Data Mining and Machine Learning

Anomaly Detection

Dr. Edgar Acuna
Department of Matematics

Universidad de Puerto Rico- Mayaguez

academic.uprm.edu/eacuna

Outlier detection: The goal is to detect unusual observations in the training dataset. Outlier detection estimators thus try to fit the regions where the training data is the most concentrated, ignoring the deviant observations.

Novelty Detection: The training data is not contaminated by outliers and we are interested in detecting whether a **new** observation is an outlier. In this context an outlier is also called a novelty.

The data mining community got interested in outliers after Knorr and Ng (1998) proposed a non-parametric approach to outlier detection based on the distance of a instance to its nearest neighbors.

An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism (Hawkins, 1980).

A comprehensive treatment of outliers in the field of statistics appears in Barnet and Lewis (1994). They provide a large list of outlier detection methods.These methods have two main drawbacks:

First, almost all of them are for univariate data.

Second, all of them are distribution-based. Real-world data are commonly multivariate with unknown distribution.

Anomaly detection has many applications among them:

- Data cleaning
- Intrusion detection
- Fraud detection
- Systems health monitoring
- Event detection in sensor networks
- Ecosystem disturbances (Earthquakes)

Frequently, outliers are removed to improve accuracy of the estimators. However, this practice is not recommendable because sometimes outliers can have very useful information.

Univariate outliers

Let $\bar{x}$ be the mean and let *s* be standard deviation of the data distribution. One observation is declared as an outlier if lies outside of the interval

$$(\bar{x} - ks, \bar{x} + ks) \quad (1)$$

where the value of k is usually taken as 2 or 3. The justification of these values relies on the fact that assuming normal distribution one expects to have a 95.45% (99.75%, respectively) percent of the data on the interval centered in the mean with a semi-length equal to two (three, respectively) standard deviation.

From equation (1), the observation x is considered an outlier if

$$\frac{|\, x - \bar{x} \,|}{s} > k \quad (2)$$

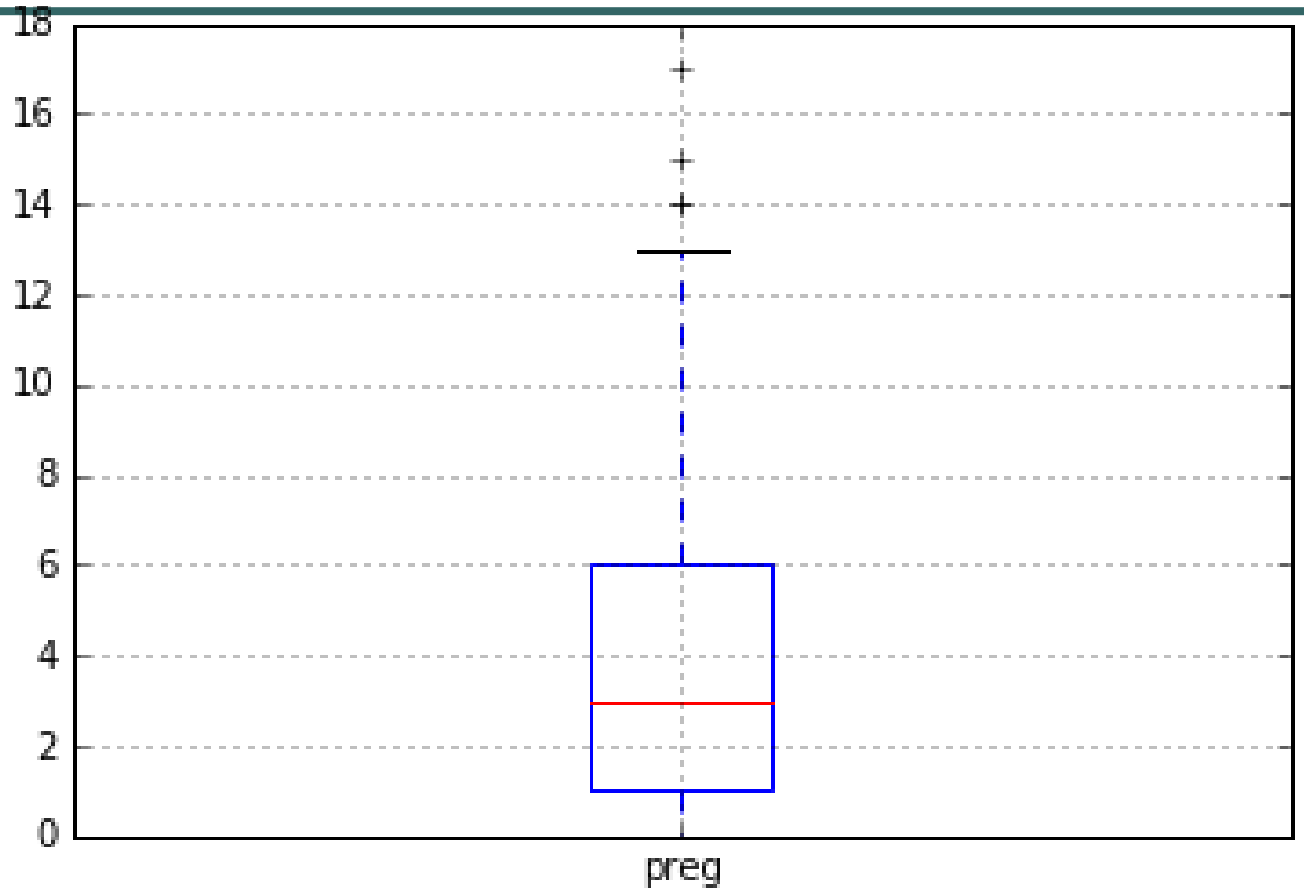
The problem with the above criteria is that it assumes normal distribution of the data something that frequently does not occur. Furthermore, the mean and standard deviation are highly sensitive to outliers.

The ***Boxplot*** (Tukey, 1977) is a graphical display for exploratory data analysis, where the outliers appear tagged. Two types of outliers are distinguished: ***mild outliers*** and ***extreme outliers***.

An observation x is declared an ***extreme outlier*** if it lies outside of the interval (Q_1-3×IQR, Q_3+3×IQR),where IQR=Q_3-Q_1 is called the ***Interquartile Range***. An observation x is declared *a* ***mild outlier*** if it lies outside of the interval (Q_1-1.5×IQR, Q_3+1.5×IQR).

The numbers 1.5 and 3 are chosen by comparison with a normal distribution.

Univariate outliers using boxplots

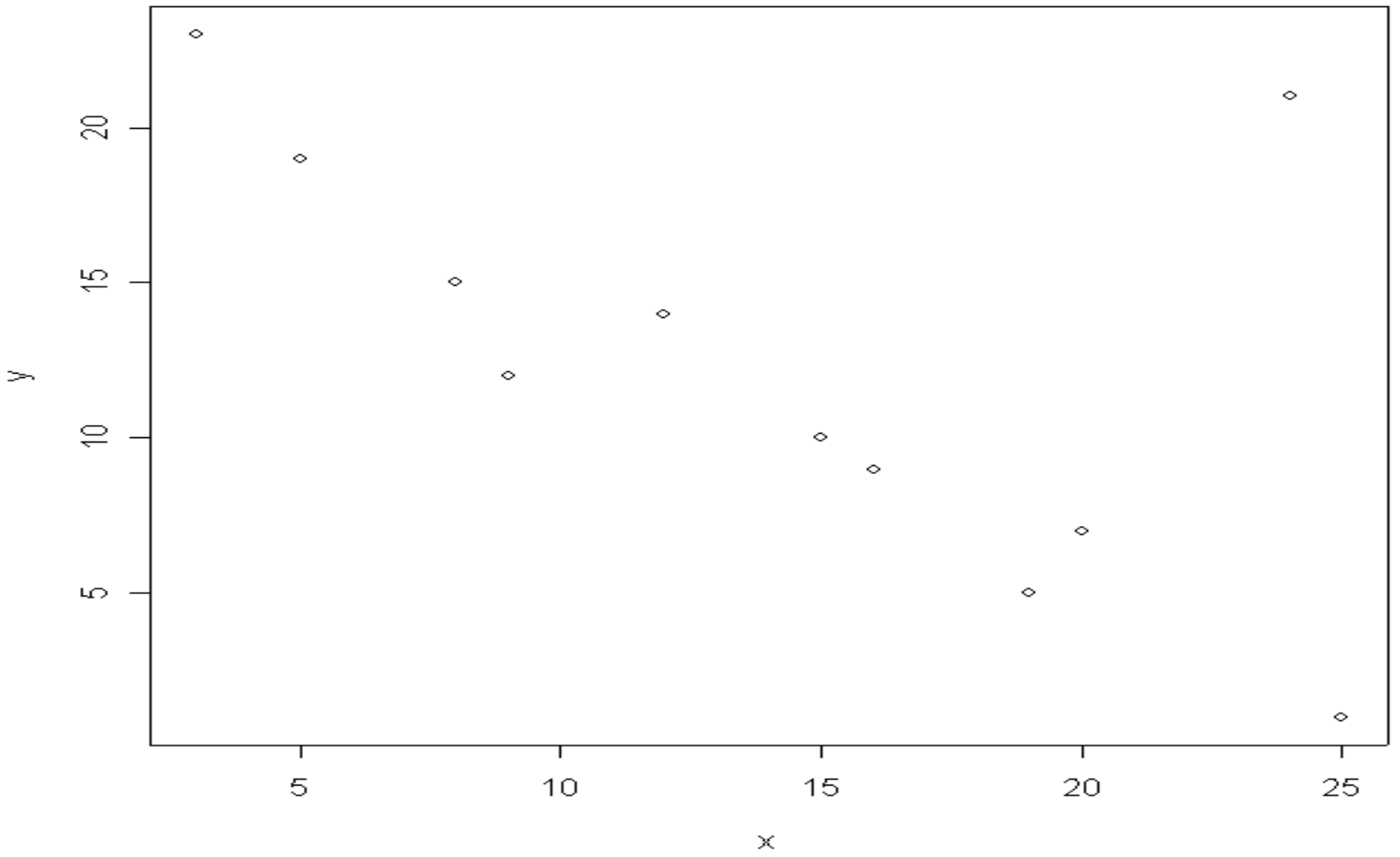


Multivariate Outliers

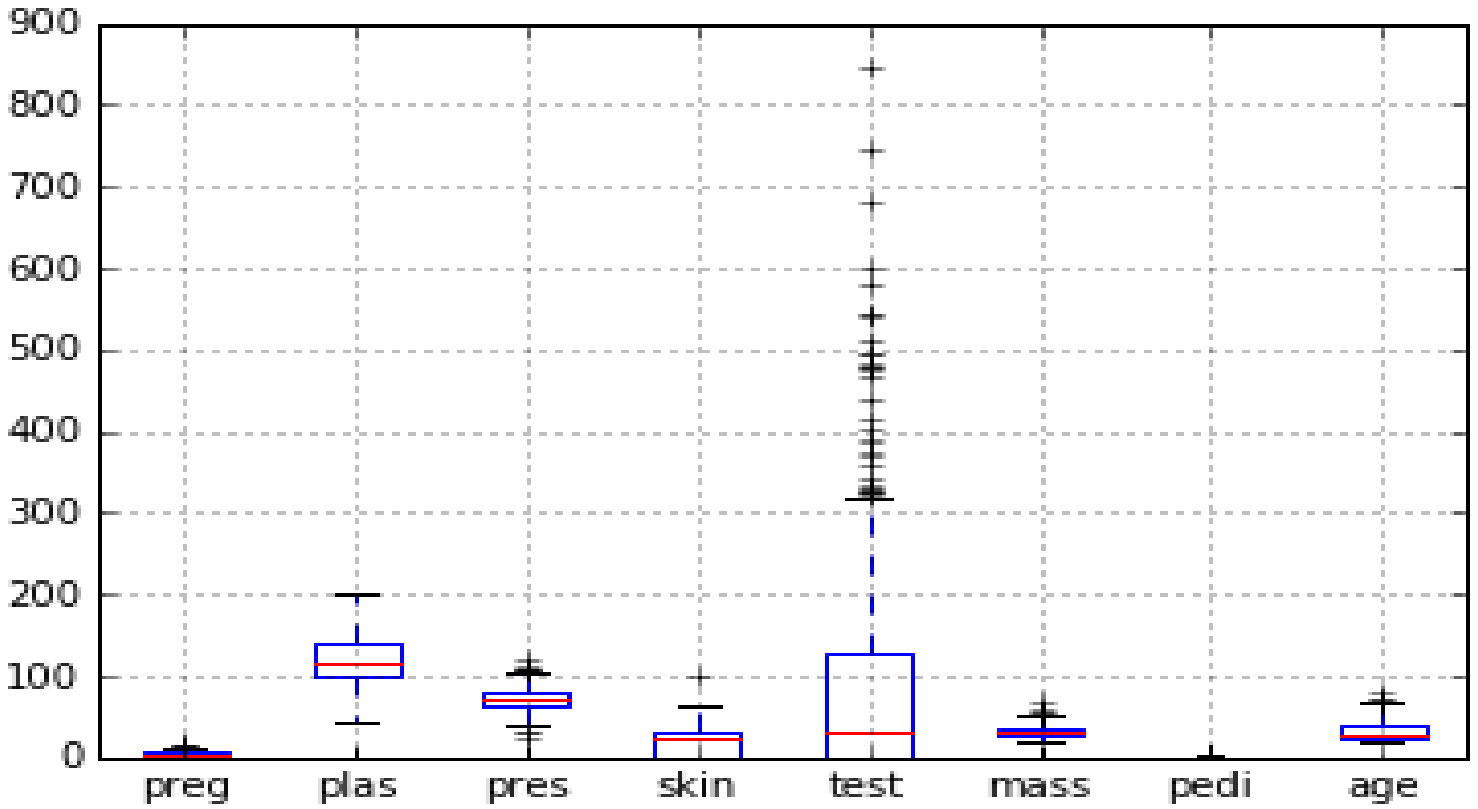
Let us consider a dataset D with p features and n instances. In a supervised classification context, we must also know the classes to which each of the instances belongs.

The objective is to detect all the instances that seem to be unusual, these will be the multivariate outliers.

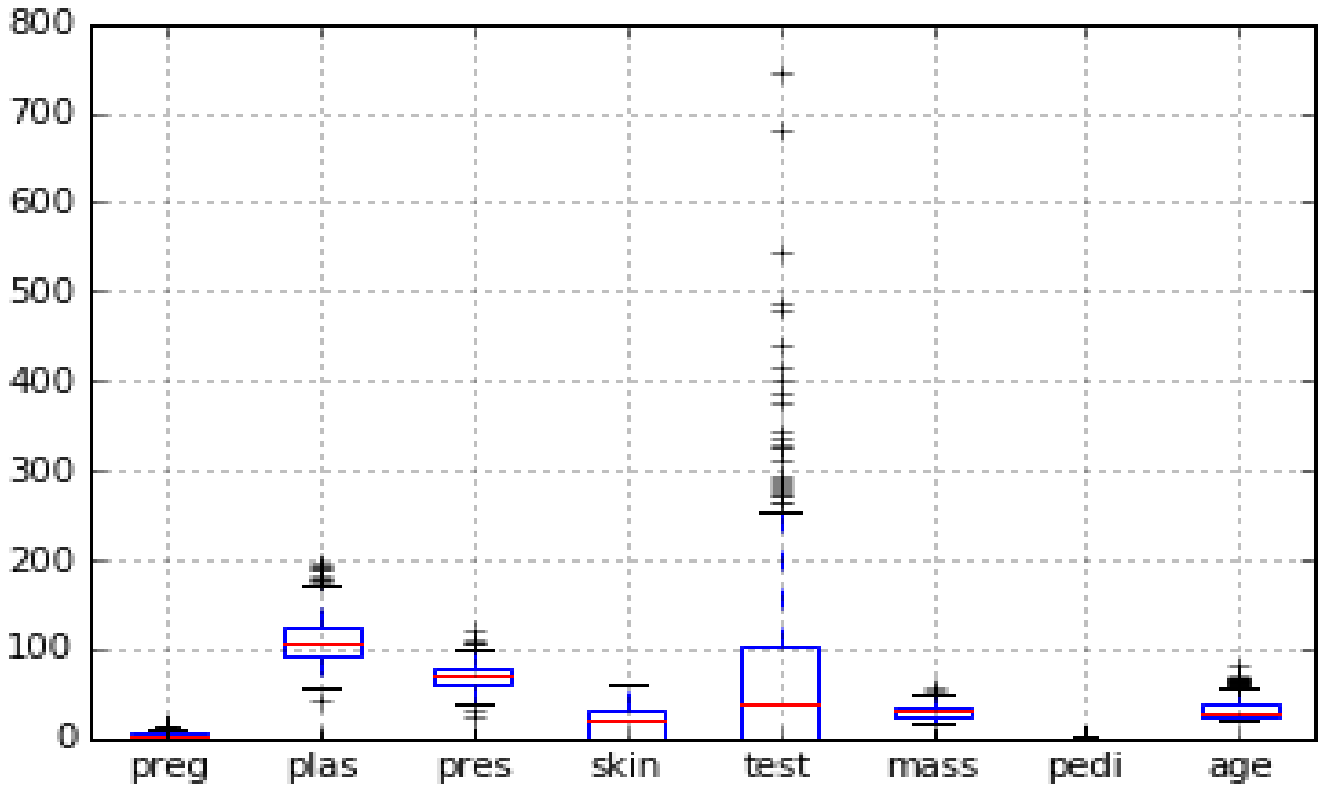
One might think that multivariate outliers can be detected based on the univariate outliers in each feature, but this is not true. On the other hand, an instance can have values that are outliers in several features but the whole instance might not be a multivariate outlier.



A bi-dimensional outlier that is not an outlier in either of its projections.



Detecting outliers for the eight features in the *Diabetes* data set



Outliers in the eight features of the first class of Diabetes dataset

Methods to detect multivariate outliers

- Robust statistical-based outlier detection,
- Outlier detection by clustering,
- Distance-based outlier detection,
- Density-based local outlier detection, and
- Using Neural networks: Autoencoders

Robust Statistical based outlier detection

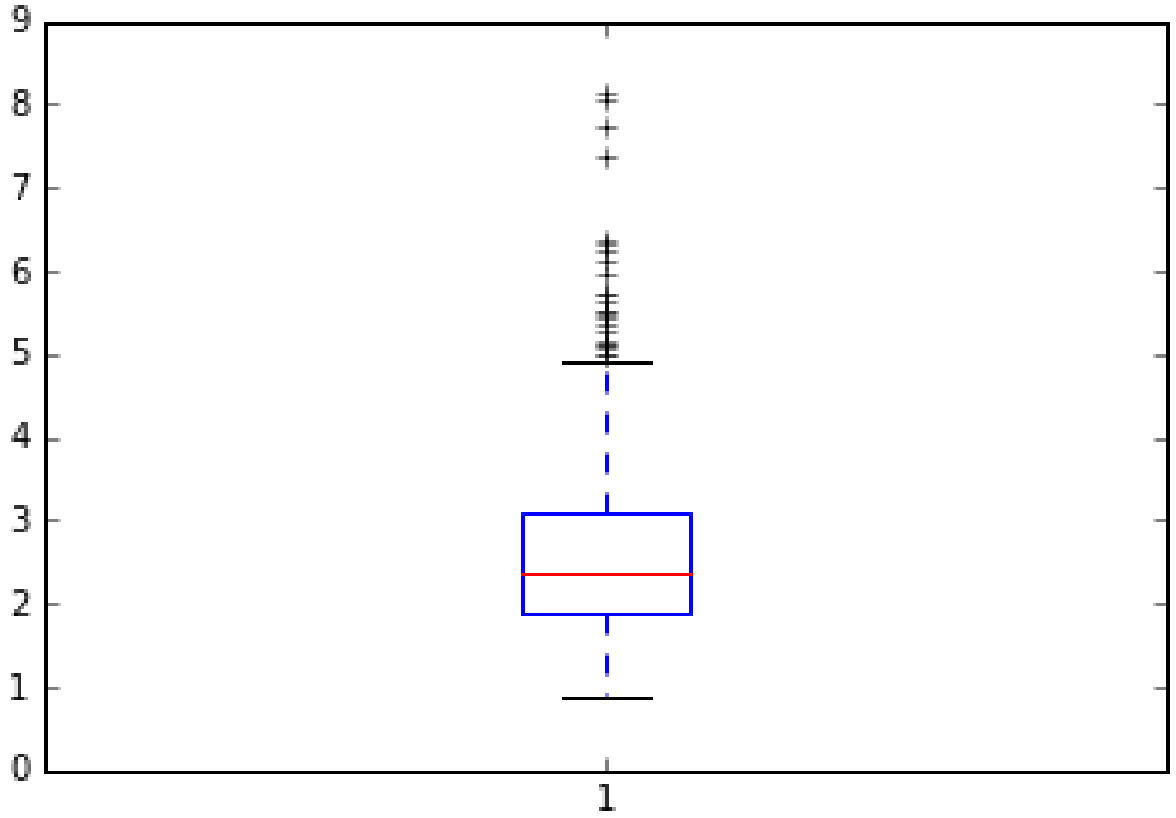
Let **x** be an observation of a multivariate data set consisting of n observations and p features. Let $\bar{\mathbf{x}}$ be the centroid of the dataset, which is a p-dimensional vector with the means of each feature as components. Let **X** be the matrix of the original dataset with columns centered by their means. Then, the p×p matrix **S=**1/(n-1) **X'X** represents the covariance matrix of the p features. The multivariate version of equation (2) is

$$D^2(\mathbf{x},\bar{\mathbf{x}}) = (\mathbf{x}-\bar{\mathbf{x}})'\mathbf{S}^{-1}(\mathbf{x}-\bar{\mathbf{x}}) > k$$

where D^2 is called the Mahalanobis square distance from **x** to the centroid of the dataset. An observation with a large Mahalanobis distance can be considered as an outlier.

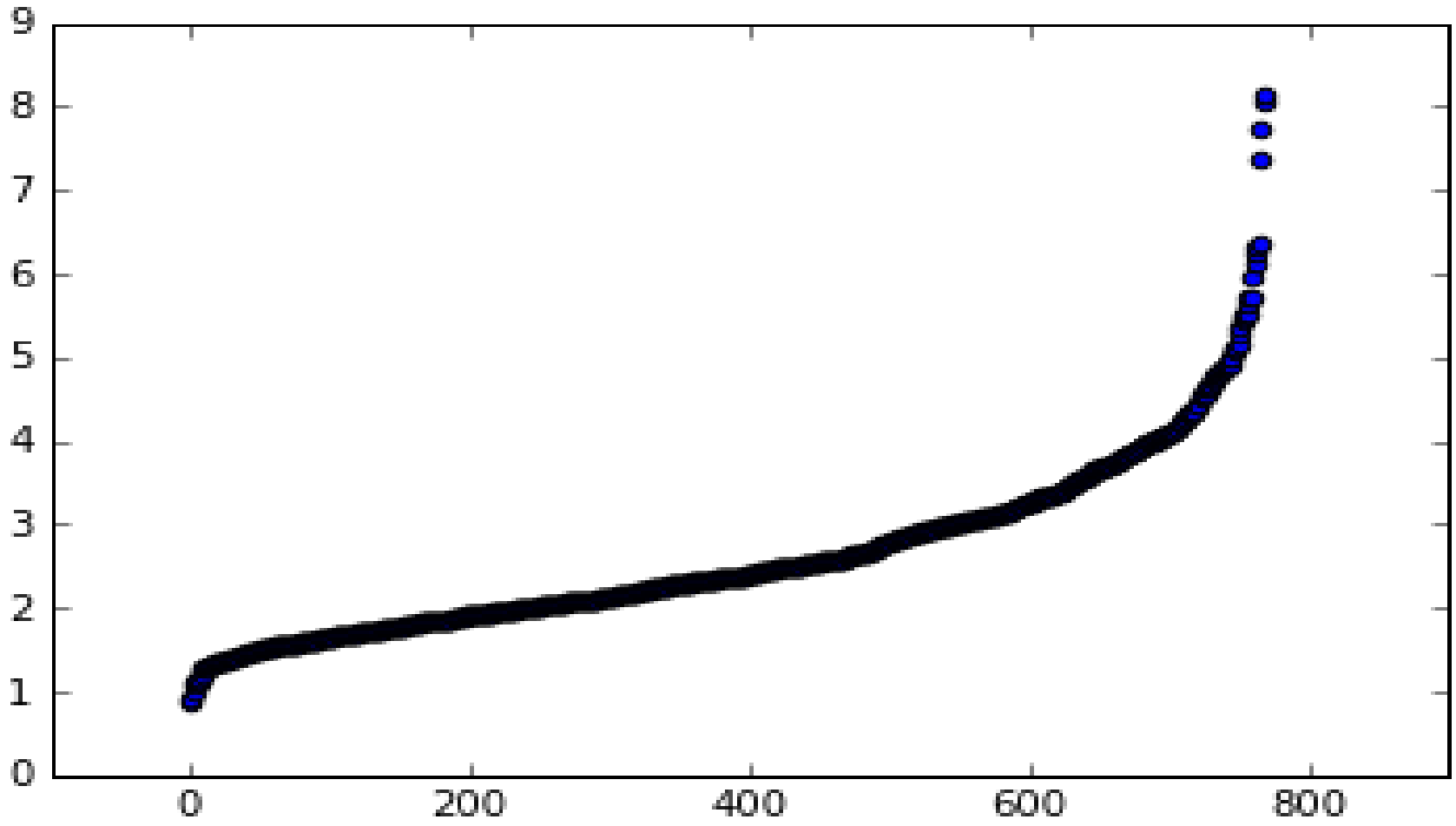
Example: Detecting outliers using the Mahalanobis distance

First, we exclude the classes from the dataset.
Second, we compute the Mahalanobis distance of all the instances with respect to the centroid.
Third, to determine the threshold to decide about the outliers, either we plot all the distance Mahalanobis and find the upper gap in the plot or draw a boxplot of all distance Mahalanobis and find the outliers automatically.



Boxplot for the Mahalanobis distance for all instances in Diabetes

Outliers in Diabetes according to Mahalanobis distance



Top 10 outliers in class 1 of the Diabetes dataset using the Mahalanobis Distance as the outlyingness measure

Instance	Outlyingness
229	76.17
248	49.67
372	43.49
454	43.07
685	33.95
59	33.38
538	33.12
8	31.27
337	30.68
704	29.54

Two effects of multivariate outliers

Masking effect. It is said that an outlier masks a second one that is close by if the latter can be considered an outlier by itself, but not if it is considered along with the first one.
Equivalently after the deletion of one outlier, the other instance may emerge as an outlier.

Swamping effect. It is said that an outlier swamps another instance if the latter can be considered outlier only under the presence of the first one. In other words after the deletion of one outlier, the other outlier may become a “good” instance.

To deal with these effects a robust estimator of the Mahalanobis distance is recommended.

Robust estimator of multivariate location and covariance matrices

- ***The Minimum Volume Elipsoid*** (MVE) estimator, Rousseeuw(1983).
- The ***Minimun Covariance Determinant*** (MCD) estimator, Rousseeuw (1983).
- ***The Donoho-Stahel estimator (1981).***

The ***Minimum Volume Elipsoid (MVE)*** *estimator* is the center and the covariance of a subsample size h ($h \leq n$) that minimizes the volume of the covariance matrix associated to the subsample. Formally,

$$\text{MVE}=(\ \bar{\mathbf{x}}_J^*, S_J^* \ )$$

where

J={set of h instances: $Vol(S_J^*) \leq Vol(S_K^*)$ for all K s. t. #(K)= h}.

$Vol(S_k)=\{|S_k| med_{i=1,2\ldots h} \ d_i^2\}^{1/2}$, d_i represents the Mahalanobis distance of the i-th instance in S_k

The elipsoid is defined by $(x-\bar{x})'S^{-1}(x-\bar{x}) \leq a^2$.

The value of h can be thought of as the minimum number of instances which must not be outlying and usually h=[(n+p+1)/2], where [.] is the greatest integer function and p is the number of predictors.

The ***Minimun Covariance Determinant (MCD)*** *estimator* is defined by

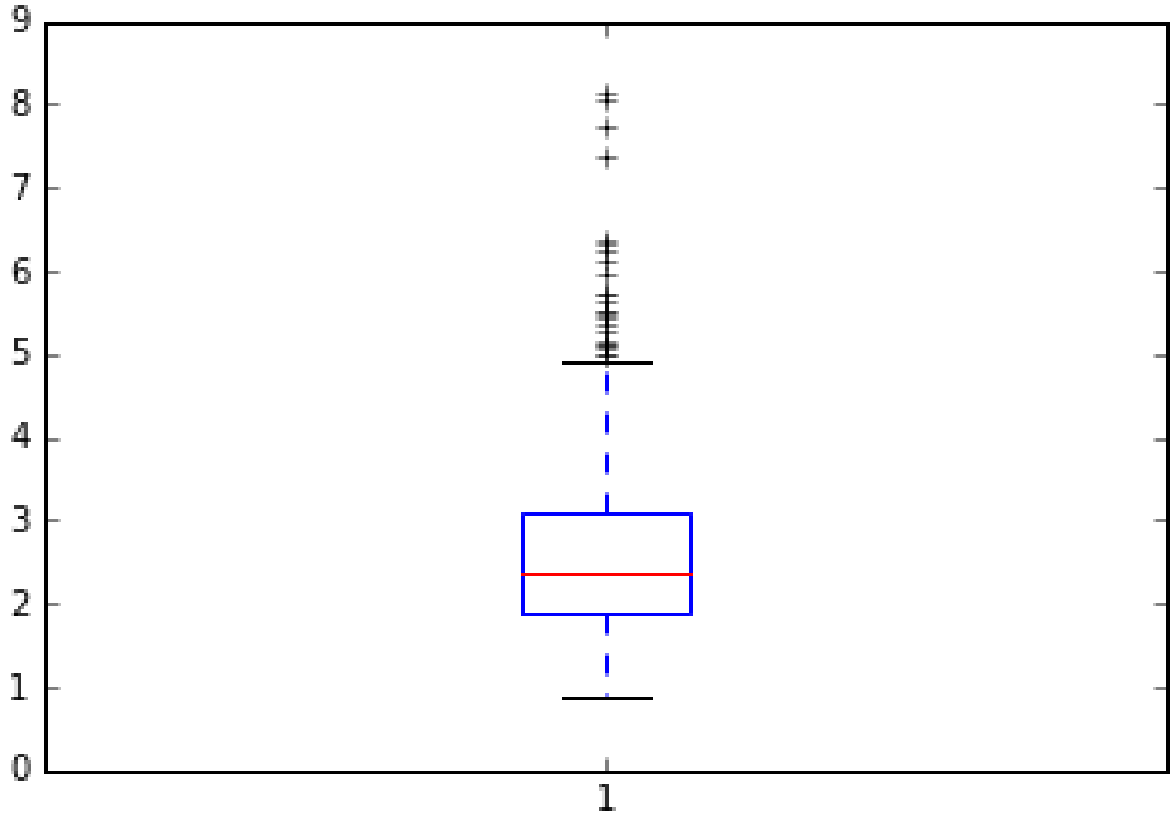
$$MCD=(\ \bar{\mathbf{x}}_J^*, S_J^*\ )$$

where J={set of h instances:$|S_J^*| \leq |S_K^*|$ for all K s. t. #(K)= h}.
El ellipsoid containing the J instances is defined as the MCD estimator and |S| denotes the determinant of S.
The Idea is that only datapoints outside the ellipsoid are outliers.

The time complexity is of polynomial order. More precisely is $O(N^{d\hat{}2})$, where N is the number of datapoints and d is the number of features (Bernholt and Fisher, 2004).

Example: Detecting outliers using Mahalanobis distance with robust estimation of the mean and covariance

First, we exclude the classes from the dataset.
Second, we compute the robust Mahalanobis distance of all the instances with respect to the robust centroid.
Third, to determine the threshold to decide about the outliers, either one can plot all the robust Mahalanobis distance (MCD) and find the upper gap in the plot, or draw a boxplot of all robust Mahalanobis distance and find out the outliers automatically.



Boxplot for the Mahalanobis distance for all instances in Diabetes

Top 10 outliers in class 1 of the Diabetes dataset using the MCD estimator as the outlyingness measure

Instance	Outlyingness
454	220.60
538	142.86
460	141.08
295	131.72
229	129.80
685	126.69
457	95.12
124	90.15
337	83.93
490	76.22

SVM one class

The detection of outiers is done in the context of unsupervised classification. The outliers are classified in the class labeled as -1 and the inliers(normal data points) are classified in the class 1.
The nu parameter, corresponds to the probability of finding a observation outside the frontier. The parameter gamma is an scale parameter and usually gamma=1/n_features

```
from sklearn.svm import OneClassSVM
clf = OneClassSVM(kernel='rbf', nu=.05, gamma=0.125)
clf.fit(Xd)
pred=clf.predict(Xd)
outsvm=inst[np.where(pred==-1)]
print(outsvm)
[ 8 13 22 60 62 75 81 111 146 153 182 186 220 228 247 286 319 332 342 349 409 415 426 486 489
494 502 537 549 579 584 589 604 655 661 695 753 759]
```

Top 10 outliers in class 1 of the Diabetes dataset the outlyingness measure is SVM-one-class

Instance	Outlyingness
229	0.265
372	0.263
685	0.261
454	0.253
488	0.248
745	0.248
146	0.244
520	0.243
248	0.240
59	0.237

Outlier Detection using Isolation Forest

Isolation Forest(IF), similar to Random Forest, are based on decision trees. Since there are no pre-defined labels it is an unsupervised model.

The basic Idea to detect outliers is that them are the data points that are “few and different”.

In an Isolation Forest, randomly sub-sampled data is processed in a tree structure based on randomly selected features. The samples that go deeper into the tree are less likely to be outliers as they required more cuts to isolate them. Similarly, the samples which end up in shorter branches indicate outliers as it was easier for the tree to separate them from other observations.

```
model=IsolationForest(n_estimators=50, max_samples='auto',contamination=float(0.1),max_features=1.0)

model.fit(Xc1)

scores=model.decision_function(Xc1)
```

The score of an instance depends on the length of the search path in the tree of the given instance. A score closer to one indicates an anomaly.

Top 10 outliers in class 1 of the Diabetes dataset the outlyingness measure is IF score

Instance	Outlyingness
229	0.10
454	0.085
248	0.078
745	0.069
29	0.066
488	0.064
59	0.064
538	0.062
764	0.060
372	0.060

Outlier Detection using clustering

Scattered outliers will form a cluster of size 1 and clusters of small size can be considered as clustered outliers. There are a large number of clustering techniques. Here, we only considered k-means and DBSCAN methods.

Top outliers in the first class of the Diabetes dataset detected by the k-means algorithm

A large numbers of clusters is formed , say 25 clusters. Small size clusters are the one including outliers

```
#Finding the outliers, joining the elements of all the small clusters
inst1=Xc1.index
out11=inst1[np.where(clustlabels==0)]
out12=inst1[np.where(clustlabels==11)]
out13=inst1[np.where(clustlabels==16)]
out14=inst1[np.where(clustlabels==23)]
out15=inst1[np.where(clustlabels==27)]
out1all=np.concatenate([out11,out12,out13,out14,out15])
print out1all
```

```
[75,182,342,49,60,81,426,494,522,228,336,453,58,622]
```

DBSCAN

The outliers appear in the cluster with the label -1. The disadvantage of this method is that we have to make a lot of tuning of the parameters eps and Minpts.

```
from sklearn.cluster import DBSCAN
eps=50
dbscan = DBSCAN(eps,min_samples=10).fit(Xd)
dbscanlabels=dbscan.labels_
Inst=Xc1.index
outd1=inst[np.where(dbscanlabels==-1)]
print(outd1)
```

[75, 153, 182, 228, 247, 248, 258, 286, 342, 392, 486, 519, 645,710]

Time complexity of DBSCAN is O(nlogn).

Distance based outlier detection

Given a distance measure on a feature space, two different definitions of distance-based outliers are the following:

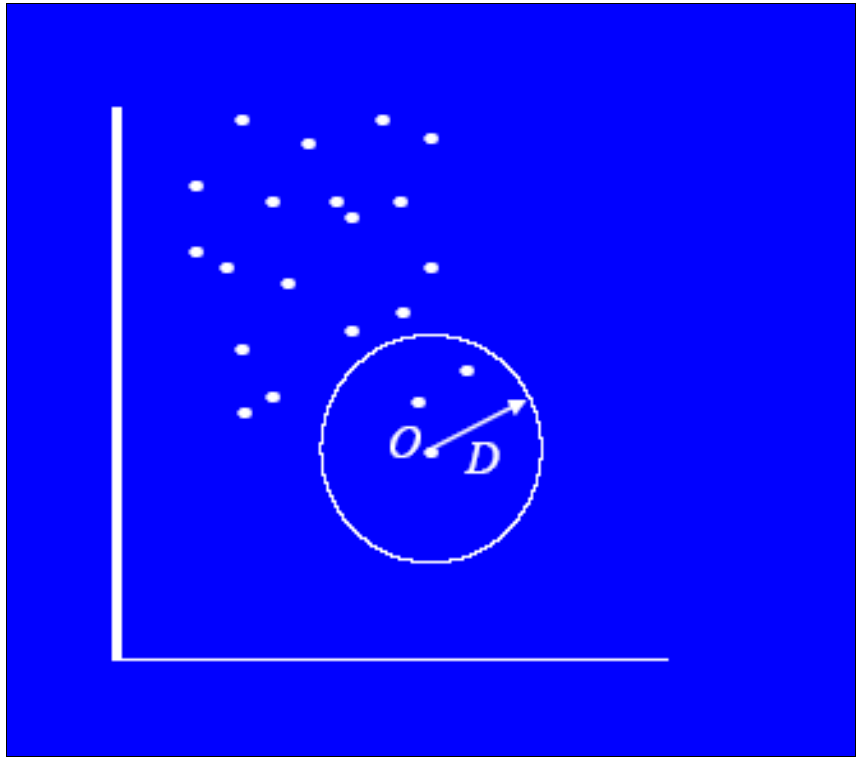
1. An instance **x** in a dataset D is an outlier with parameters p and λ if at least a fraction p of the objects are a distance greater than λ from **x**. (Knorr and Ng, 1997, 1998, Knorr et al. 2000). This definition has certain difficulties such as the determination of λ and the lack of a ranking for the outliers. Thus an instance with very few neighbors within a distance λ can be regarded as strong outlier as an instance with more neighbors within a distance λ.

2. Given the integer numbers k and n (k<n), outliers are the top n instances with the largest distance to their k-th nearest neighbor. (Ramaswamy et al., 2000). One shortcoming of this definition is that it only considers the distance to the k-th neighbor and ignores information about closer points. An alternative is to use the greatest average distance to the k nearest neighbors. The drawback of this alternative is that it takes longer to be calculated.

Distance-based outliers (cont.)

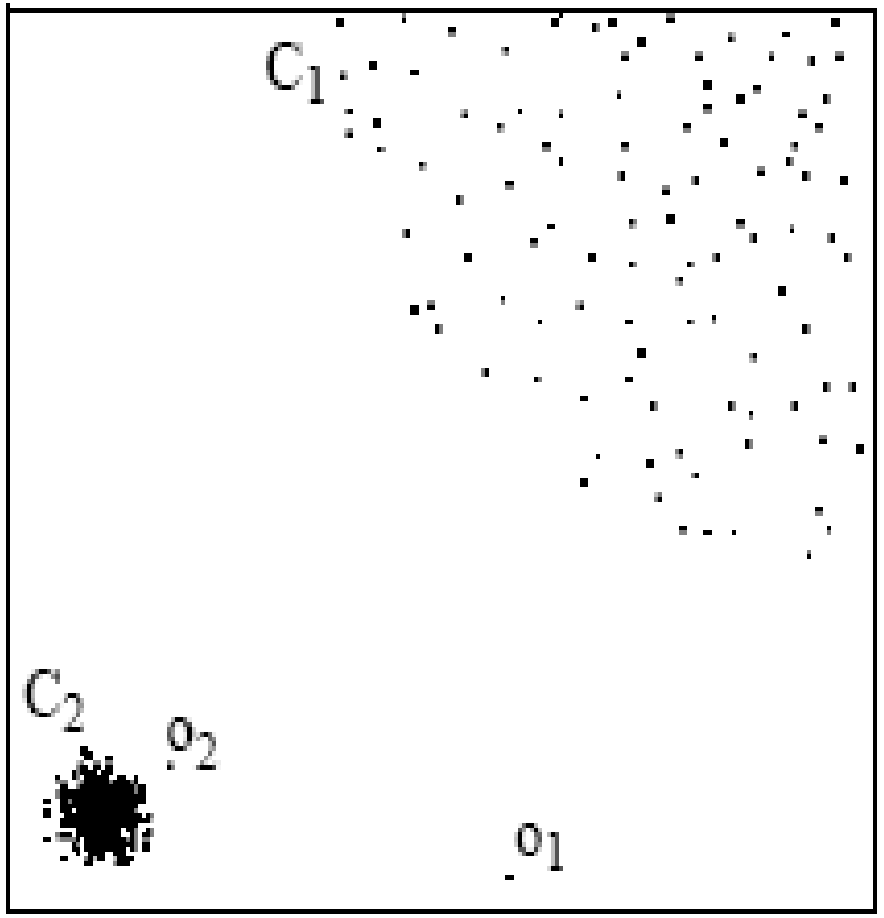
Formally, Object O in a dataset T is DB(p,D) outlier if at least a fraction p of the objects in T are at least distance > D from O.

e.g., DB(.99, 5) implies that 99% of data points are > 5 units distance away



Distance-based outliers (cont.)

- Bay and Schwabacher (2003) proposed a simple nested loop algorithm that tries to reconcile definitions 1 and 2, and at the same time reduce the theoretical time complexity of $O(kn^2)$ to almost linear in n, at least experimentally.
- The algorithm outputs *m* instances that have the greatest distance from their nearest *k* neighbors. The value of *m* is given by the user.



An example showing the weakness of the distance-based method to detect outliers. O1 is detected but O2 is not detected.

Density-based local outliers

In this type of outliers the density of the neighbors of a given instance plays a key role (Breuning et al, 2000). Furthermore an instance is not explicitly classified as either outlier or non-outlier; instead for each instance a local outlier factor (LOF) is computed which will give an indication of how strongly an instance can be an outlier.

Definition 1. *k-distance of an instance x*

For any positive integer k, the k-distance of an instance x, denoted by k-distance(x), is defined as the distance d(x,y) between x and an instance y ε D such that:

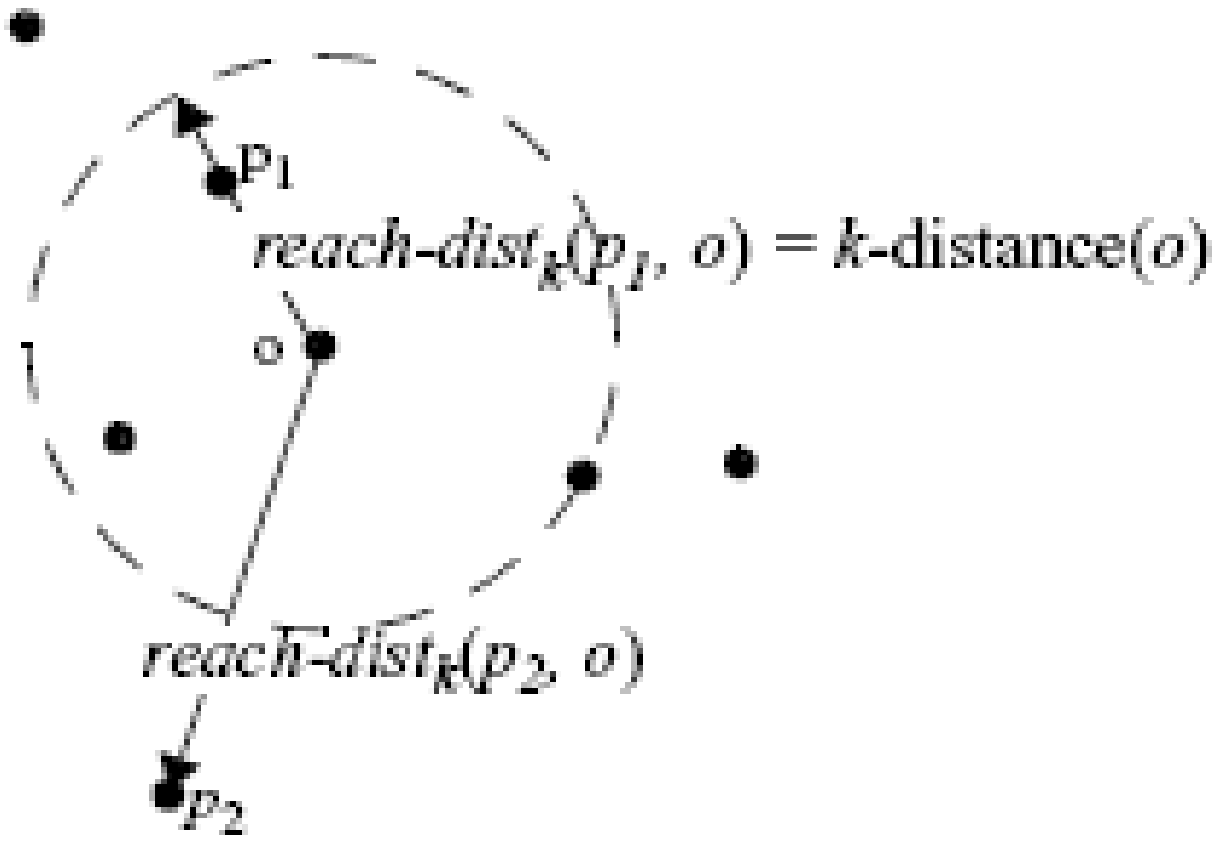
1. For at least k instances y' ε D-{x} it holds that $d(x,y') \leq d(x,y)$.
2. for at most k-1 instances y' ε D-{x} it holds that $d(x,y') < d(x,y)$.

Definition 2: k-distance neighborhood of an object p
Given the k-distance of x, the k-distance neighborhood of x contains every object whose distance from x is not greater than the k-distance, i.e

$N_{k\text{-distance}(x)}=\{y \in D-\{x\} \mid d(x,y)\leq k\text{-distance}(x)\}$

These objects y are called the k-nearest neighbors of x.

Definition 3. *Reachability distance of an instance x w.r.t. instance y*
Let k be a positive integer number. The reachability distance of the instance x with respect to the instance y is defined as $reach\text{-}dist_k(x,y)=max\{k\text{-distance}(y),d(x,y)\}$



Reach-dist$_3$(p_1,o)=3-distance(o) y reach-dist$_3$(p_2,o)=d(p_2,o) for k=3

Local reachability density of an instance x

Given an instance x of a dataset D its local reachability density is defined by

$$lrd_{MInPts}(x) = [\frac{\sum_{y \in MinPts(x)} reach - dist_{MinPts}(x, y)}{|MinPts(x)|}]^{-1}$$

This is the inverse of the average reachability distance based on the *MinPts nearest* neighbor of *x.*

The ***Local outlier factor (LOF)*** *of an instance x is defined by*

$$LOF_{MinPts}(x) = \frac{\sum_{y \in N_{MinPts}(x)} \frac{lrd_{MinPts}(y)}{lrd_{MinPts}(x)}}{|N_{MinPts}(x)|}$$

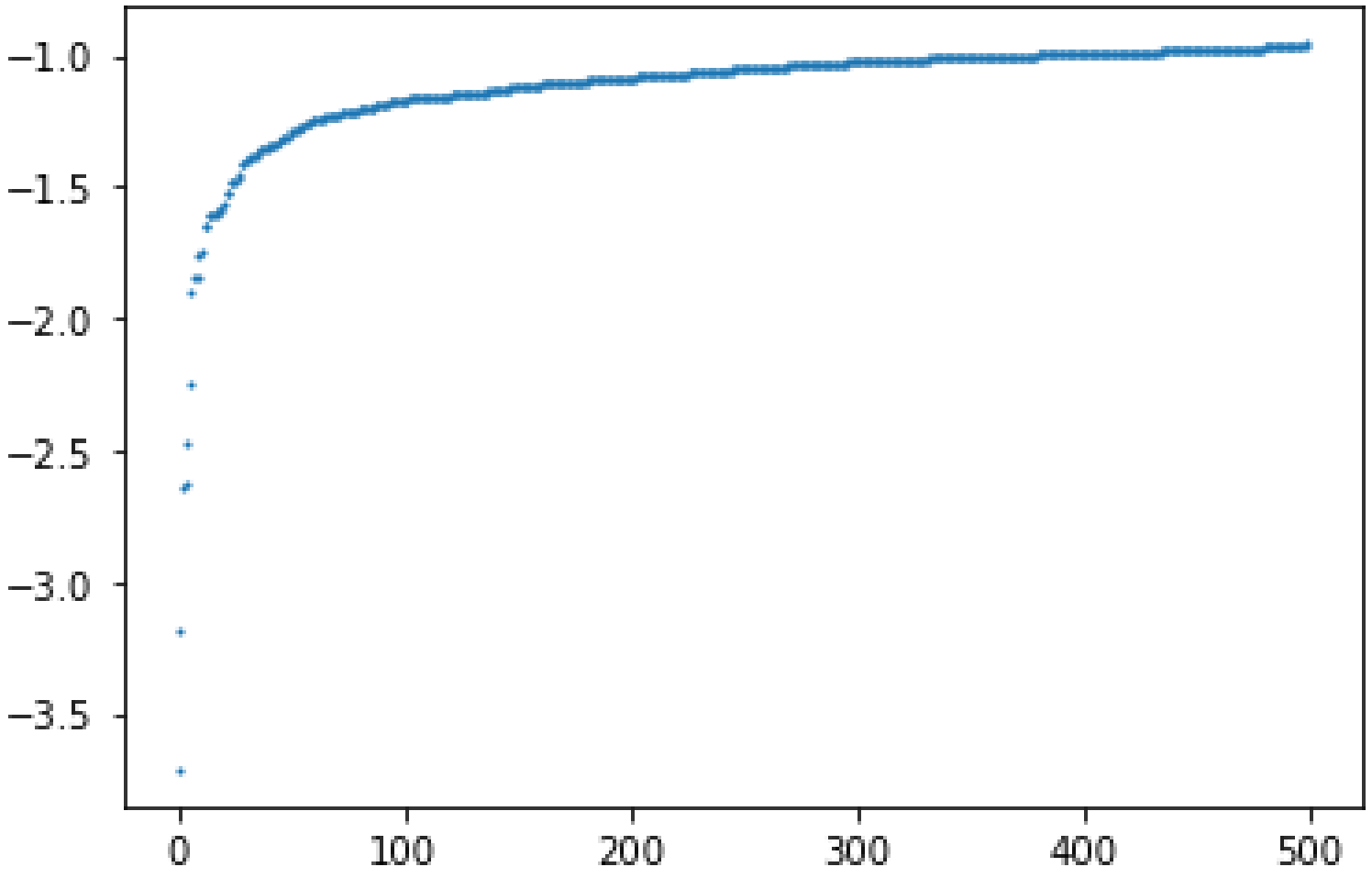
where *lrd(.)* represents the *Local reachability density of an instance.*

Algorithm for detection of density-based local outliers

Input: Dataset D, MinptsLB, MinptsUB

Let Maxlofevct=ϕ
For each i in the interval [MinPtsLB, MinPtsUB]
{
1. Find the i nearest neighbors and their distance from each instance in D
2. Calculate the local reachability density for each instance in D
3. Compute the lof of each instance in D
4. Maxlofvect=max(maxlofvect, lof)
}
end
Output: Maxlofvect; the vector of maximum LOF's

Time complexity is $O(n^2)$



Plot of the instances in Diabetes (class 1) ranked according to the LOF outlyingness measure

Top 10 outliers in class 1 of the Diabetes dataset the outlyingness measure is LOF

Instance	Outlyingness
229	3.71
248	3.18
343	2.63
76	2.62
183	2.47
287	2.24
154	1.89
487	1.84
460	1.84
107	1.75

Data Mining and Machine Learning

Outlier Detection (Novelty Detection)
using autoencoders

Dr. Edgar Acuna
Departmento de Matematicas

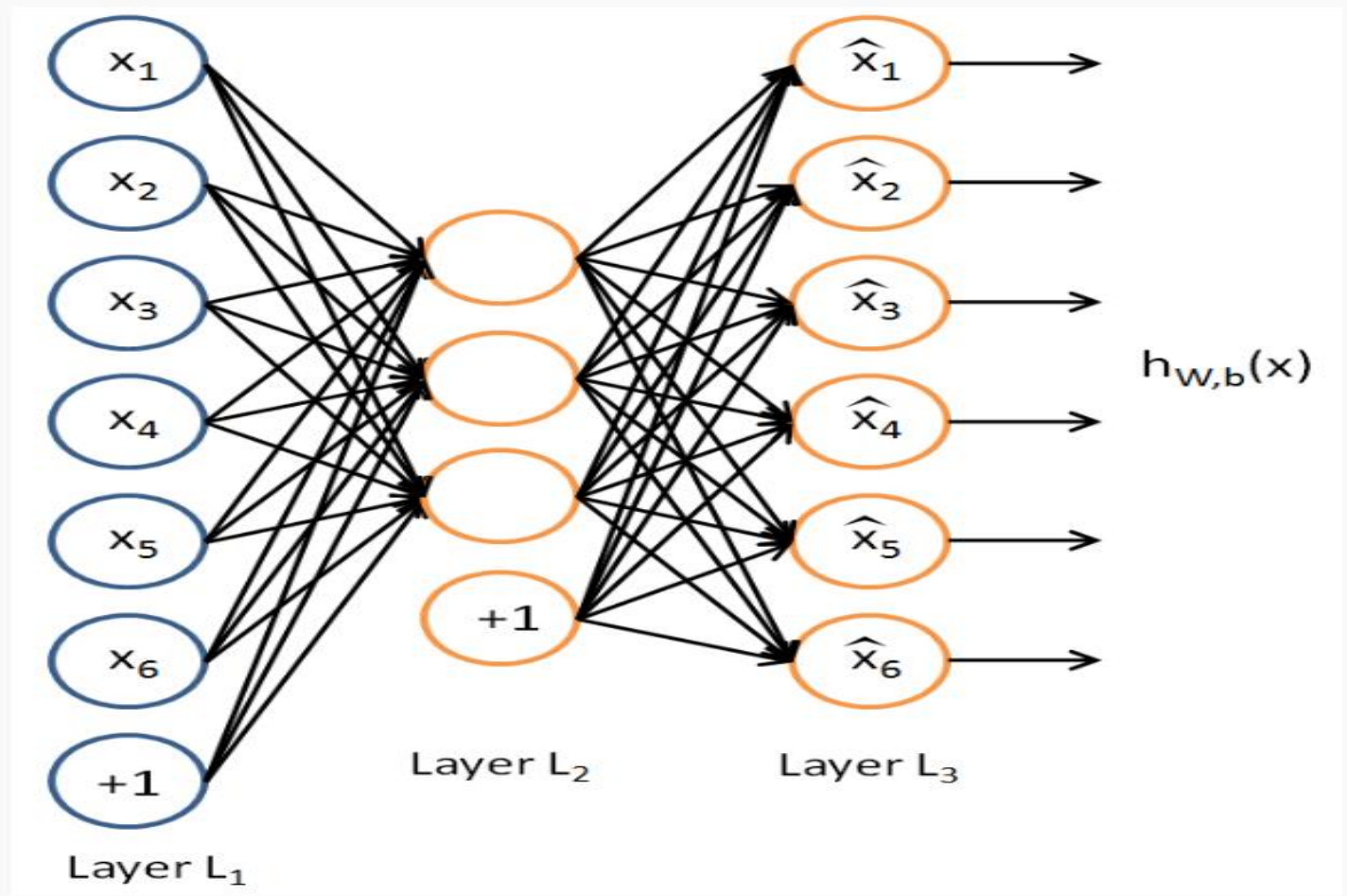
Universidad de Puerto Rico- Mayaguez

academic.uprm.edu/eacuna

Autoencoders

- Autoencoders are:
- feed Forward Neural Network models for unsupervised tasks (No Labels).
- Applies backpropagation, setting the target values to be equal to the inputs.
- simple to understand!

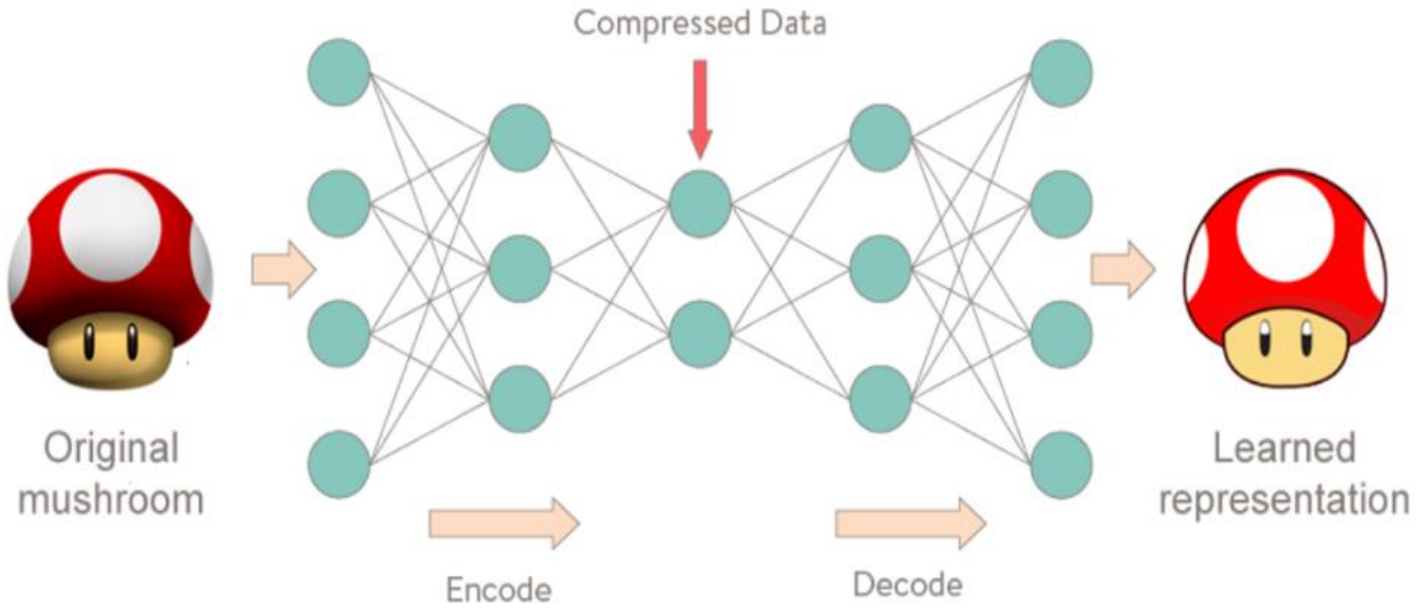
Example of autoencoder



Autoencoders are used for:

- Compressing data and learn some features, especially in non-structured data(e.g images).
- Anomaly (outlier) detections (Fraud detection is one area of application).
- Data Augmentation using the reconstructed data.

Autoencoder with 5 layers



The autoencoder model tries to minimize the reconstruction error (RE), which is the mean squared distance between input and output.

$$L(x, x') \approx ||x - x'||^2$$

Reconstruction Error

The model will train on the normal dataset by minimizing the RE. It is expected that the RE to be relatively high when it is tested on a new abnormal datapoint (outlier)

However, one has to do tuning on the RE's threshold to detect outliers

Detecting outliers for Diabetes using autoencoders

The neural network model is built by Keras. The optimized Autoencoder model has 3 fully connected hidden layers, which has 4,2,4 neurons respectively. The input and out layers have 8 neurons each for the 8 features.

The tangent hyperbolic (tanh) activation function is used in the first and third layer and the relu activation function in the second and fourth layer. The model is trained for 20 epochs and used a batch size of 50, and 10 % of the training data is used for validation during the training process.

Outliers detected on Diabetes using autoencoders

```
Using all the data
4, 12, 13, 39, 43, 45, 86, 88, 159, 177, 186, 193, 215,
228, 247, 259, 270, 298, 357, 362, 370, 375, 445, 453,
458, 459, 487, 542, 558, 579, 590, 674, 691, 740, 744, 763


 Using train and test datasets
 445, 428, 123, 228, 274, 660, 517, 323, 339, 100, 588, 708,
 691, 582, 614, 84, 24, 459, 509, 319, 12, 612, 672, 298
```

Summary of outliers in Diabetes (class=1)

Method	Outliers
Mahalanobis	229, 248,372,454,685,59,538,8,337,704
MCD	454,538,460,295,229,685,457,124,337,490
K-means	76,183,343,50,61,82,427,495,523,229,337,454,59,623
Dbscan	76, 154, 183, 229, 248, 249, 259, 287, 343, 393, 487, 520, 646,711
LOF	229,248,343,76,193,287,154,487,460,107
Isolation Forest	229,454,248,745;29,488,59,538,764,372
SVM-Oneclass	229,372,685,454,488,745,146,520,248,59
Autoencoders	13, 55, 58, 59, 87, 154, 212, 224, 229, 248, 276, 287, 359, 363,372, 379, 454, 460, 480, 487, 488, 533, 549, 658, 685, 745, 748, 764

Outliers in the Diabetes Dataset

Without considering classes:
9,14,76,229,248,333,349,454,503,580,585

Outliers in class 1 based on robust Mahalanobis distance:
229 248 454 59

Outliers in class 2 based on robust Mahalanobis distance
14 194 333 446 580

Evaluating the effect of outliers

Two main aspects to consider in supervised classification are the estimation of the misclassification error rate and feature selection.

Three classifiers considered: LDA, KNN and Rpart (a decision tree classifier)

Accuracy estimation method: 10-fold cross validation

Datasets: Diabetes

The accuracy estimated for the LDA, knn and decision trees classifiers on *Diabetes* after deleting ouliers

Classifier	Original Sample	Deleting outliers
LDA	77.00	78.00
KNN(k=5)	72.00	74.00
Decision Trees	73.20	74.00

The outliers in each class were found using clustering by kmeans

Some conclusions

- There is not a unique method to detect all outliers.
- LDA and KNN classifiers seem to be more affected by the outliers than the decision trees classifier.
- It is a good idea to use Visualization to check detected outliers.