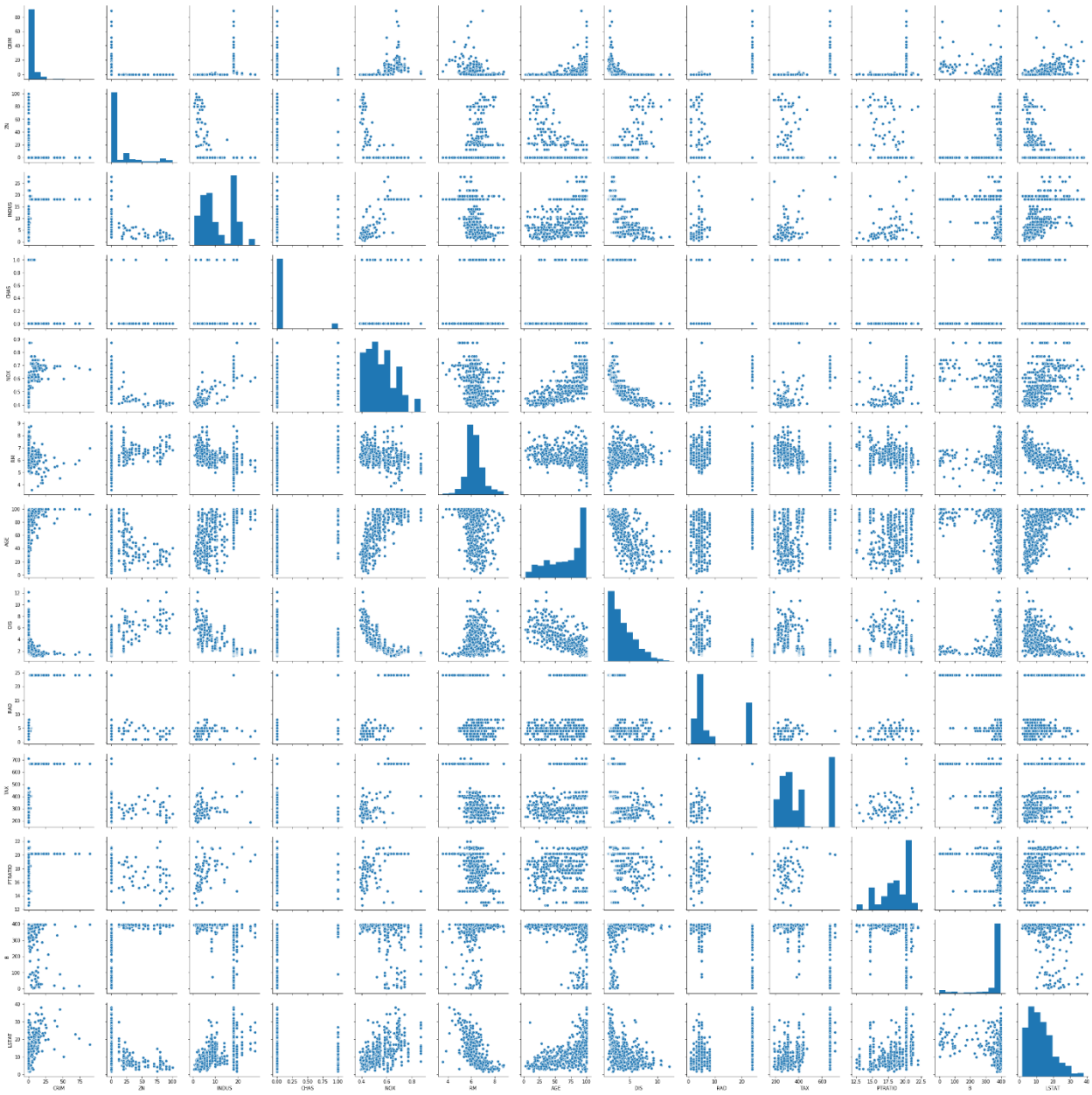

14
16
18
20
22
PTRATIO


0
50
100
150
200
250
300
350
400
B


0
5
10
15
20
25
30
35
LSTAT

As we see above, the black dots are outliers, which lye outside the box. Some features don’t have outliers, like RAD and TAX, some have moderate amounts, like LSTAT, while others have a lot, like CRIM and ZN.

The next step is to try multivariate outlier analysis, meaning that more than one feature at a time can be analyzed. Unfortunately, boxplots can only be used in multivariate attributes if they are categorial, which is not our case here. Instead, we will use scatter plots. Regions with a high density of points are normal, while those scattered farther away from others are outliers:



This matrix contains scatter plots for every pair of features. As mentioned above, if we look at every plot, we can detect lone points. Those are the outliers.

By far we have just visualized the outliers. But we don't actually know which points they are, numerically. To address this issue, we will use two statistical functions, the Z-score and the IQR score.

Z-score describes every point by calculating its relationship with the mean and standard deviation of the entire data points. It computes a normal

distribution of the data. By calculating the Z-score we reshape the data and center them, so points that are far from zero indicate a potential outlier.

These are the calculated Z-scores for every data point:

```
[[0.41978194 0.28482986 1.2879095  ... 1.45900038 0.44105193 1.0755623 ]
 [0.41733926 0.48772236 0.59338101 ... 0.30309415 0.44105193 0.49243937]
 [0.41734159 0.48772236 0.59338101 ... 0.30309415 0.39642699 1.2087274 ]
 ...
 [0.41344658 0.48772236 0.11573841 ... 1.17646583 0.44105193 0.98304761]
 [0.40776407 0.48772236 0.11573841 ... 1.17646583 0.4032249  0.86530163]
 [0.41500016 0.48772236 0.11573841 ... 1.17646583 0.44105193 0.66905833]]
```

It’s hard to understand anything from the results. So, we will define a threshold of 3. Since the normal rescaled values are normally between 0 and 3, any point bigger that 3 will indicate an outlier:

```
(array([ 55,  56,  57, 102, 141, 142, 152, 154, 155, 160, 162, 163, 199,
       200, 201, 202, 203, 204, 208, 209, 210, 211, 212, 216, 218, 219,
       220, 221, 222, 225, 234, 236, 256, 257, 262, 269, 273, 274, 276,
       277, 282, 283, 283, 284, 347, 351, 352, 353, 353, 354, 355, 356,
       357, 358, 363, 364, 364, 365, 367, 369, 370, 372, 373, 374, 374,
       380, 398, 404, 405, 406, 410, 410, 411, 412, 412, 414, 414, 415,
       416, 418, 418, 419, 423, 424, 425, 426, 427, 427, 429, 431, 436,
       437, 438, 445, 450, 454, 455, 456, 457, 466], dtype=int64), array([ 1,  1,  1,
11, 12,  3,  3,  3,  3,  3,  3,  3,  1,  1,  1,  1,  1,
        1,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  5,  3,  3,  1,  5,
        5,  3,  3,  3,  3,  3,  3,  1,  3,  1,  1,  7,  7,  1,  7,  7,  7,
        3,  3,  3,  3,  3,  5,  5,  5,  3,  3,  3, 12,  5, 12,  0,  0,  0,
        0,  5,  0, 11, 11, 11, 12,  0, 12, 11, 11,  0, 11, 11, 11, 11, 11,
       11,  0, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11],
      dtype=int64))
```

The result array has two inner arrays. The first part is an array consisting of rows, and the second array includes columns of the outliers. That way we know the locations of the outliers (or by our definitions points with Z-score values of bigger than 3) and we can access them by simply printing them. For example, the first number in the upper array is 55, and the one in the second array is 1. That means the point located in row 55 and column 1 is an outlier. We can simply print its value:

```
3.375038763517309
```

The second score to use is the IQR score. We know that boxplots actually use this score to detect and plot the outliers. But we want to access the outliers. That’s why we will compute the IQR manually and do outlier

analysis based on that. We know that IQR has a simple formula, it is the difference between the third and first Quartiles of the data.

Thus, let's compute the IQR for each attribute:

```
CRIM         3.595038
ZN          12.500000
INDUS       12.910000
CHAS         0.000000
NOX          0.175000
RM           0.738000
AGE         49.050000
DIS          3.088250
RAD         20.000000
TAX        387.000000
PTRATIO      2.800000
B           20.847500
LSTAT       10.005000
dtype: float64
```

Now we want to detect the outliers. According to a simple formula, we know that any data point which doesn't lye in the range of 1.5 x IQR (meaning the value is less or more than that), is an outlier. So we will compute the range, and we will show normal points that lye in that range with 'False', and we will show outliers with 'True':

	CRIM	ZN	INDUS	CHAS	NOX	...	RAD	TAX	PTRATIO	B	LSTAT
0	False	False	False	False	False	...	False	False	False	False	False
1	False	False	False	False	False	...	False	False	False	False	False
2	False	False	False	False	False	...	False	False	False	False	False
3	False	False	False	False	False	...	False	False	False	False	False
4	False	False	False	False	False	...	False	False	False	False	False
5	False	False	False	False	False	...	False	False	False	False	False
6	False	False	False	False	False	...	False	False	False	False	False
7	False	False	False	False	False	...	False	False	False	False	False
8	False	False	False	False	False	...	False	False	False	False	False
9	False	False	False	False	False	...	False	False	False	False	False
10	False	False	False	False	False	...	False	False	False	False	False
11	False	False	False	False	False	...	False	False	False	False	False
12	False	False	False	False	False	...	False	False	False	False	False
13	False	False	False	False	False	...	False	False	False	False	False
14	False	False	False	False	False	...	False	False	False	False	False
15	False	False	False	False	False	...	False	False	False	False	False
16	False	False	False	False	False	...	False	False	False	False	False
17	False	False	False	False	False	...	False	False	False	False	False
18	False	False	False	False	False	...	False	False	False	True	False

The first 18 instances are shown. According to the picture, the point in row 18 and column 12 (B) is an outlier.

For the last step, a comparison is done between the Z-score method and the IQR score method. We will calculate the initial shape of the data, and

for each score we will remove the corresponding outliers detected by each method. The reason we do this is to count the number of outliers detected by each method:

```
(506, 13)
(415, 13)
(274, 13)
```

This pictures sums everything up. The initial number of points were 506. With the Z-score method, 415 of them are left. That means that Z-score detected 91 instances as outliers. IQR did even more. The number of instances left with IQR are 274, meaning that 232 were detected as outliers.

If we want to compare the two methods, we could say that IQR detects more points as outliers. Whether or not we should correct them, leave them as they are, or remove them depends on the real-world project we deal with. If we use the IQR method, then removing the outliers doesn't seem to be a logical move as this is already a small dataset and removing almost half of the points leaves us with very little to work on. So, for this project, Z-score seems to be more efficient.

2.2. Supervised and Unsupervised Outlier Detection

For the next part we will do outlier analysis on the Credit Card Fraud dataset. As mentioned above, this dataset is highly imbalanced, and stronger tools are necessary for outlier detection:

```
0    284315
1       492
Name: Class, dtype: int64
```

There are only 492 fraud cases available, compared to 284,315 normal cases.

The dataset is fairly huge, and the amount of computations caused very slow run time. That's why the codes are executed in Google Colab and then a python file was generated from it, included in the projects folder. The original Colab file with executed outputs is also included in the projects folder in the form of a python notebooks.

Let’s view the 5 first instances of the dataset, along with its shape:

	Time	V1	V2	V3	...	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	...	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	...	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	...	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	...	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	...	0.219422	0.215153	69.99	0

[5 rows x 31 columns]
(284807, 31)

Below is a statistical description of the dataset:

	Time	V1	...	Amount	Class
count	284807.000000	2.848070e+05	...	284807.000000	284807.000000
mean	94813.859575	3.919560e-15	...	88.349619	0.001727
std	47488.145955	1.958696e+00	...	250.120109	0.041527
min	0.000000	-5.640751e+01	...	0.000000	0.000000
25%	54201.500000	-9.203734e-01	...	5.600000	0.000000
50%	84692.000000	1.810880e-02	...	22.000000	0.000000
75%	139320.500000	1.315642e+00	...	77.165000	0.000000
max	172792.000000	2.454930e+00	...	25691.160000	1.000000

Which confirms an imbalanced distributed dataset.

We should also check if there any missing values, so they won’t be mistaken with outliers:

```
False
```

Now its time for outlier analysis. Before we go any further, I should point that even with conducting the experiments in the environment of Colab, which allows for faster computations, the run time was extremely slow. Therefore, a sample from the data will be taken for modeling and outlier detection:

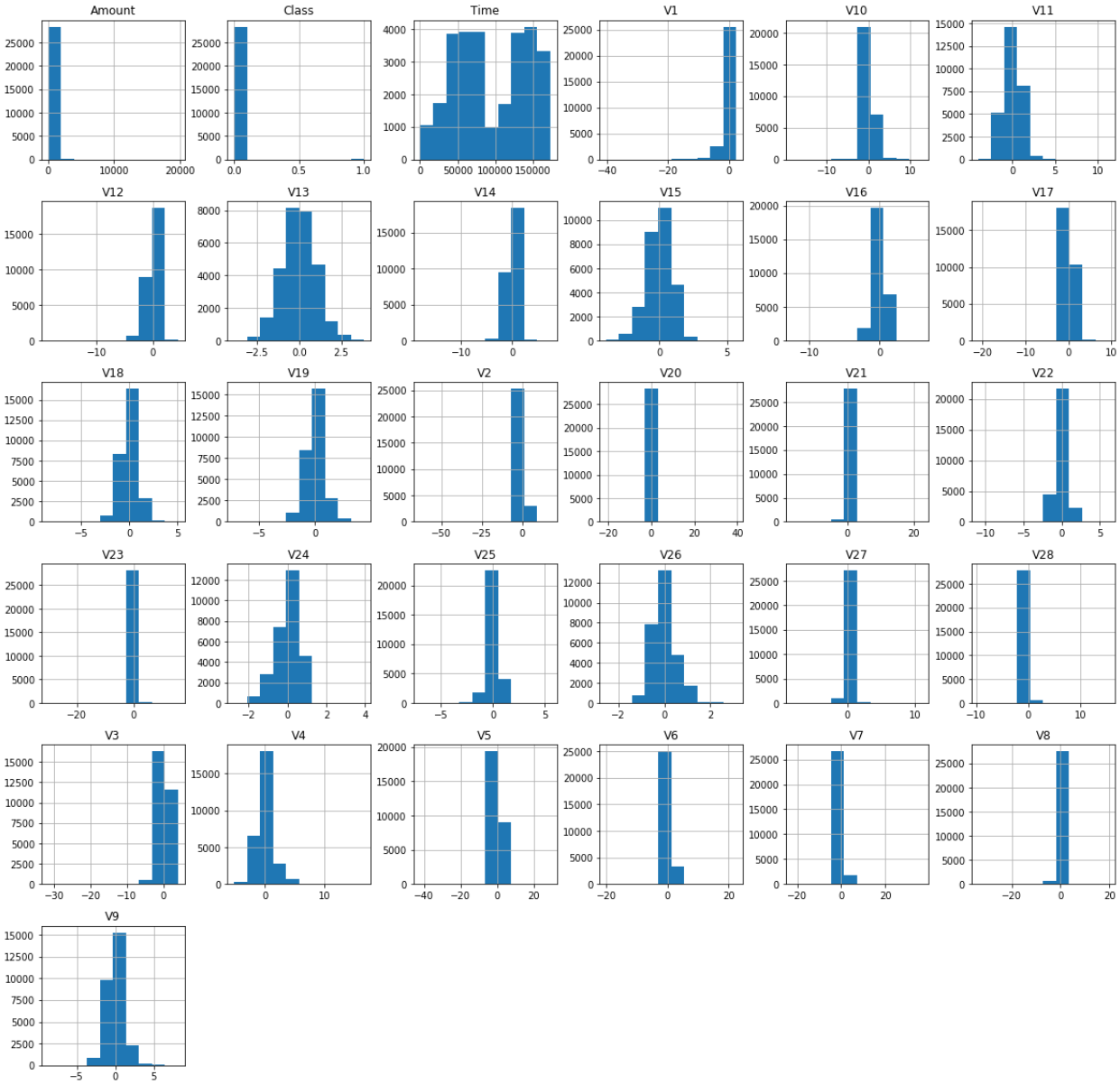
```
(28481, 31)
```

This new sample has 28,481 samples, which allows us to do faster computations.

This sample is still very imbalanced, with only 49 fraud cases compared to 28,432 normal ones:

```
Number of None-Fraud Transactions 28432
Number of Fraud Transactions 49
```

Let’s see histogram plots for each attribute, which will visualize the outliers for us:



We can see that some bins include very little samples in each plot, so those might suggest outliers. Now it’s time to run some algorithms.

For supervised outlier detection, we will run the Isolation Forest algorithm on the data. The Isolation Forest is a rather new algorithm for outlier detection, which is claimed to have a better performance compared to existing methods. We will see if that’s true.

This algorithm is based on the idea that outliers are isolated points, which are not many in numbers and are rather different. The concept of isolation is therefore used instead of densities, or distances. The algorithm randomly selects a feature, and then randomly selects a split value between the min and max values of that particular feature. The idea is that isolating outliers requires less conditions than normal cases, so, the number of conditions required to separate a point can be introduced as a measure of anomaly. The algorithm constructs random decision trees and then, the score is reported as the length of the path to isolate an observation.

This is the result obtained by the Isolation Forest algorithm:

```
Number of Outliers Detected by Isolation Forest: 77
```

Now it's time to conduct unsupervised methods. The first one is LOF. LOF computes the local density deviation of each sample with respect to its neighbors. A point having a lower density than its neighbors is reported as an outlier. This is the result obtained by LOF:

```
Number of Outliers Detected by Local Outlier Factor: 97
```

The second unsupervised method is One Class SVM. This algorithm is only trained on normal points, meaning those belonging to the none-fraud cases. It learns boundaries for the normal points and so any points lying outside those boundaries are detected as outliers.

This is the output by One Class SVM:

```
Number of Outliers Detected by Local Outlier Factor: 8516
```

For the last step let's do a comparison between these three algorithms. It seems that Isolation Forest is indeed better than existing methods, with 77 samples detected as outliers. Although this number is more than the actual number of 49, but have in mind that detecting some types of outliers are truly difficult, and there

will always be a possibility for some mistakes. But also recall that these algorithms were performed on a sample of the actual dataset, so the actual results might be even better with fewer errors.

LOF was worse than Isolation Forest, with 97 detected outliers, which is quite different than the real number. The worst one is One Class SVM, with 8516 detected cases, a number which is far away than reality.

In Conclusion, in terms of performance, Isolation Forest is the ideal outlier detection algorithm, specially if the dataset is highly imbalanced such as this one.