INFO 4000
Informatics III

Data Science Specialization - Advanced

Week10 Anomaly Detection & algorithms

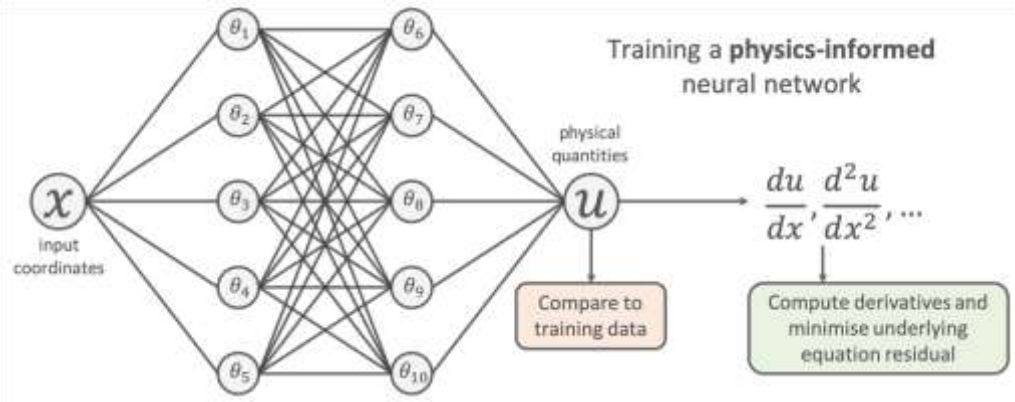
Course Instructor

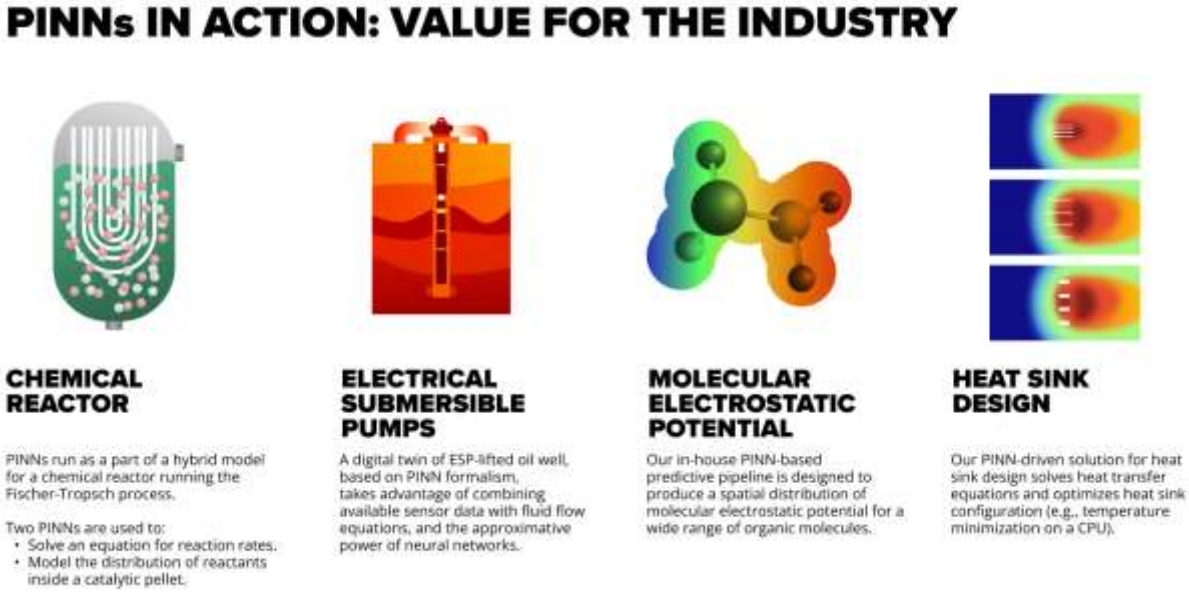
Jagannath Rao

raoj@uga.edu

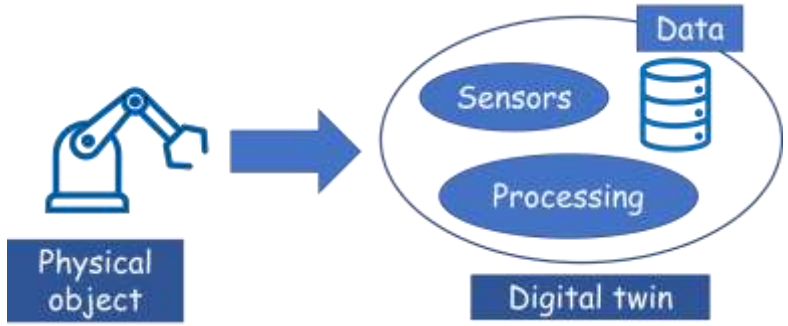
Recap

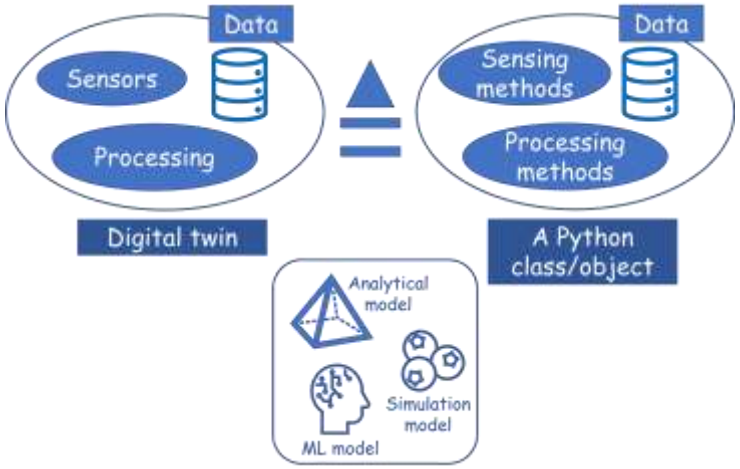
PINN – Great when we have an idea about the underlying physics

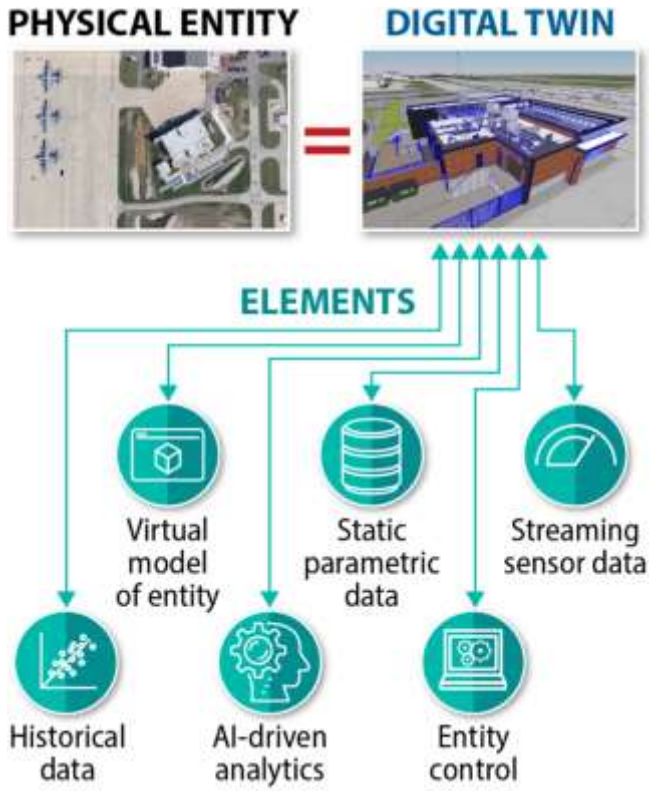


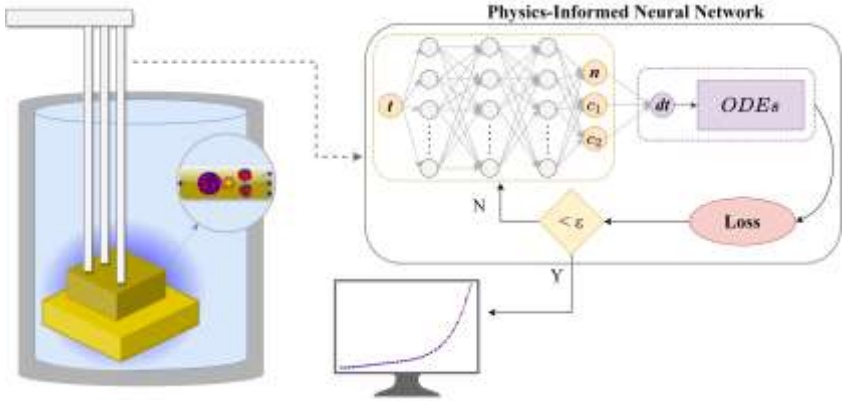


Digital Twin









We will expand a little more on Digital Twins in the next weeks to get ready for MP3

Anomaly detection in Industry

What is anomaly detection?

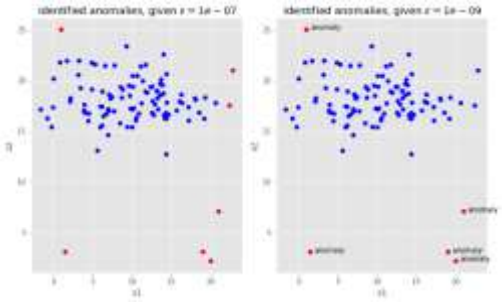
- Fighting fraud, defending networks, finding outliers, and flagging faulty equipment may all seem like different problems, but they are not.
- They share one common trait - no clear labels or definitions of what constitutes an anomaly.
- One way to develop anomaly detection systems is to use *predictive* ML.
  - For example, we could collect fraudulent transactions.
  - Then, using features like transaction amount, time, location, and type of merchant,
  - we can build a model that distinguishes these from normal transactions.

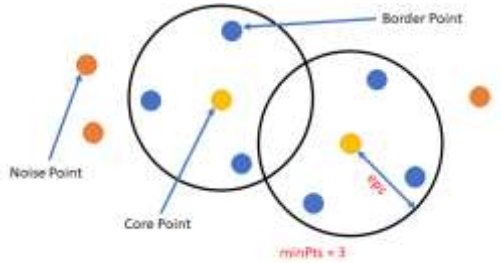
What is the challenge in using a predictive model to detect anomalies??

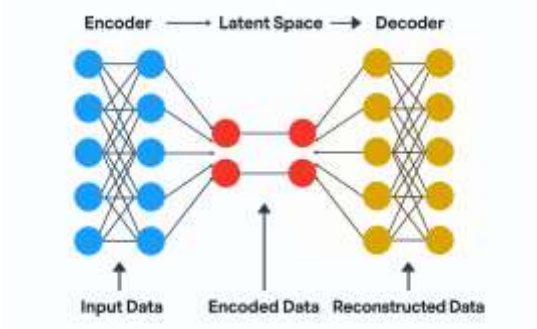
- First problem with this approach is – we can only detect fraud similar to the one's we found.
- Fraudsters will often change their strategies, machines can fail due to many reasons, and these can be unexpected.
- This is why unsupervised methods are often used to detect anomalies.
  - They work by comparing all transactions and identifying ones that have unusual feature values,
  - Importantly, this means we do not have to label any transactions beforehand.
- Anomalies:
  - In manufacturing, machines can fail at any time in a variety of unexpected ways.
  - In cyber security, novel attacks are constantly being found and patched.
  - In medicine, rare health issues can be hidden among many test results.

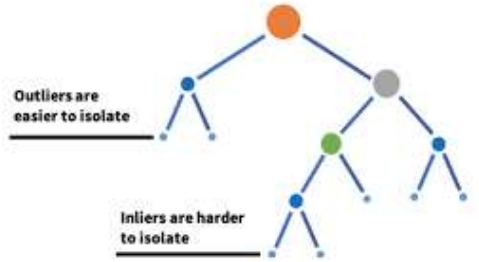
Algorithms for anomaly detection

- Gaussian methods
- DBSCAN – A form of hierarchical clustering algorithm
- Isolation Forests – A tree-based algorithm
- Auto Encoders – A neural network-based algorithm





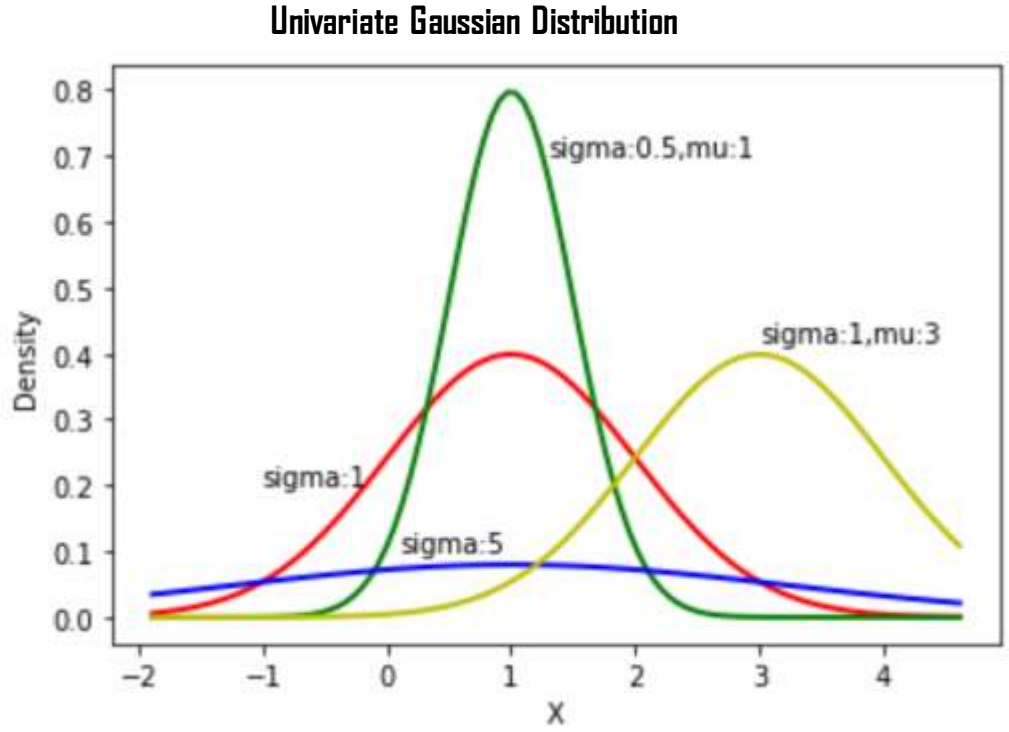




1. Gaussian Methods

Gaussian distributions also known as Normal distribution

- These are represented as Probability Distributions.
- μ is the mean of the distribution and typically centered at zero, whereas σ^2 is the variance and σ (square root of variance) is the standard deviation.
- The area under the curve = 1 is the total probability and cannot be > 1.
- Every point on this curve is a probability density point.

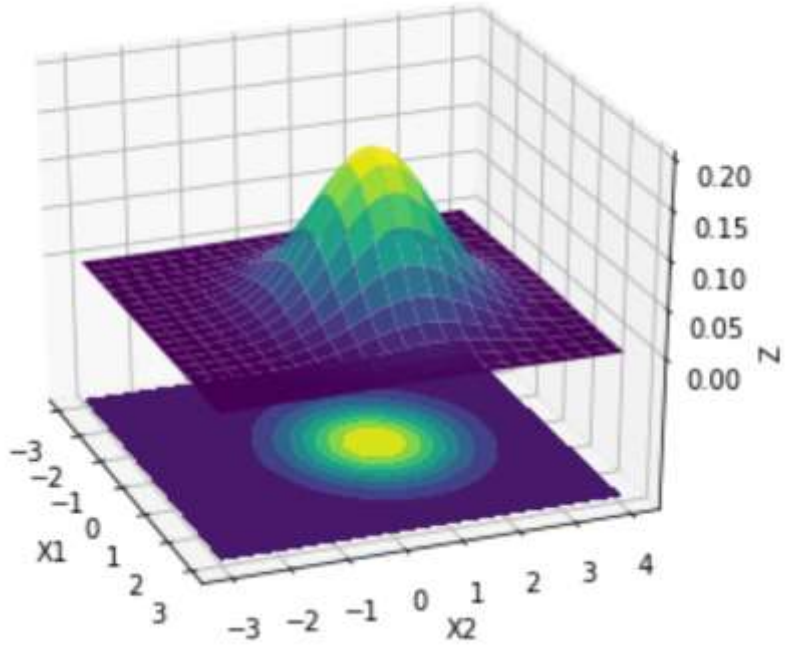


$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

Multivariate Gaussian distributions

- As the name suggests, this distribution models data which has more than one variable.
- For example, you will have a variable X1 on the x-axis and X2 on the y-axis and so on for 'n' dimensions.
- X, μ, Σ are all matrices / vectors and Σ is also known as the covariance matrix
- The plot on the right shows the gaussian for 3-dimensions and it also shows the projection of the contours on a plane (2-D).
- Unlike the univariate version (which by the way is a specials case of the multivariate) the orientation of the gaussian can be changed to fit different needs.

Multivariate Gaussian Distribution



$$p(X;\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^n|\Sigma|)}}\exp(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu))$$

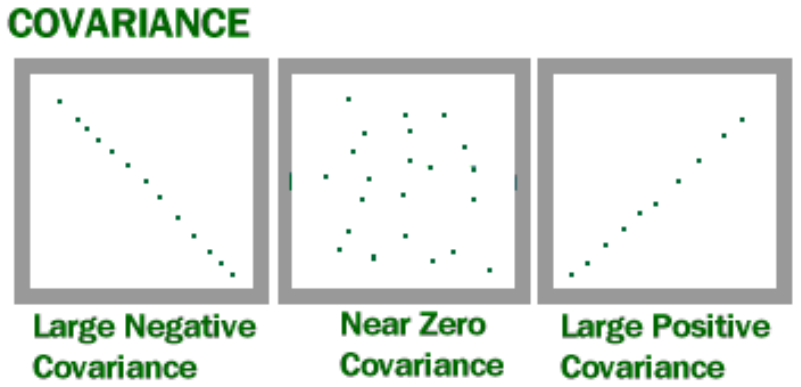
Covariance – What is it?

- We can say that positive covariance implies a direct relationship between the variables (increasing x increases y), and a negative covariance implies an indirect relationship between the variables (increasing x decreases y).

- Correlation on the other hand not only gives us the direction but also gives the strength of the correlation on a scale of 1 to -1.

- Covariance is always represented as a matrix called the Covariance Matrix and its always a square matrix (2x2 if there are 2 features) and the diagonal elements represent the variances of each feature in a dataset.

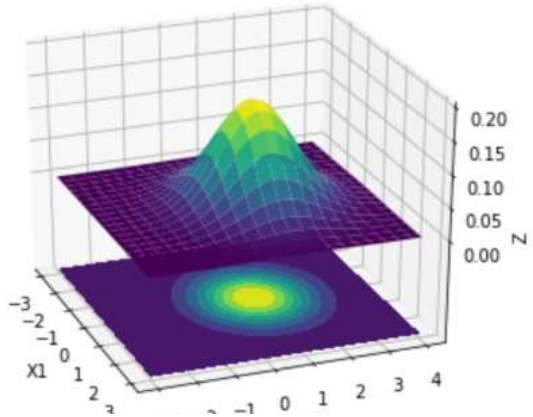
- If the off diagonals of the covariance matrix (see pic on right) are zeros, then it means that there is no covariance detected or in other words the individual features of the dataset are independent of each other.



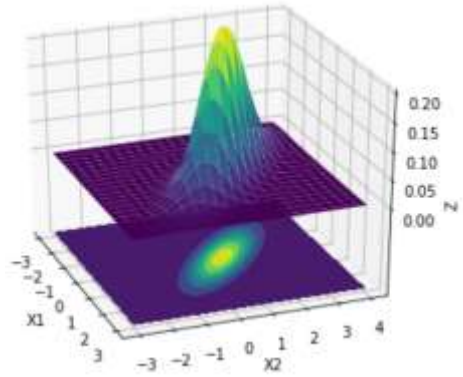
$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Varying the covariance matrix varies the orientation

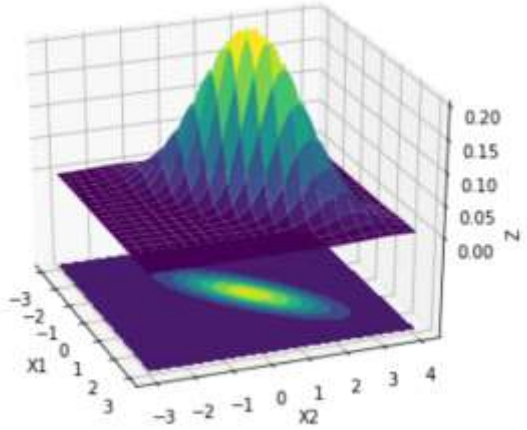
- When $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$



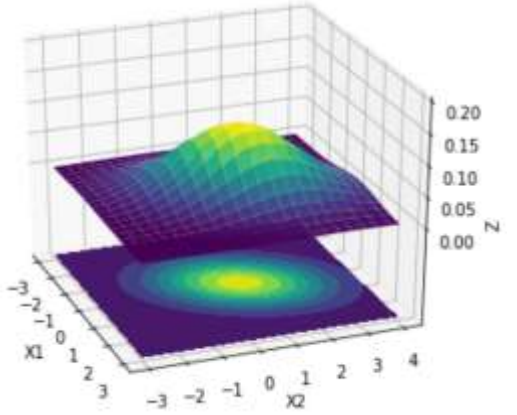
- When $\Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}$



- When $\Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$

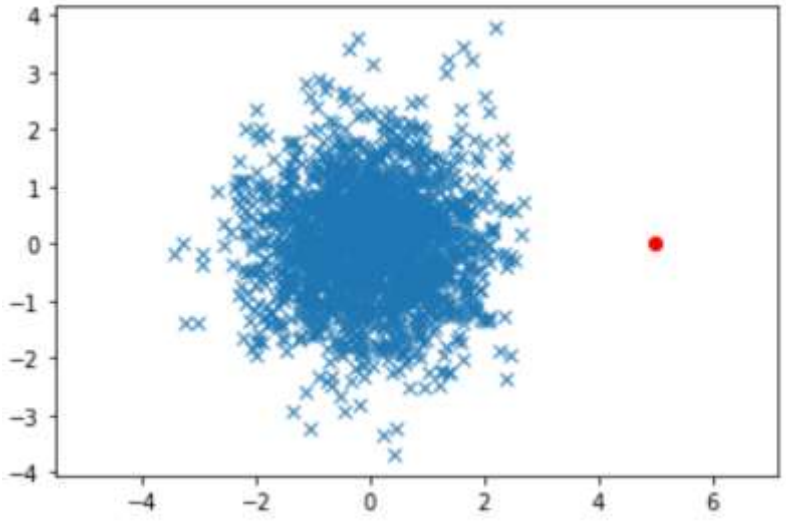


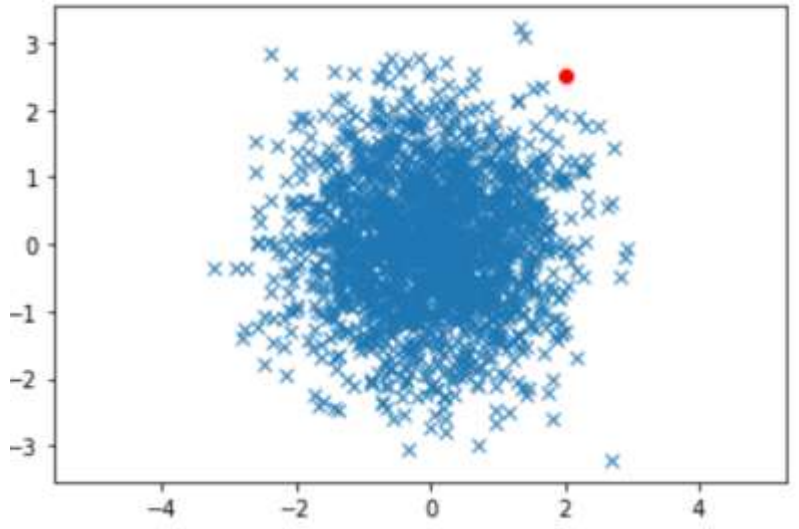
- When $\Sigma = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix}$



Intuition behind Anomaly detection using Gaussians

- This is mainly an unsupervised learning algorithm although sometimes it seems like a hybrid.
- Anomaly detection is the problem of identifying data points that don't conform to expected (normal) behavior.
- Applications: Fraud detection, Machine failure detection, looking for anomalous behavior in data center servers, etc.



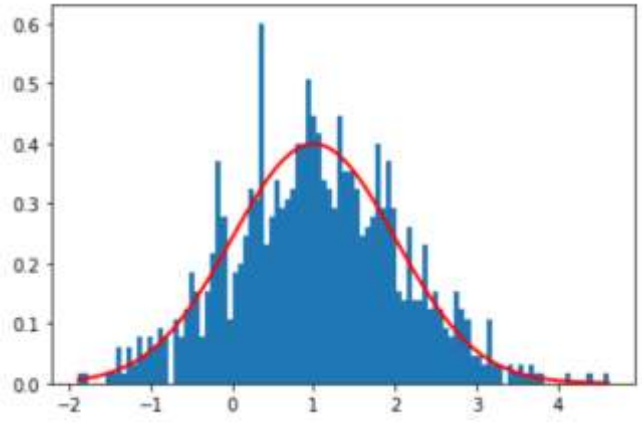


The Mathematical Approach

- Let us consider a univariate dataset, dataset with single feature $= \{x_1, x_2, \ldots\ldots x_n\}$
- We are free to assume that they are values belonging to a Gaussian distribution.
- If that is true, we can calculate the mean and variance as - ,

$$\mu = \frac{1}{m}\Sigma x_i \quad \sigma^2 = \frac{1}{m}\Sigma(x_i - \mu)^2$$

- This we can compute and get the parameters of the Gaussian.



Probability density

- Extending the concept from the previous slide, if we have 2 features in our dataset (bivariate), each m samples, we can assume that each of them come from a gaussian distribution.

- Therefore, each will have their mean and variance which can be computed.

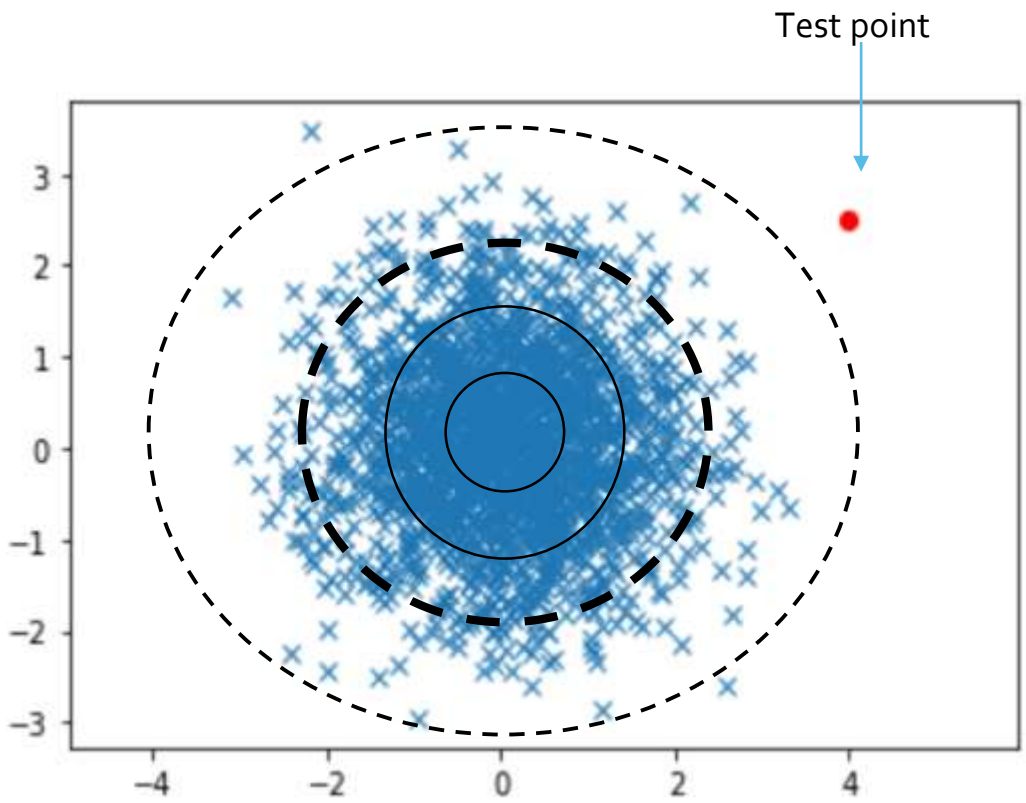
- These individual Gaussians are probability distributions of each of the features.

- If that is true, we can calculate the mean and variance of each of the features. ,
- Therefore, the total probability density of a data point in the dataset can be computed as :

$$p(x) = p(X_1; \mu_1, \sigma_1^2) * p(X_2; \mu_2, \sigma_2^2)$$

How can anomalies be detected?

- Let a dataset have 2 features {X1,X2}.
- We calculate and find that μ1 =0 and μ2 =0 and σ1 = 2, σ2 = 1.
- Plot of the dataset looks as in the picture.
- We introduce a parameter called ε, say 0.02. We set this value.
- We calculate p(X1) * p(X2) for each data point and compare it with ε .
- If the probability density of Test point < ε then it is an anomaly, else it is not.

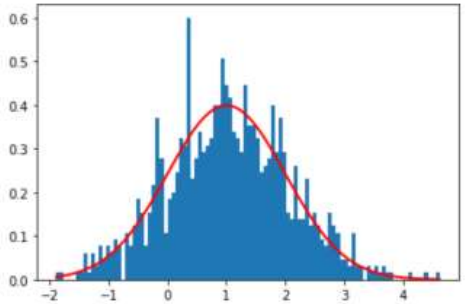


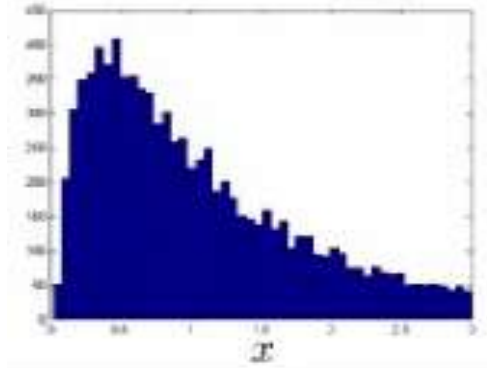
Anomaly detection algorithm

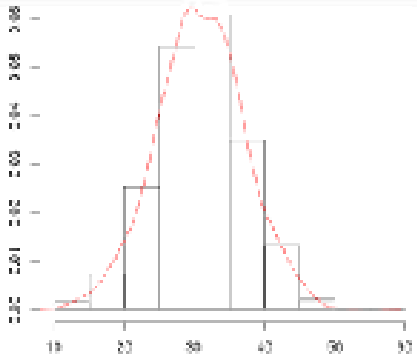
- Choose features x_i that you might think are indications of anomalies.
- Fit parameters μ and σ for each feature.
- Choose a threshold value, ε, a probability density value, below which represents anomalies.
- Given a new example, compute $p(x)$ for that example.
- An anomaly if $p(x) < \varepsilon$

Dealing with non-Gaussian data

- When we have a random sample of data and we believe it comes from a Gaussian distribution, we can check that by plotting its histogram.
- Picture on the right shows the histogram and also shows that it seems to fit a gaussian very well.
- When we plot histograms of such samples, typically the features of a dataset, we could also get a histogram that does not look like a gaussian.
- In such case, a pre-processing step would be to convert it to a gaussian to the extent possible.
- This can be done using power transformations like log(x) or log(x+c) or x^2 or x^0.5 and so on.
- It is therefore always a good idea to check the nature of a distribution.







Setting up an anomaly detection system

Building a model for use with the example of auto engines:

- Let's say we have data for 10000 good engines (where y=0) and 20 anomalous engines (y=1).
- Take data of 6000 good engines into a training set.
- Take data of 2000 good engines and 10 anomalous engines into CV set.
- Take the balance data of 2000 good engines and 10 anomalous one's into a Test set.
- Follow the method below to build the model:
  a. Fit the model to the training set by determining the appropriate value of ε
  b. See if it works well on the CV set - tweak ε till it fits.
  c. Test it on the test set - you should get good results.
  d. Measure the accuracy of the model by measuring True and False, Positives and Negatives.

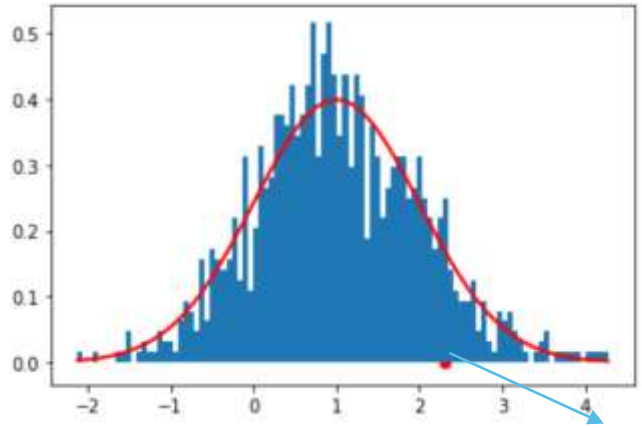
Adding features that influence anomalies

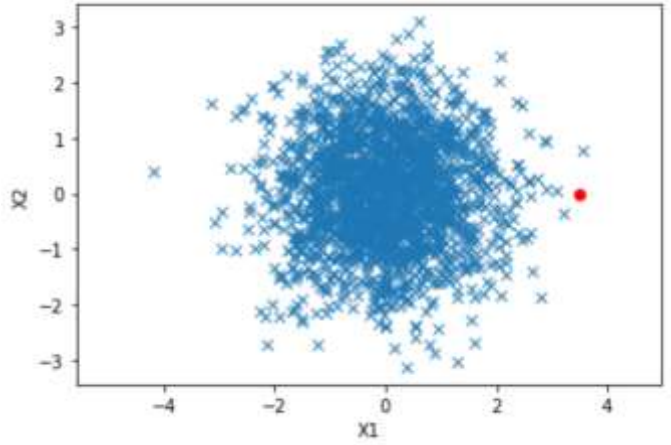
- When we set up an anomaly detection system, we want large probability density for normal examples and small one's for abnormal examples.
- A common problem that occurs is p(x) lies in the normal range for both cases. This happens either because :

  > The particular fault is new and never seen before.

  > We have not considered all features that flag that anomaly and that is usually the case.

- Let's assume we have a dataset with one feature X1 and is distributed as shown.
- The data point marked in red is an anomaly but seems to be in the normal range. It is very likely that there is a 2nd feature which influences this but has not been considered





Anomaly would have been detected with X_2 included.

Derived Features to detect hidden anomalies

Monitoring computers in a data center:

Features that are typically considered are:

x1 = memory use of computers

x2 = number of disk accesses/sec

x3 = CPU load
x4 = network traffic

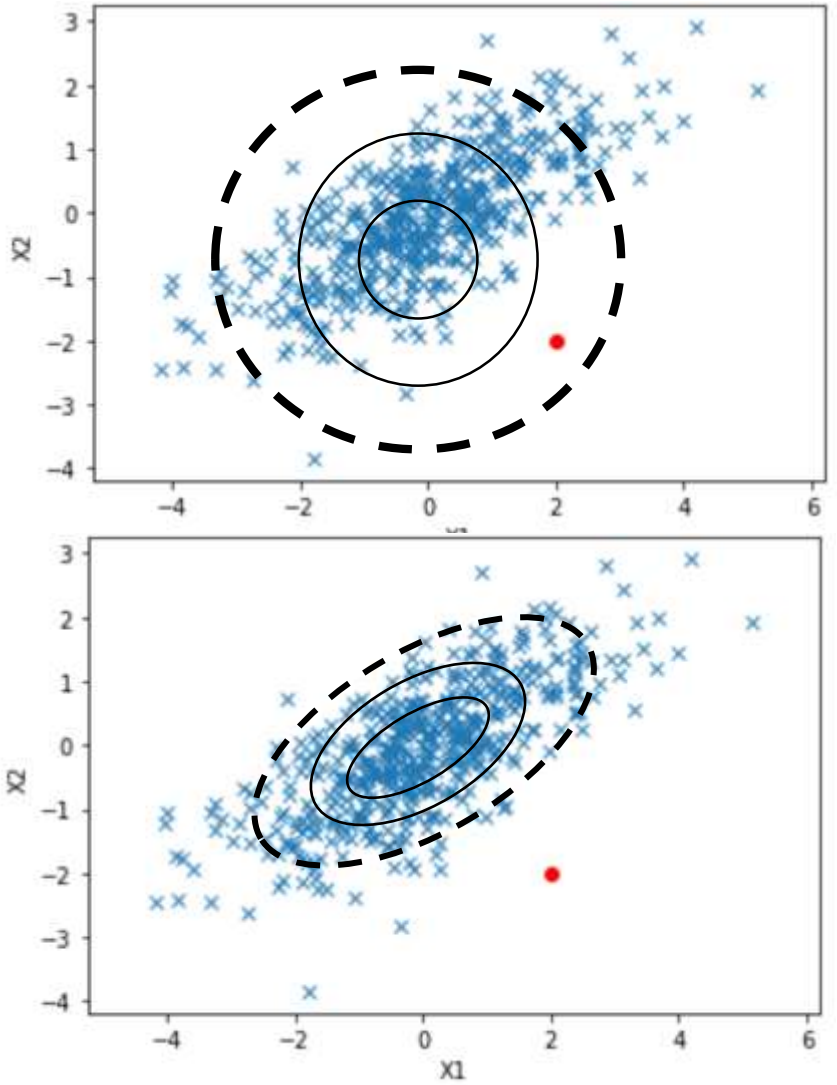
If an anomaly occurs like a program is stuck in an infinite loop - the CPU load will be high, but rest of the parameters may remain normal. This high CPU load may not create a threshold value that flags it as an anomaly. To avoid this from happening, we can create new features which look as follows:

x5 = CPU load / network traffic
x6 = $(\text{CPU load})^2$ / network traffic

These features will also be large when CPU load is large, and the error can be detected by the model easily.

Case for Multivariate Gaussian

- The anomaly example on the right is clearly visible as an anomaly in the picture.
- However, it did not get flagged as an anomaly as the system monitoring it , was based on the univariate model and therefore a product of individual probabilities were calculated.
- Univariate probability distributions are always aligned with the axis, x and y in this case. This leads to probability contours as shown superimposed on the data. Of course, the anomaly point looks like within threshold for such a system.
- If we use the multivariate version, we are now able to change the orientations of the probability contours as shown, because we are taking covariance into consideration, and hence can easily detect the anomaly.
- So, it is important to understand such orientations.



Implementing the multivariate model

- To implement the Multivariate model (when we have 2 or more features) the following are to be calculated:

  The mean of the features:

$$\mu = \frac{1}{m}\Sigma x_i$$

  The covariance Σ between the features:

$$\Sigma = \frac{1}{m}\Sigma_i (x^i - \mu)(x^i - \mu)^T$$

- We then incorporate the elements in the Multi-Variate Gaussian formula:

$$p(X;\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^n|\Sigma|}} \exp(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu))$$

- | Σ | is called the determinant of the covariance matrix and Σ^{-1} is the inverse of the covariance matrix.

- The univariate model or the independent variables model is just a special case of the multivariate with a constraint that the off diagonals of the covariance matrix are always zeros.

Computations can all be done using the "np.linalg" library and the SciPy library

```
# Calculate covariance of a dataset
sigma = np.cov(x['x1'],x['x2’])

# Calculate the determinant of the covariance matrix
sigdet = np.linalg.det(sigma)

# Calculate the inverse of the covariance matrix
siginv = np.linalg.pinv(sigma)
```

Once we have these and everything is in matrix form, we can compute:

$$p(X;\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^n|\Sigma|)}}\exp(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu))$$

Original model versus Multivariate Gaussian

Original Model (Univariate based)

$$p(x) = p(X_1; \mu_1, \sigma_1^2) * p(X_2; \mu_2, \sigma_2^2)$$

- Manually create features when needed for unusual combinations.
- Computationally cheaper, scales well even for large values of features (10000 and greater too).
- Works well for large and small train sizes of data.

Multivariate Gaussian

$$p(X; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp(-\frac{1}{2}(X-\mu)^T)|\Sigma|^-1(X-\mu)$$

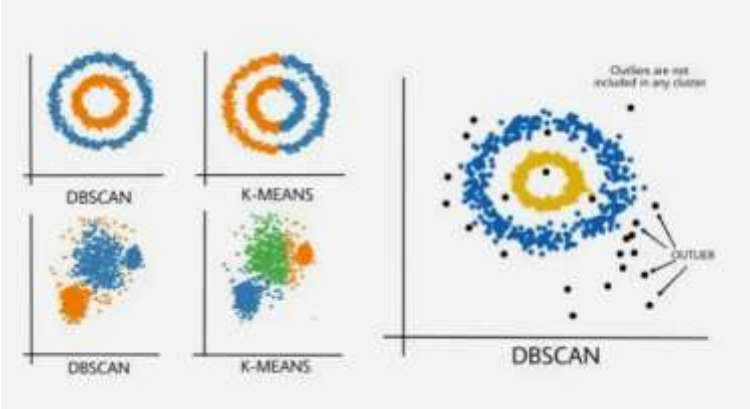
- Automatically captures correlations between features because of the covariance entity.
- Computationally heavy – calculating the inverse of a large matrix can be expensive.
- Number of samples must be greater than the number of features else we cannot find the inverse of Sigma mathematically. This also true if we accidentally have redundant features.

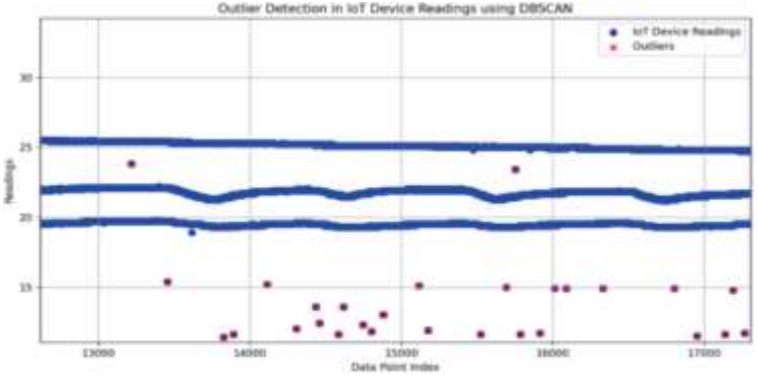
Coded Example

2. DBSCAN

DBSCAN -Density-Based Spatial Clustering of Applications with Noise

- **Concept :**
  - **Unsupervised** clustering algorithm.
  - Groups data based on **density**.
  - Identifies **anomalies** as points in low-density regions.
- **Intuition**
  - **Core Idea:** Points in dense regions form clusters;
  - Sparse points are anomalies.
  - **No Need for Cluster Count:** Unlike k-means, DBSCAN does not require specifying the number of clusters.
  - **Handles Noise:** Automatically labels outliers as "noise."



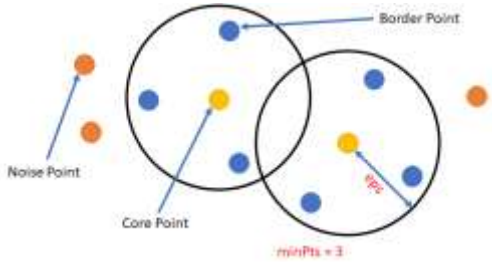


DBSCAN – How it works?

Key Parameters

- **eps (ε):** Maximum distance between two points to be considered neighbors.
- **min_samples:** Minimum number of points required to form a dense region (cluster).

Point Types



- **Core Point:** Has at least min_samples neighbors within eps.
- **Border Point:** Within eps of a core point but has fewer than min_samples neighbors.
- **Noise/Anomaly:** Not a core or border point.

Algorithm Steps

- **Randomly select** a point.
- If it's a **core point,** expand the cluster by adding all density-reachable points.
- Repeat until all points are visited.

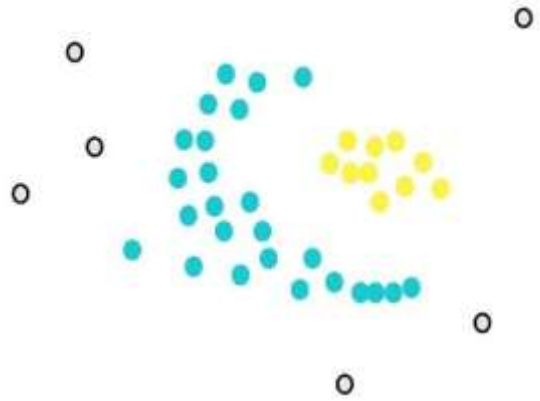
DBSCAN – Math and Visualization

Math

- **Density-Reachable:**
  - A point p is density-reachable from q if there's a path of core points from q to p, each within eps.
- **Cluster Formation:**
  - A cluster is a set of density-connected points (core + border points).

Visualization

- **Diagram:**
  - Show clusters as dense regions.
  - Highlight noise points outside clusters.



Anomaly Detection

- Points labeled as **noise** (-1) are anomalies.

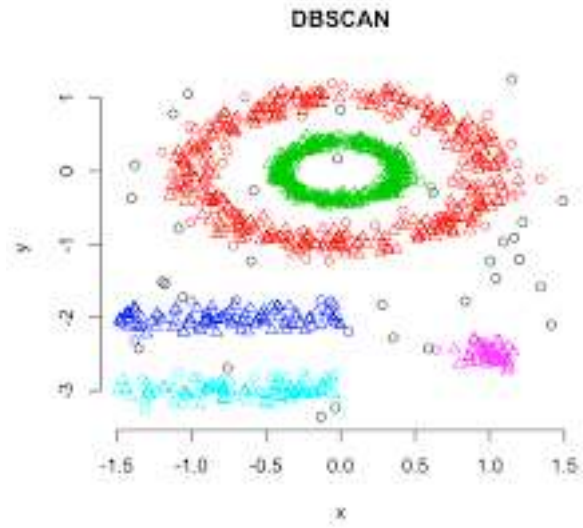
DBSCAN – Strengths and limitations

Strengths

- **No Need for Cluster Count:** Automatically determines clusters.
- **Handles Noise:** Explicitly labels outliers.
- **Works with Arbitrary Shapes:** Can detect non-spherical clusters.

Limitations

- **Parameter Sensitivity:** Performance depends on eps and min_samples.
- **Scalability:** Can be slow for large datasets.
- **Density Variation:** Struggles with varying densities.



DBSCAN – When to use?

Use Cases:

- Anomaly detection in **spatial or geometric data** (e.g., sensor readings, geolocation).
- When clusters have **arbitrary shapes** or noise is present.

Avoid When:

- Clusters have **similar densities** or data is high-dimensional.

DBSCAN is like finding crowded neighborhoods in a city; anomalies are isolated houses

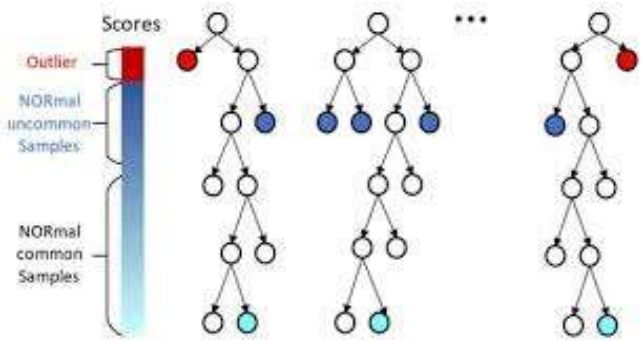
Coded Example

3. Isolation Forest

Isolation Forest (iForest)

Intuition

- **Isolation Trees (iTrees):** The algorithm constructs an ensemble of isolation trees.
- **Idea:** Anomalies are "few and different" → easier to isolate.
- **Isolation:** Randomly split data until all points are isolated.
- **Anomalies:** Isolated with fewer splits (shorter path lengths).



How It Works and Math

- Build an ensemble of isolation trees.
- Compute **path length** for each point - $h(x)$ - is the number of edges traversed from the root node to the leaf node where the data point is isolated.
- Anomaly score: $s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$
- $E(h(x))$:Average path length of 'x' across all trees
- $c(n)$:Normalization factor based on sample size of n (Simply taking the average, E[h(x)], will lead to issues as there are large and small trees)

Isolation Forest Visualization and anomaly scores

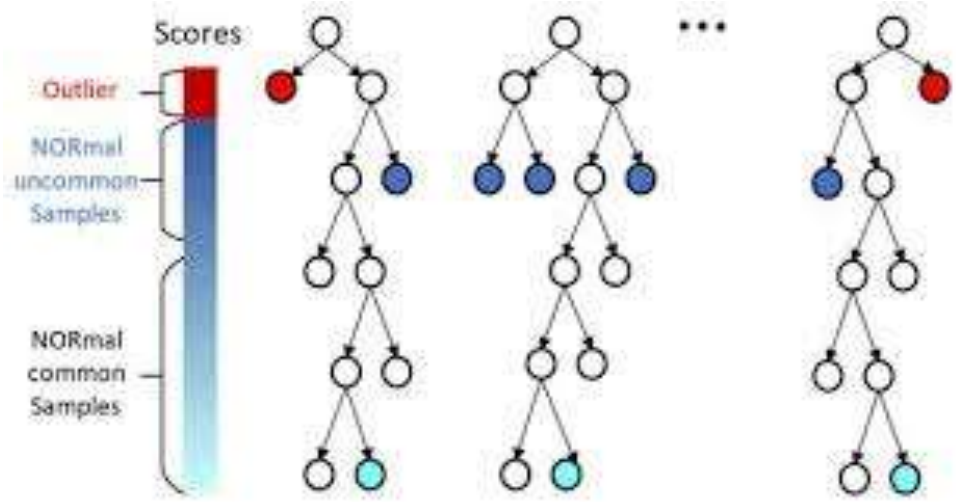
Diagram:

- Tree structure with random splits.
- Plot of anomaly scores vs. data points.

Anomaly score:

- **High Anomaly Score (closer to 1):** Indicates a higher likelihood of the data point being an anomaly, as it was isolated with a relatively short path length.
- **Low Anomaly Score (closer to 0):** Suggests the data point is likely normal, as it required a longer path length to be isolated.
- **Score around 0.5:** Implies the data point's isolation characteristics are similar to those of a typical inlier.

Isolation Forest algorithm quantifies how "different" a data point is from its neighbors by measuring the ease with which it can be separated, assigning a score that directly reflects this distinction.



Coded Example

3. AutoEncoders

Auto Encoder

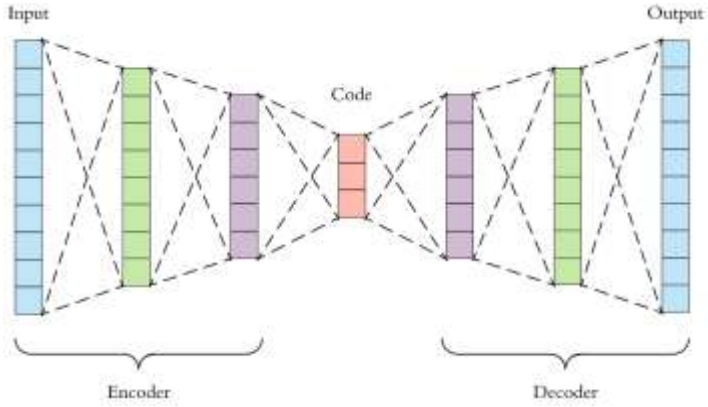
Intuition

- **Idea:** Learn to compress and reconstruct normal data.
- **Anomalies:** Data points that are poorly reconstructed.

How It Works

- **Encoder:** Compresses input into a smaller latent space.
- **Decoder:** Reconstructs the original input from the latent space.
- **Training**:

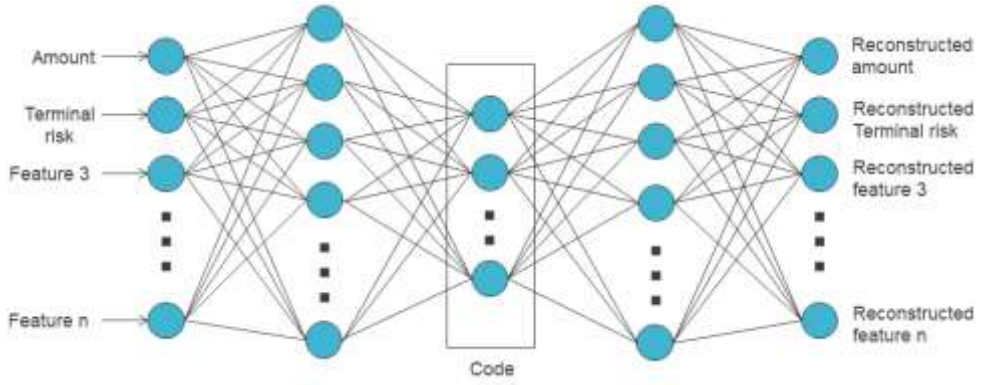
1. The network is trained by minimizing the "reconstruction error," which is the difference between the original input and the reconstructed output.
2. This is often done using a self-supervised approach, where the input is also the target output.
3. **Anomaly detection**: A well-trained autoencoder will have a low reconstruction error for normal data. When it encounters an anomaly, it will struggle to reconstruct it, resulting in a high error, which can be used to flag the anomaly.



Auto Encoder workings

Math

- **Input:** x
- **Latent Representation:** $z = f(x)$
- **Reconstruction:** $\hat{x} = g(z)$
- **Loss:** $\mathcal{L}(x, \hat{x}) = \| x - \hat{x} \|^2$

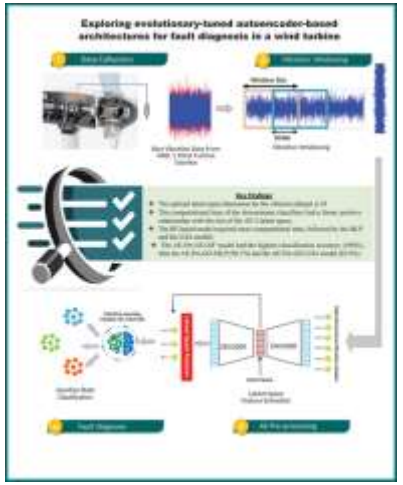


Anomaly Detection

- Compute reconstruction error for each point.
- If $\mathcal{L}(x, \hat{x}) > \tau$,flag as anomaly.

Strengths and limitations

- Works well for **high-dimensional** data (e.g., images, time series).
- Captures **non-linear** patterns.
- Requires tuning (e.g., network architecture, threshold).
- Computationally expensive.



Coded Example

AutoEncoder example observations

Reconstruction Error Plot Analysis

- **X-Axis: Data Point Index**
- Represents each data point in your dataset (normal and anomalous).
- **Y-Axis: Mean Squared Error (MSE)**
- Measures the difference between the original input and the reconstructed output.
- **Higher MSE** indicates the AutoEncoder struggled to reconstruct the data point, suggesting it's an anomaly.

Threshold (Red Dashed Line)

- **95th Percentile:** The red dashed line represents the threshold for anomaly detection (95th percentile of MSE).

Key Observations

- **Normal Data (First ~1950 Points)**
- **Low MSE:** The reconstruction error is consistently low for the first ~1950 data points.

Interpretation:

- **Anomalous Data (Last ~50 Points)**
- **High MSE:** The reconstruction error spikes sharply for the last ~50 data points.
- **Interpretation:** These points are **anomalies** (e.g., defective images or outliers). The AutoEncoder fails to reconstruct them accurately.
- **Effectiveness:** The threshold successfully separates normal data from anomalies.

Real-World Application

- Replace synthetic data with **real-world images** (e.g., from a manufacturing line).
- Use the trained AutoEncoder to **monitor live data** and flag anomalies in real-time.

Summary

DBSCAN:

- Works well for **1D/2D spatial data** (e.g., vibration, sensor readings).
- Anomalies are points labeled as -1.

Isolation Forest:

- Ideal for **tabular data** (e.g., temperature, pressure).
- Anomalies have **low scores** (e.g., below 5th percentile).

AutoEncoder (PyTorch):

- Best for **image data** (e.g., product defects).
- Anomalies have **high reconstruction error** (e.g., above 95th percentile).

Exercise 10 – Due by October 29th midnight

Exercise 10

Autoencoder:

- A defect detect dataset is provided and it has train and test data as images
- Work out an Autoencoder solution for it and identify the anomalous images in the test data

DBSCAN:

- Develop and detect anomalies in the dataset provided ('equipment_anomaly_dataset.csv')
- This dataset contains simulated data representing real-time monitoring of various industrial equipment, including turbines, compressors, and pumps.
- Each row in the dataset corresponds to a unique observation capturing key parameters such as temperature, pressure, vibration, and humidity. The dataset also includes information about the equipment type, location, and whether the equipment is classified as faulty.
- You may not need all the features in it so judiciously drop the ones you feel are of no value.
- Plot original and after anomaly detection plots (you may need to do PCA to 2-D).

End of Lesson