**Anomaly Detection**

So far, you have learned many concepts that are there in Machine Learning

The purpose of this lecture is to let you know how you can take the concepts that you already know like Random Forests, SVMs, KNN, etc., and modify them for using them for another purpose other than classification i.e. Anomaly Detection.

**What is an Anomaly?**

* Anomaly is synonymous with an outlier. These terms are often interchanged and may be called Novelty depending on the context.

**What's the difference?**

* Anomaly means something which is not a part of the normal behavior
* Novelty means something unique, or something that you haven't seen before(novel)

**Applications of Anomaly/Outlier/Novelty Detection:**

* Credit Card Fraud Detection
* System Intrusion Detection
* Ecosystem Disturbances in Weather and Environment (alarming about Tsunami, Earthquake, etc.)
* You can read how some startups are thinking out of the box using these techniques from a blog [**here**](https://www.prescouter.com/2021/05/how-can-ai-be-used-in-anomaly-detection/#:~:text=Applications%20of%20anomaly%20detection%20include,health%20data%2C%20and%20predictive%20maintenance.)

## **1. Distribution Based**

* The simplest way for detecting an outlier would be to use distribution parameters (mean and standard deviation).
* Consider a feature X in some observations x1, x2,…., xn with some outliers.
* We know that it will follow some distribution which will have parameters θ.
* Let **x** follow a gaussian distribution with some mean(µ) and standard deviation(σ)
* If we know the distribution of the data, we'll try to fit the distribution. But, the problem arises with the distribution parameters.
* While we know the distribution, the parameter estimates of the distribution are often corrupted by the noise/outlier
* Hence, we need to robustly estimate the parameters of the distribution. We do this using an algorithm called RANSAC which stands for Random Sample Consensus

## **2. Random Sample Consensus (RANSAC)**

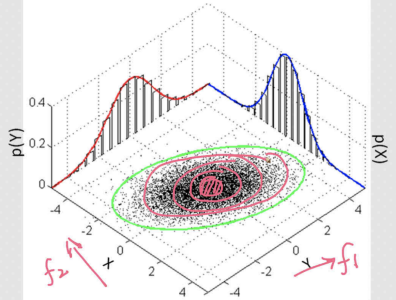
* RANSAC is a trial-and-error approach that works very well in real life.
* Imagine a dataset X with a number of points having parameters µ and σ. Let's call them collectively θ
* There are mainly three steps involved in RANSAC. They are
  + Sample a subset of points from the dataset (n’). We consider this point as an inlier.
  + Now, compute a model that estimates the parameters of the sampled points.
  + Score the model which indicates how many points will support the model.
* We repeat these three steps iteratively and then select the model best supported by the data, which then tells which points are inliers and which are outliers.

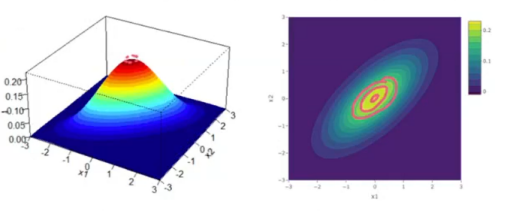
**Extending the idea to higher dimensions**

* Till now, we assumed that we have only a single feature X which was following Gaussian Distribution.
* Now, imagine we have d-dimensional data where each point xi Є Rd and the data is not labeled.
* If we know the data points xis follow multivariate gaussian distribution(unimodal), then X follows normal distribution; X ~ (, Σ), where is mean vector and Σ is a covariance matrix
* Here we'll consider (, Σ) as θ
* In GMMS, in multidimensional space, the shape of gaussian was similar to a hill where the density of the points was highest in the middle contour, and it keeps getting low as we move away from the center
* In this case too, RANSAC can be applied. Farther away from centroid, we'll know that it is an outlier.
* Similar principle is used in another method called Elliptical Envelope.

## **3. Elliptic Envelope**

* We know that a Unimodal Multivariate Gaussian Distribution on a single plane will look like ellipses if visualized on a plane. This idea can be extended to find out an outlier





* Given some data X where xis Є Rd  and X follows Normal Distribution being unimodal, Elliptical Envelope robustly estimates the parameters (, Σ).
* The term robustly means without getting impacted by outliers
* Next, then we remove the points that are outliers which are very far away from the centroid

**What if the distribution is non-gaussian? Do we need to convert it into gaussian distribution?**

* While the elliptical envelope method makes an assumption that the distribution is gaussian, the strategy can be applied to any distribution.
* As long as we know any distribution and its parameters θ, we can extend our strategy to use RANSAC and estimate parameters θ.
* These can be other distributions such as multivariate Poissons, multivariate log normals, etc., but we don't use them as much.
* One other strategy is to convert them into Gaussians but we don't necessarily have to.
* As long as there is any distribution we can calculate the probability of xi Є X using PMF/PDF.

**Sklearn walkthrough:**

Scikit-learn implements **EllipticEnvelope** as a part of the **covariance** module. Let's walk through the parameters that are important:

1. **assume\_centered**: It is for assuming that the data is centered at 0, i.e. 𝜇⃗ =0.

By default, it is set as 𝐹𝑎𝑙𝑠𝑒. If set to 𝑇𝑟𝑢𝑒, it will just estimate the covariance matrix.

It uses the FastMCD approach and not the RANSAC approach. MCD refers to the Minimum Covariance Determinant.

FastMCD performs in a similar way to RANSAC where it takes a subset of points and using them tries to estimate the distribution parameters robustly.

2. **support\_fraction**: It tells how many points to use to estimate the parameters.

3. **contamination**: It says what percentage of our data we think are outliers.

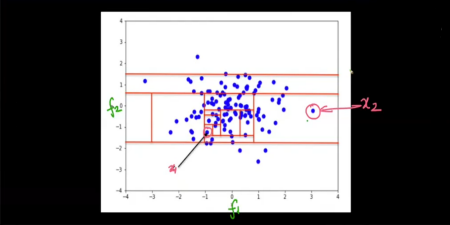
It takes values from 0 to 0.5, where 0.5 represents that 50 % of our data is noisy. The default assumption value is 0.1 ~ 10 %

**Disadvantages:**

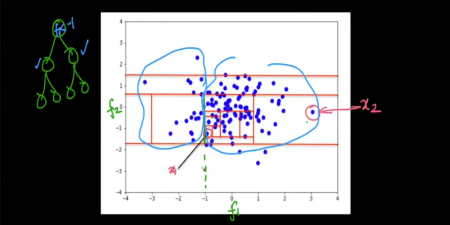
* It cannot be used for non-unimodal data
* It is specifically for multivariate Gaussians
* If the data fails to meet the assumptions of unimodal and multivariate gaussian, the whole thing crashes.

## **4. Isolation Forests**

* Consider a dataset D which contains data points x1, x2,…., xn. Just like Random forests, Isolation Forests build many trees.
* Following are the steps involved in Isolation Forest:
  + Build many trees like random forests
  + For each tree:
    - Randomly pick a feature
    - Randomly threshold that features
    - Build each tree until the leaf consists of only one datapoint
* Isolation Forests are also known as iForests
* Consider the plot along feature f1 and f2 given below:



* In isolation forests, we are building totally random trees. So if we pick feature f1 and put a threshold there will be a vertical bar.
* Similarly, if we pick feature f2 and put a threshold there will be a horizontal bar.
* For example, if we pick feature f1 and we select threshold as f1 < 1, then our first root node will be based on this condition



* Based on the diagram above,
  + The node containing x1 will be at more depth.
  + Observe that the point x1 is in a dense region, and point x2 is far away
  + That is because, to break the point x1 from all the other points, more and more splits will be required and that will increase the depth of the node containing point x1.
* So, to sum it up, the idea behind Isolation forest is:
  + On average outliers have lower depth in the random trees
  + On average, inliers have more depth in the random trees

**Evaluation of Isolation Forest**

* Imagine, we have to build random trees. For each point xi in the dataset, we can get an average depth.
* We use this average depth to convert it into a metric.
* Apart from this, there are a lot of different metrics, that people have come up with over the years
* But, the basic intuition is that the lesser the average depth, the higher the likelihood is there that it is an outlier

**Deciding average depth of a point:**

* There are a lot of metrics that researchers have come up with over the years.
* But, studying them in this lecture is out of scope.

**Sklearn walkthrough**

We can implement Isolation Forest with the help of sklearn's **IsolationForest** method present in the **ensemble** module.

Let's see some of the parameters that IsolationForest expects:

1. **n\_estimators**: It represents the number of base learners. By default, the value is set equal to 100

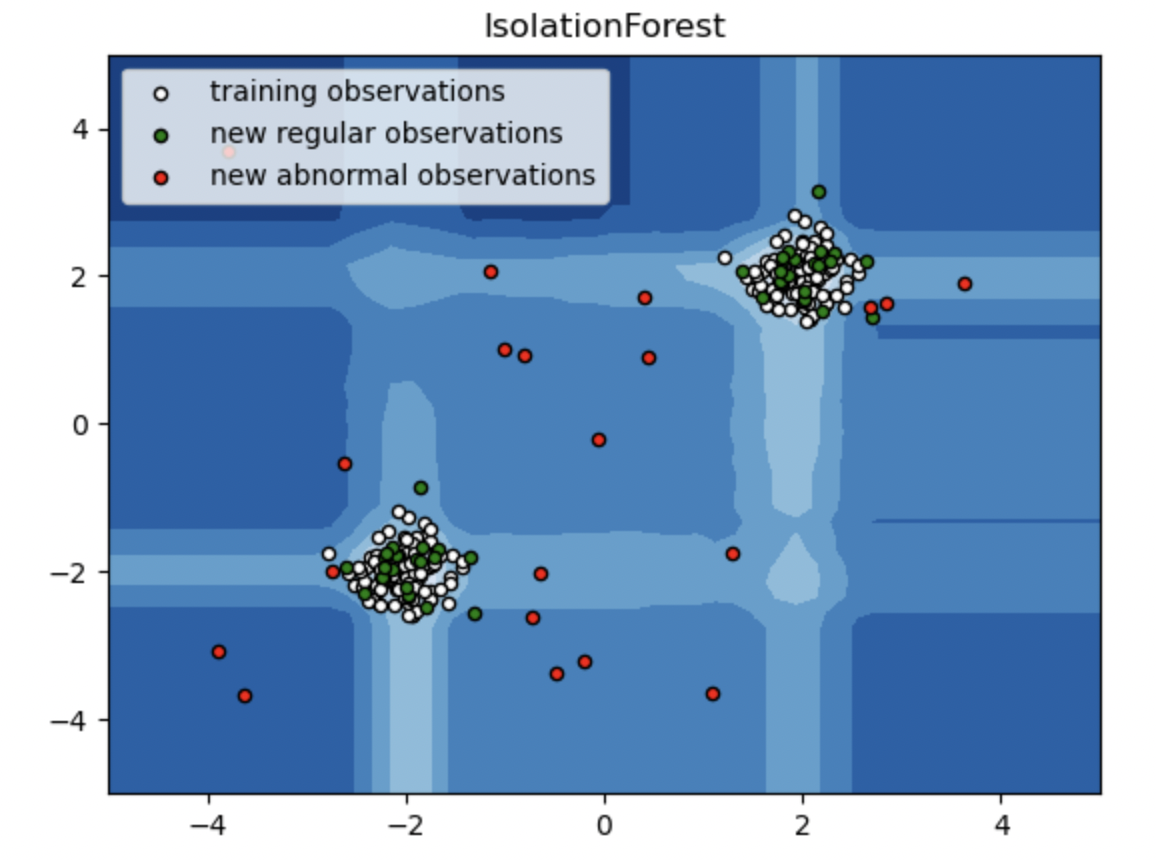
2. **max\_sample**: It is the number of samples to extract from the dataset to build the trees(row sampling). By default the value is set to auto and sklearn picks reasonably a good figure for iForests

3. **contamination**: It tells the proportion of outliers in the data. The range is between [0,0.5]

4. **max\_features**: It is the number of features to extract from the dataset to build the trees(column sampling).

**Disadvantages**

* One of the major limitations of iForests is that they are biased towards axis parallel splits.
* iForests makes splits and these splits are always parallel to either of the axis.
* Because of this, the boundary will not be smoothened.
  + In the diagram given below, the different shades of blue represent the likelihood of a point is an outlier. The darker the color, it is more likely that the point in that region will be an outlier
  + We've trained the iForest model using training data(white points)
  + It is tested on testing data(red + green) where red color indicates outliers

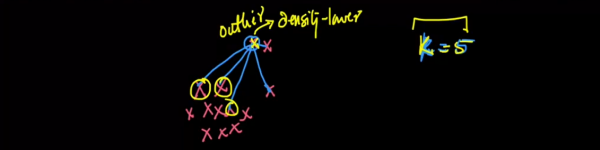


* Now imagine two points x1 and x2 as shown in the diagram given below.
* Both the points are almost equidistant from the nearest cluster. x1 is on the axis and point x2 is off-axis.
* Because the model is biased towards the axis, it will classify the point as an inlier and as an outlier
* This is also known as banding in signal processing

### 

## **5. Local Outlier Factor (LOF)**

* On a higher level, LOF is based on two ideas: KNN and density
* The core idea behind LOF is to compare the density of a point with its neighbors' density
* If the density of a point is less than the density of its neighbors, we flag that point as an outlier
* Imagine a bunch of datapoints as shown below



* We compute the density of a point based on average distance.
* If the average distance between a point and it's **K** nearest neighbors is large, it is more likely that the point will be an outlier
* Also, the larger the value of **K**, the more confident are the results.

Some concepts to understand the working of LOFs:

**1(a) K-distance**

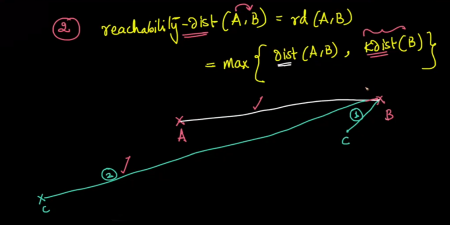
* We define K-distance of a point **A** as the distance of point **A** to its **Kth** nearest neighbor
* In general, the larger the value of k-distance is, the farther away the point is from other datapoints

**1(b) Set: Nk (A)**

* It is a set of k-nearest neighbors of point **A**.

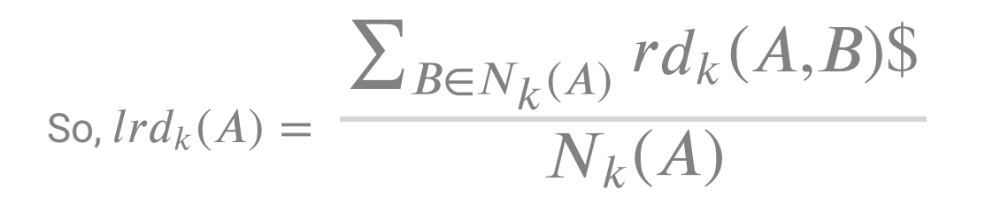
**2. Reachability distance**

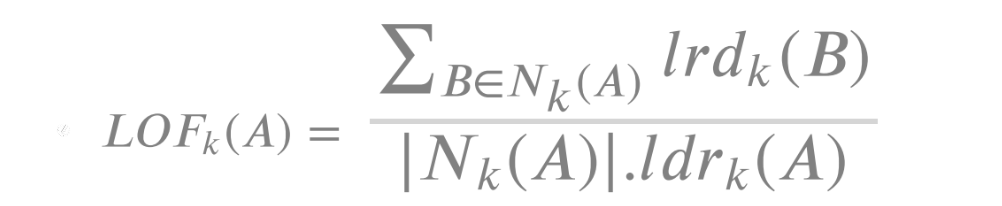
* From point **A** to point **B**, we define reachability distance as a maximum of the distance from point **A** to point **B** and the maximum k-distance of point **B**
* Consider point **B** with some **k** nearest neighbors shown in the diagram below.

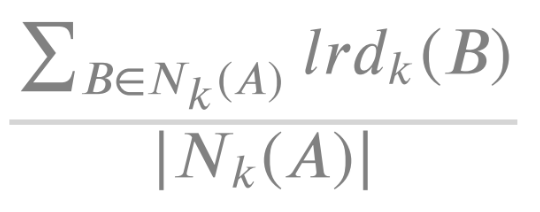


* There is a possibility that some neighbors might be close(condition 1) and some neighbors might be very far away(condition 2)
* In this case, there is a neighbor of point **B** whose k-distance is greater than the distance between point **A** and **B**, and hence, it is considered as its reachability distance.

**3. Local Reachability Density**

* It is often represented as **lrdk(A)**, which tells the local reachability density of a point **A**
* It is defined as the average reachability distance between point **A** and **k** neighbors
* 
* The summation in the above equation represents the sum of reachability distances from a point **A** and set of neighbors **B** as **B** Є **Nk(A)**
* We define Local Outlier Factor of point as follows:



* **lrdk(A)** is the density of point **A**
* The expression  is the average neighborhood density
* So, LOF of point **A** is nothing but the average neighborhood density(lrd) of point **A** divided by the density of **A**

**Interpretation of LOF**

* If **LOF(A) = 1**, then we can say that the point has the same density(lrd) as its **k** nearest neighbors
* If **LOF(A) > 1**, then the **k** neighbors of point **A** have a higher density than point **A**.
  + That does not mean point **A** is an outlier. It may or may not be.
  + But if **LOF(A) >>> 1**, then the point is definitely an outlier.
* If **LOF(A) < 1**, then the point has more density than its nearest neighbors.

**Disadvantages of LOF**

* Finding optimal K
* Finding threshold.
  + If LOF(A) >> 1, what is the threshold??
* Cannot handle high dimensional data efficiently
* High Time Complexity